Faster Projection Based Methods for Circuit Level Verification

Chao Yan & Mark Greenstreet

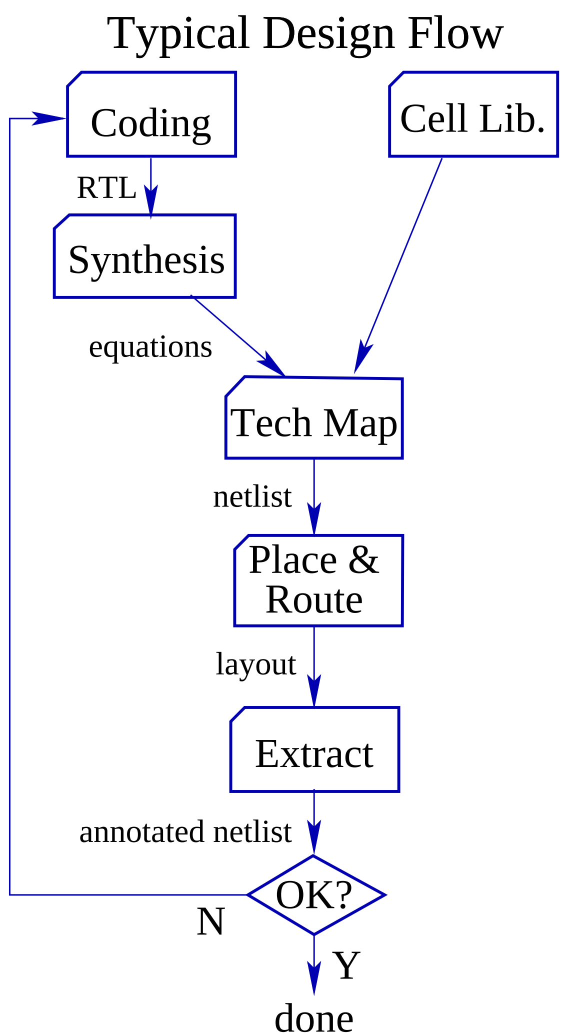
University of British Columbia

Overview

- Motivation
- Coho
  - Reachability analysis approach
  - Projection based representation of reachabel space
- Verification Example
  - Toggle Circuit
  - Verification Using Coho
- Performance Improvement
  - Faster LP solver
  - Improved Bloating and Time-Step
- ***Formal Verification of Digital Circuits Using SPICE-Level Models is Possible and Practical.***

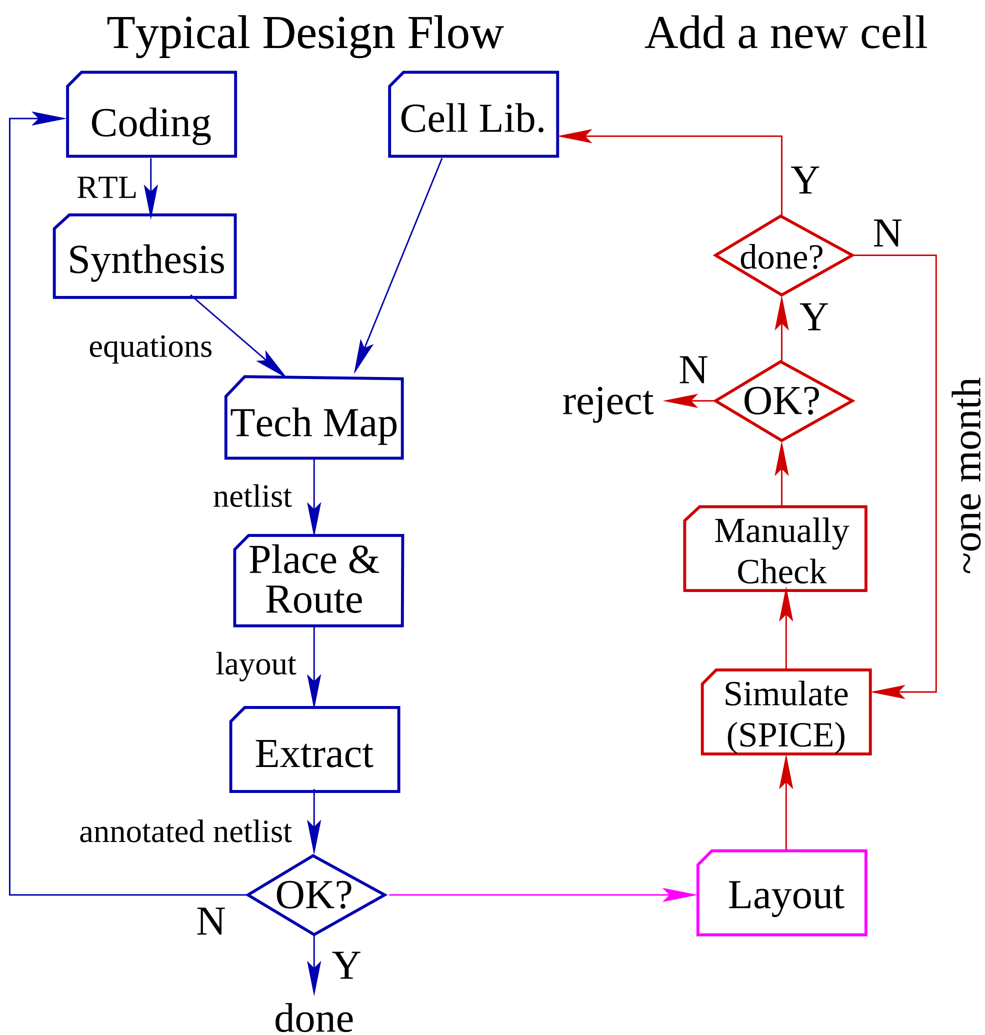
Motivation

- Design Flow



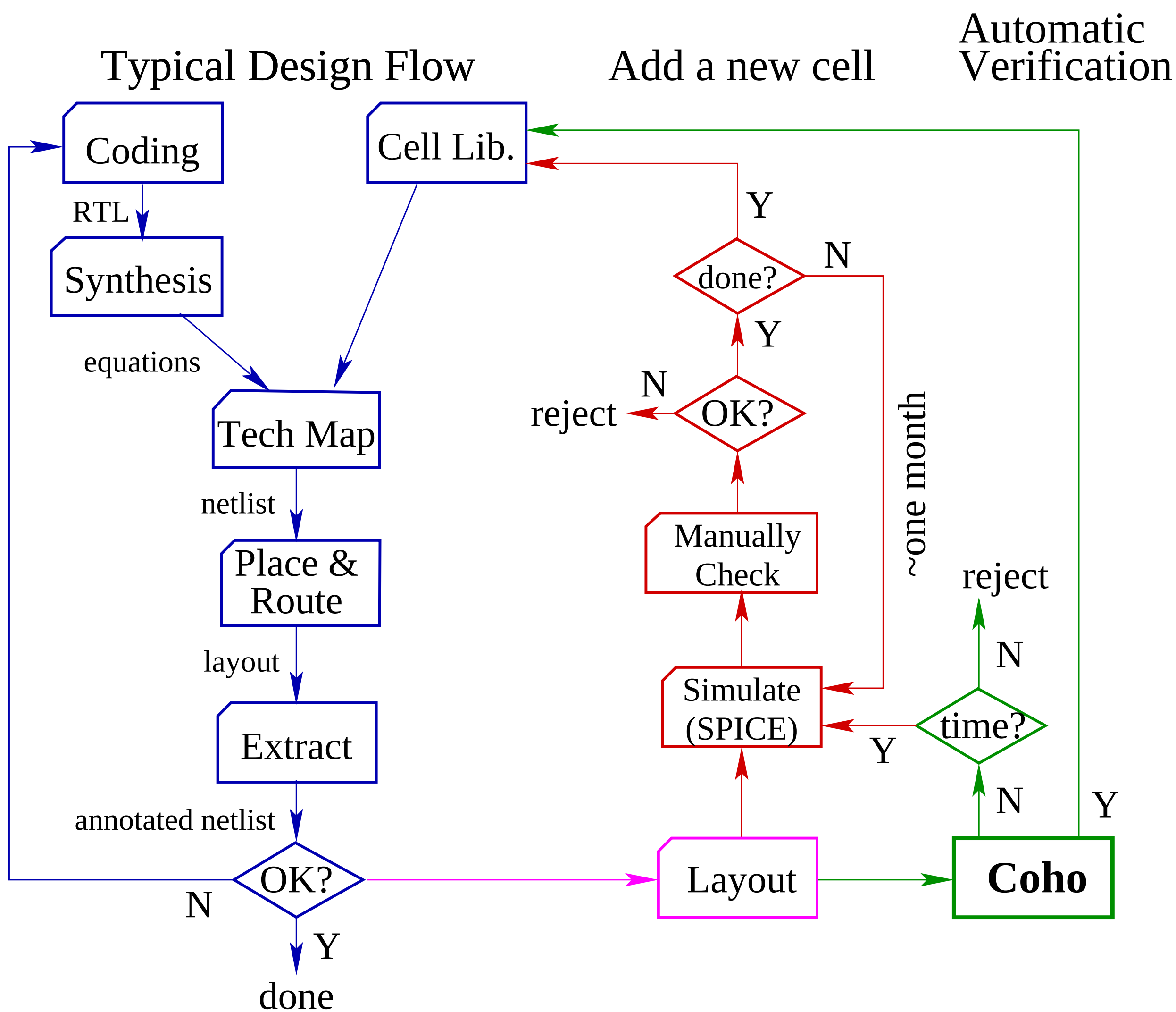
Motivation

- Design Flow



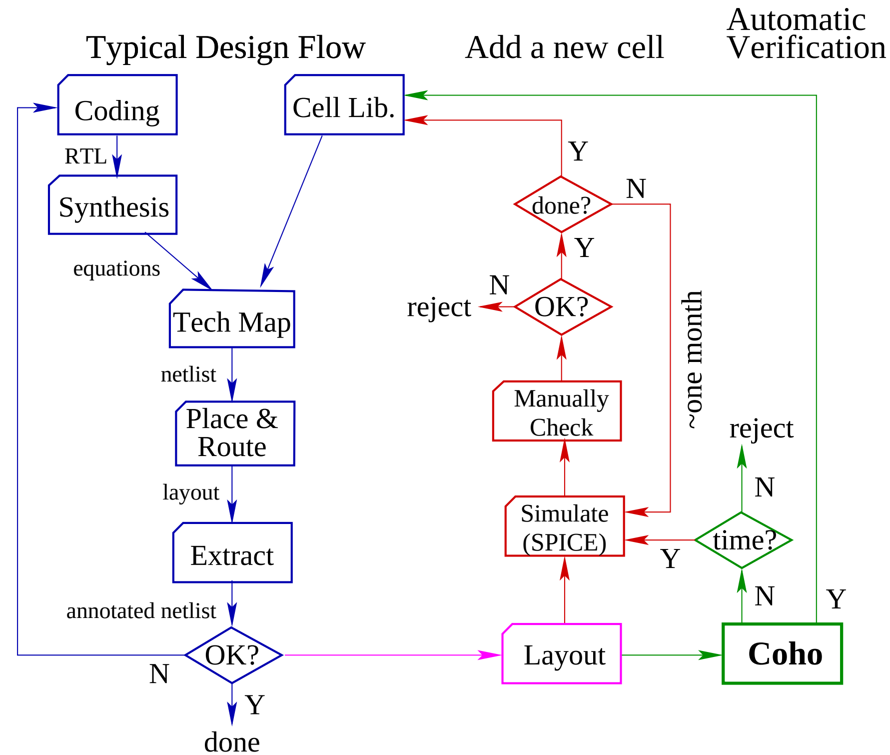
Motivation

- Design Flow



Motivation

- Design Flow
- Similar Problems
  - crosstalk analysis
  - power noise problems
  - leaky transistors
  - mixed-signal design



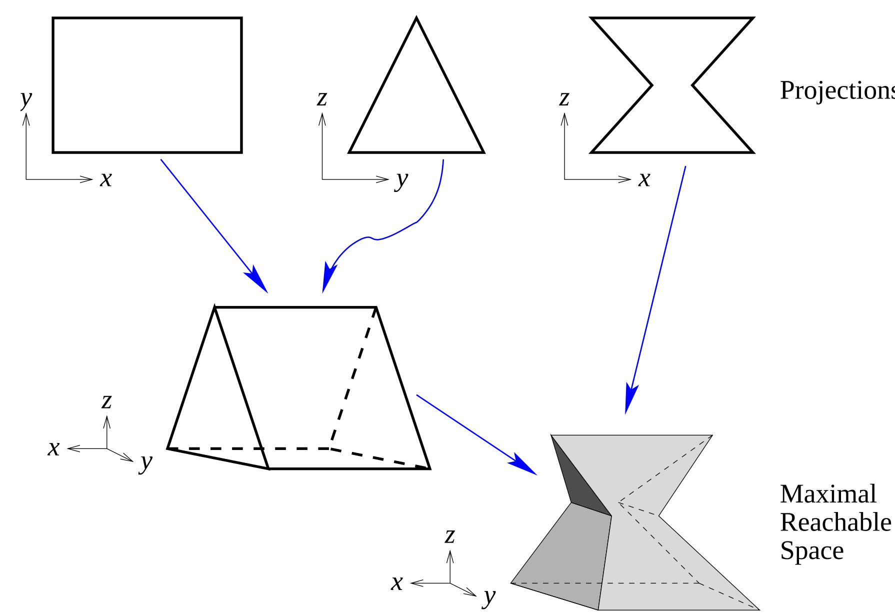
Coho

- A verification tool using reachablity 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)
- Approximate the ODEs in small neighborhoods by *linear differential inclusions*:

$$Ax + b - err \quad \leq \quad \dot{x} \quad \leq \quad Ax + b + err$$

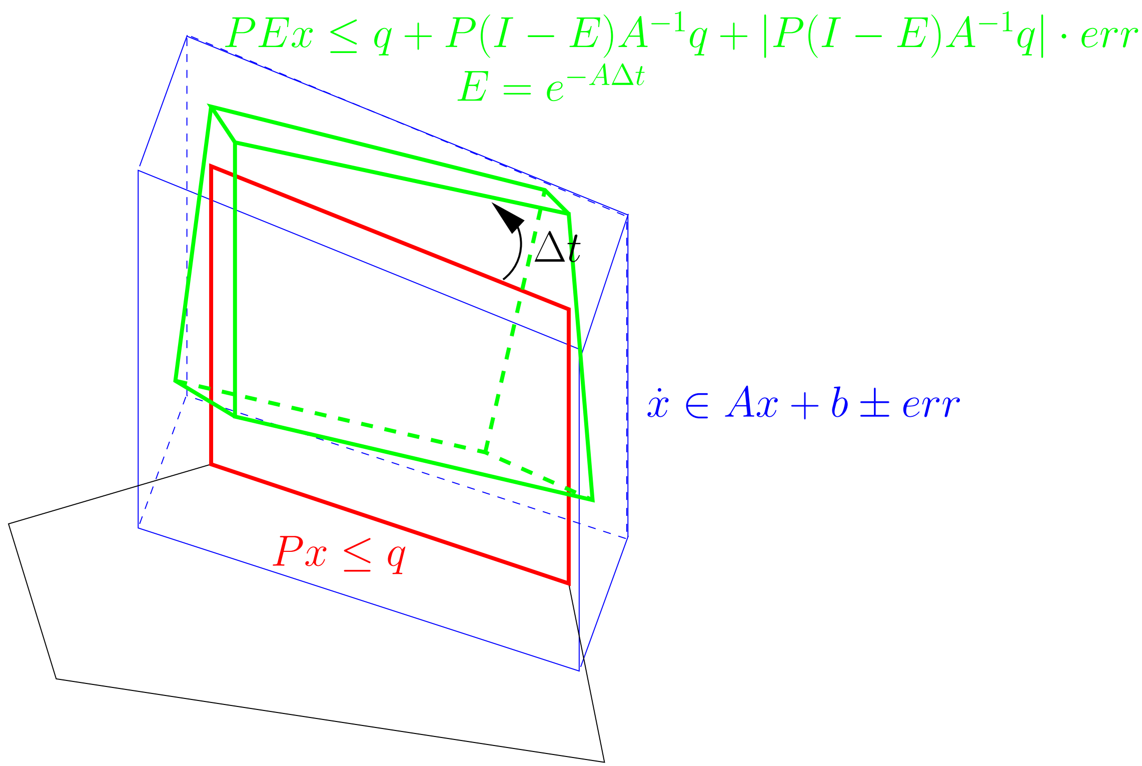
Representing the Reachable Space

- 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.
  - A projectagon is the feasible region of a *linear program* (LP).

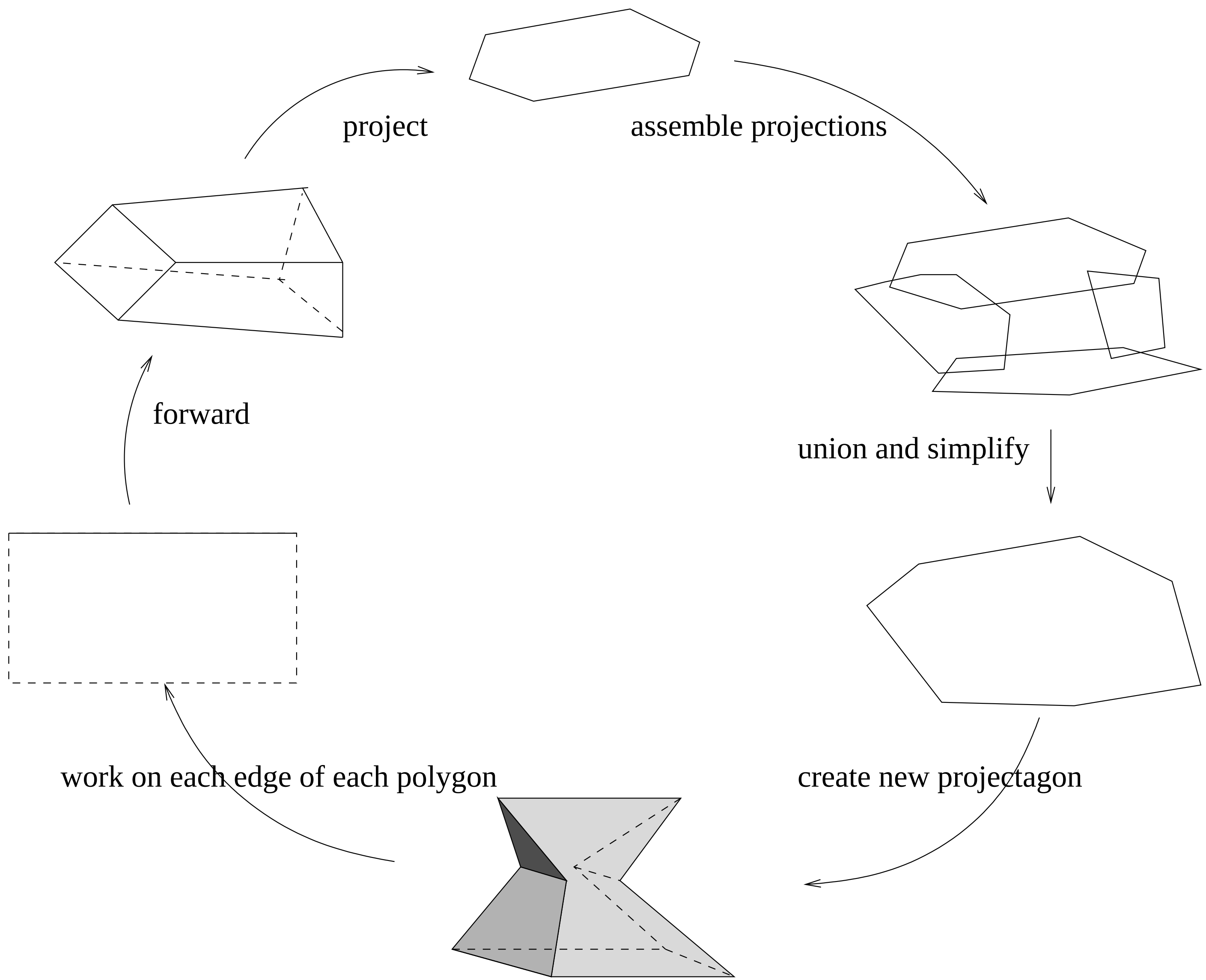


Reachability for Projectagons

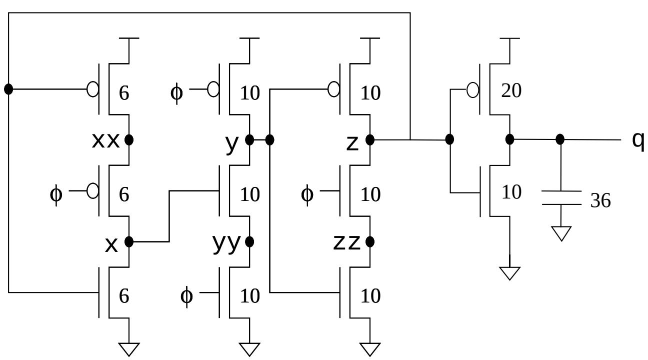
- Compute the reachable space contains all trajectories that start in a projectagon with the linearized model and time Δt
- Extremal trajectories original from projectagon faces.
- Coho computes time-advanced projectagons by working on one edge at a time.

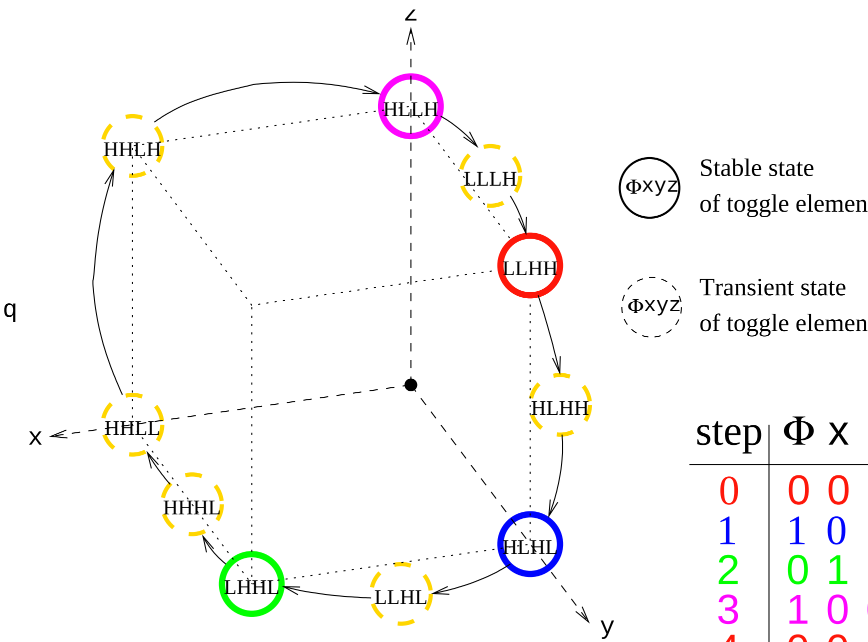


Basic Step of Coho

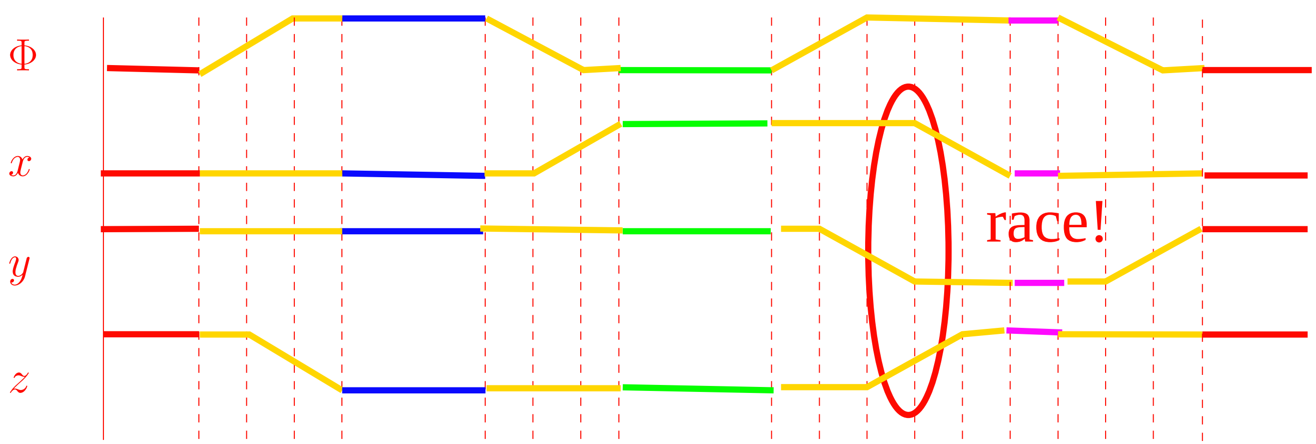


Toggle Circuit

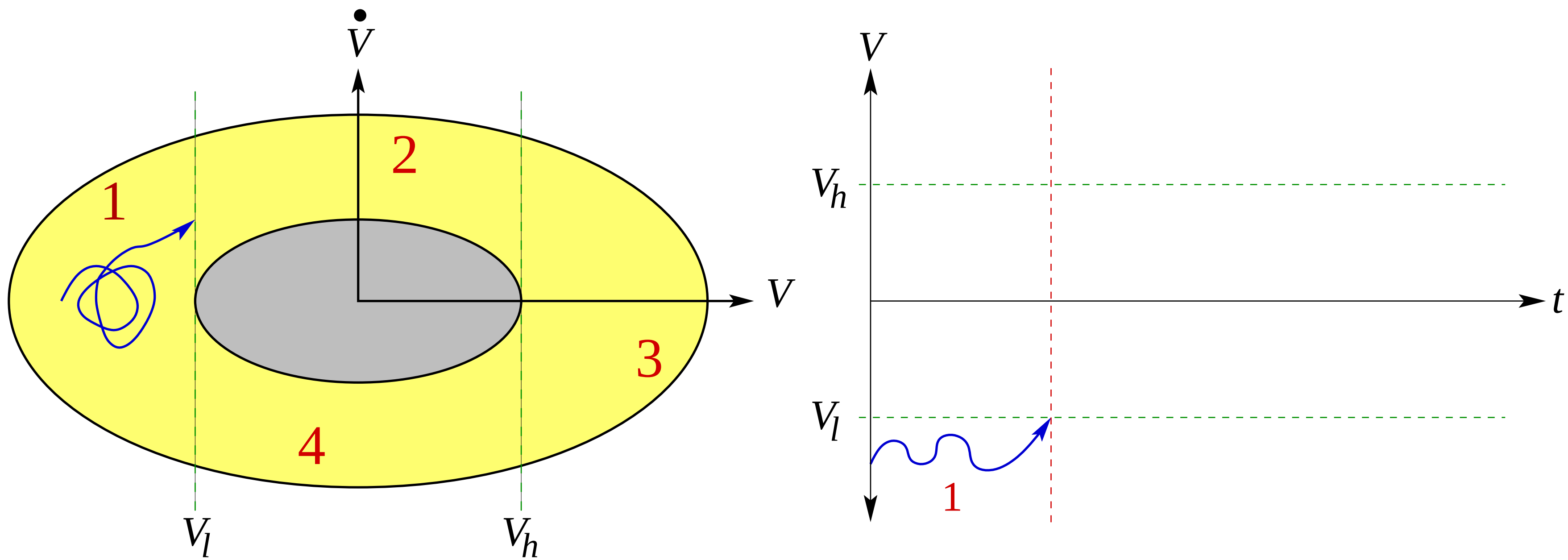




step	Φ	x	y	z
0	0	0	1	1
1	1	0	1	0
2	0	1	1	0
3	1	0	0	1
4	0	0	1	1

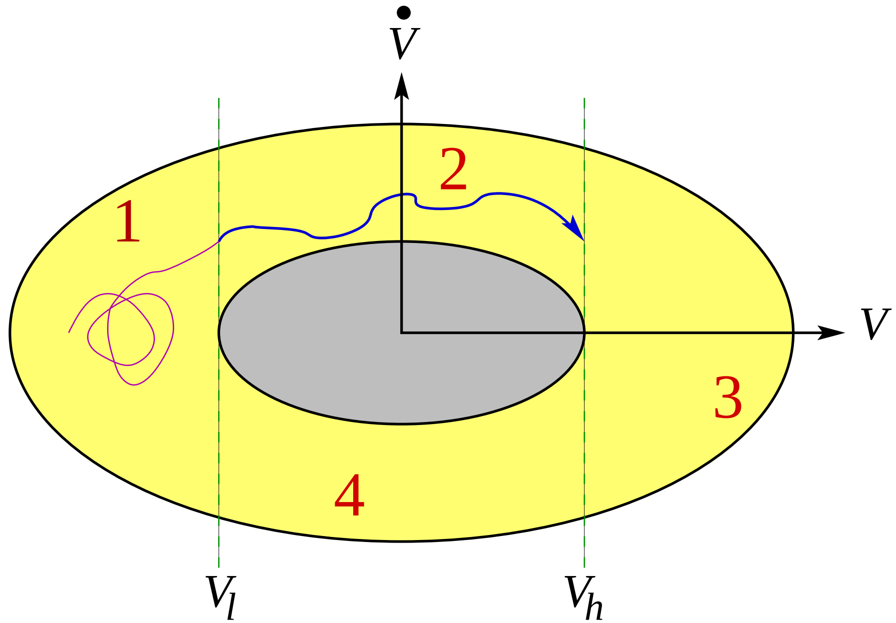


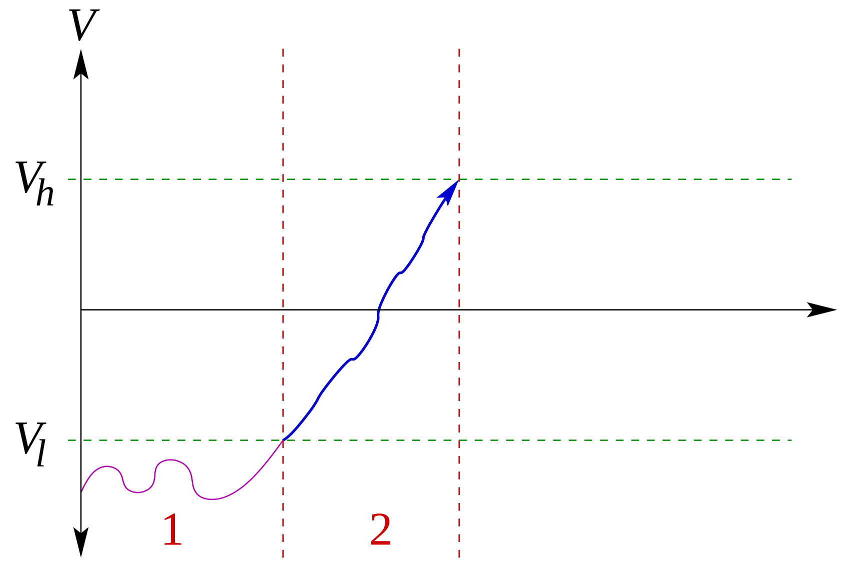
Brockett's Annulus



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

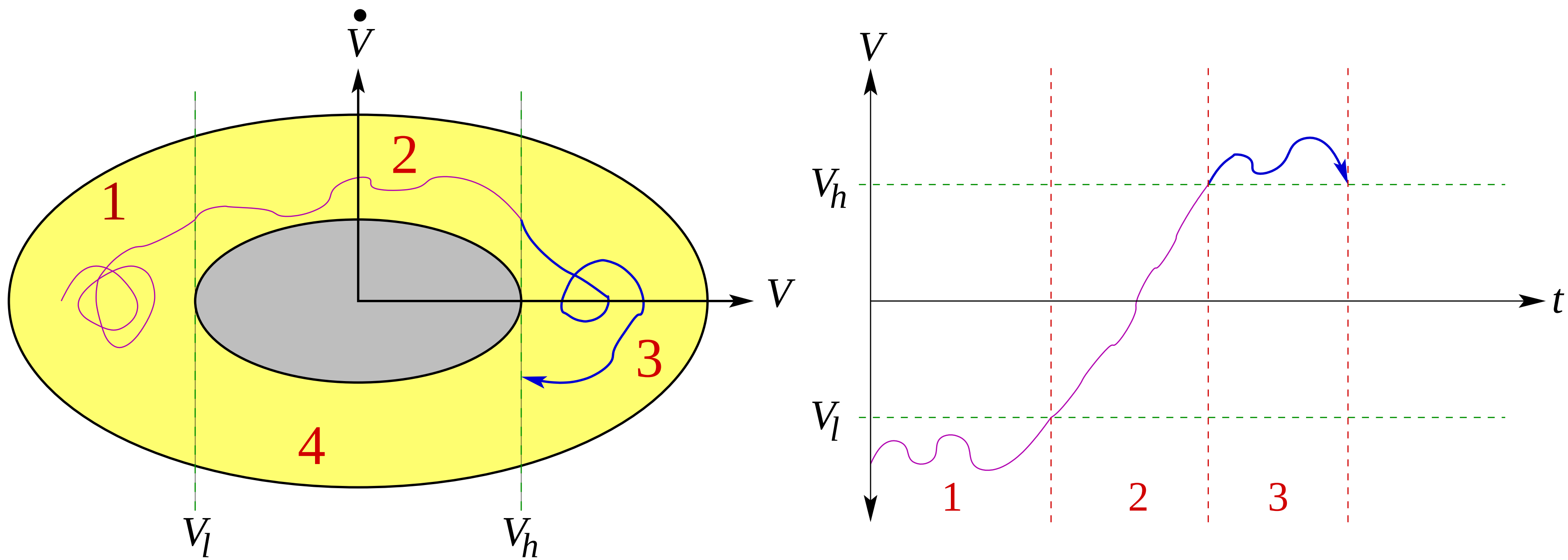
Brockett's Annulus





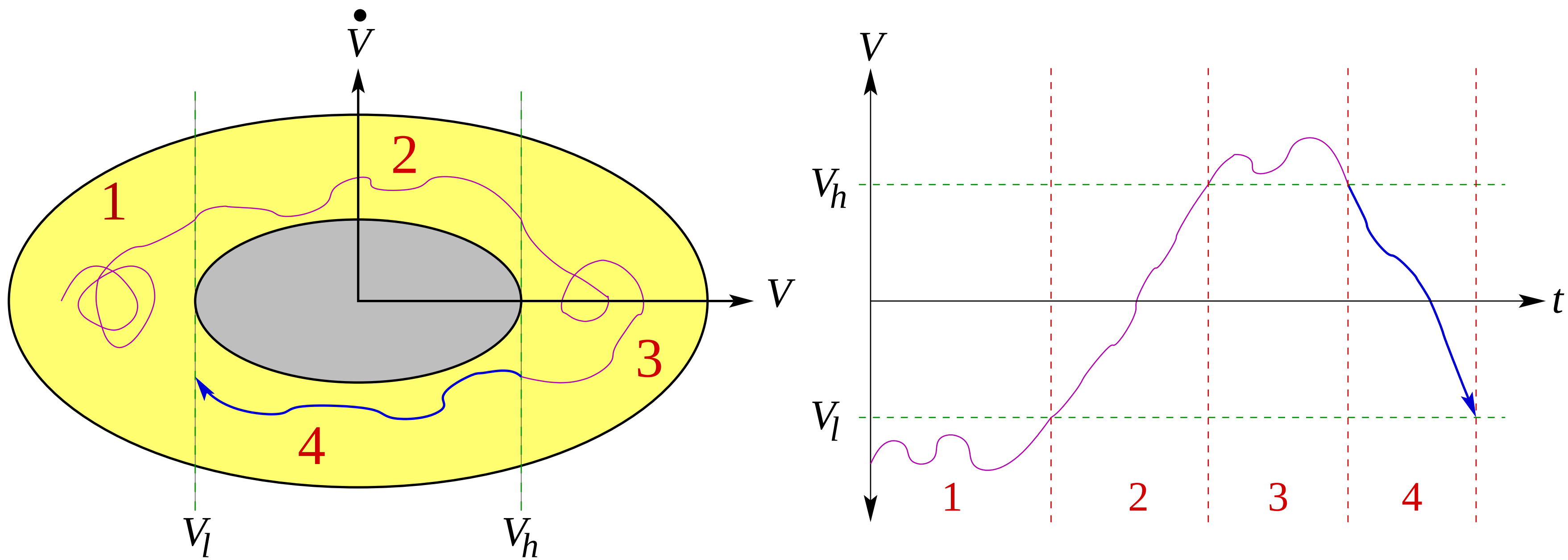
- Region 1 represents a logical low signal. The signal may wander in a small interval.
- Region 2 represents a monotonically rising signal.
- Region 3 represents a logical high signal.
- Region 4 represents a monotonically falling signal.
- Brockett's annulus allows entire families of signals to be specified.

Brockett's Annulus



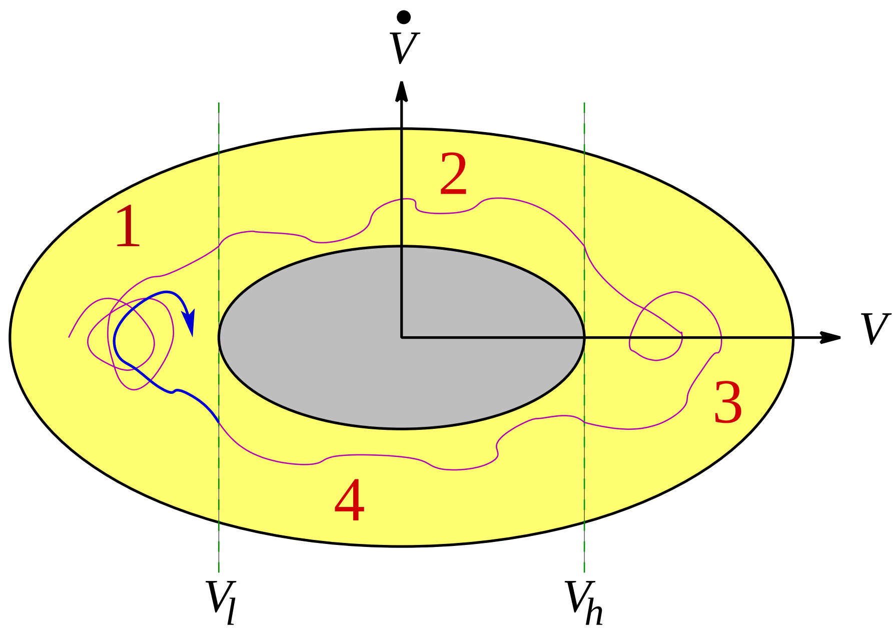
- Region 1 represents a logical low signal. The signal may wander in a small interval.
- Region 2 represents a monotonically rising signal.
- Region 3 represents a logical high signal.
- Region 4 represents a monotonically falling signal.
- Brockett's annulus allows entire families of signals to be specified.

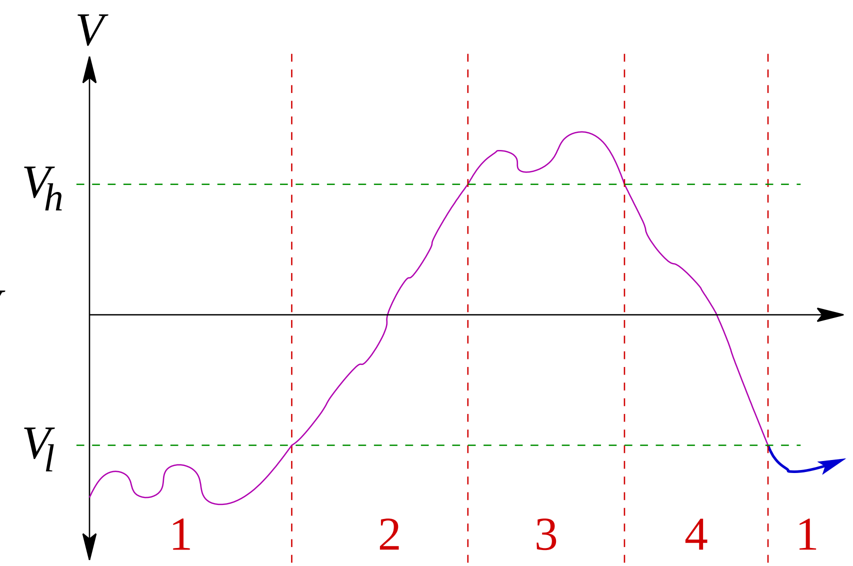
Brockett's Annulus



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

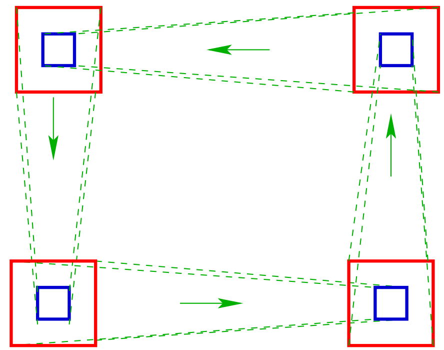
Brockett's Annulus





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

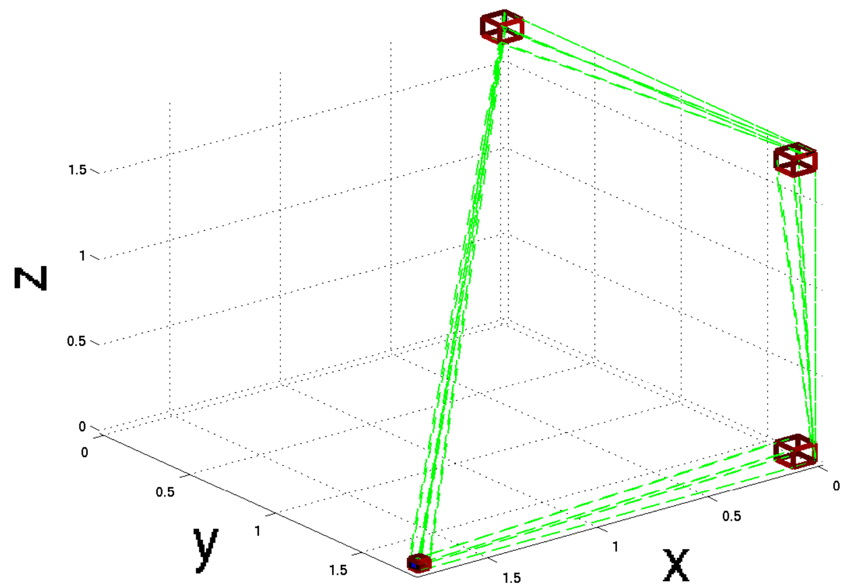
Reachable Space Computation



Separate computation into four phases

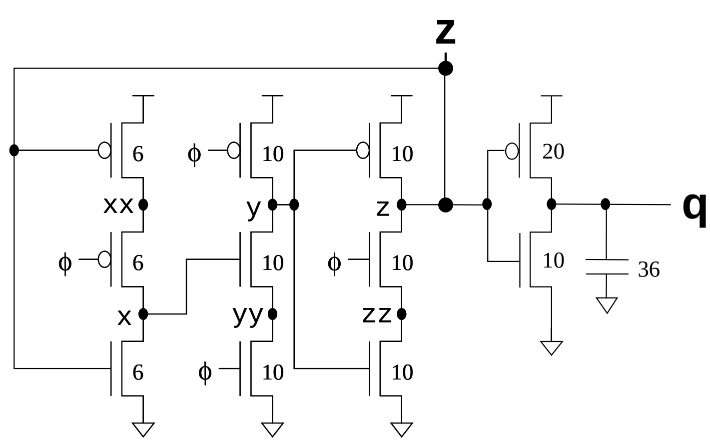
- One phase for each transition of Φ.
- Assume bounding hyperrectangle for start of phase.
- Establish bounding hyperrectangle at end of phase.
- Containment establishes invariant set.
- Allows parallel execution and parallel debugging.

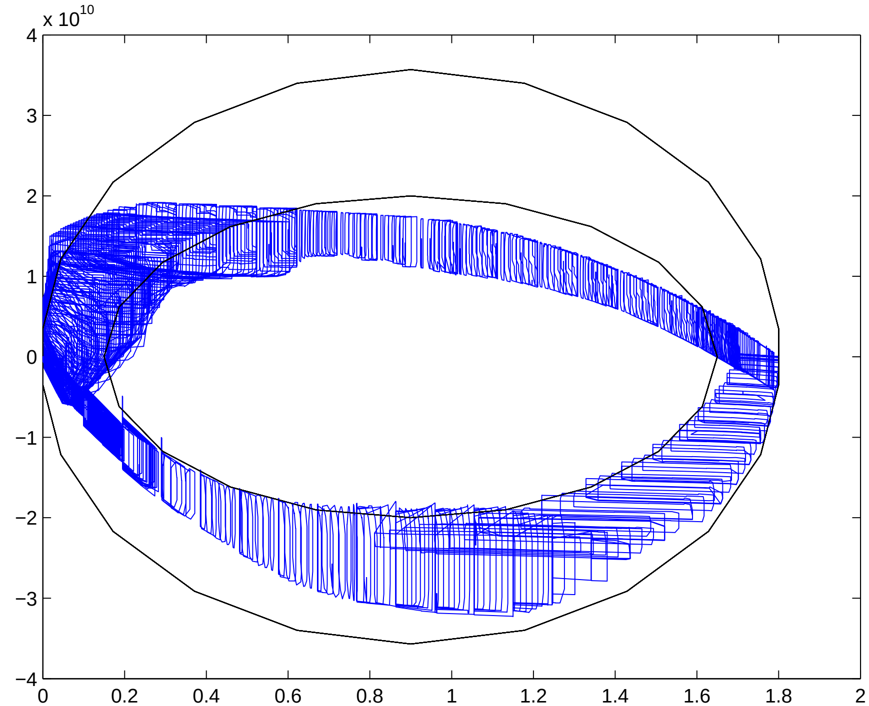
The Invariant Set



- Red: Hyperrectangles at beginning of each phase.
- Blue: Hyperrectangles at end of each phase.
- An invariant set with twice the period of the clock has been established.

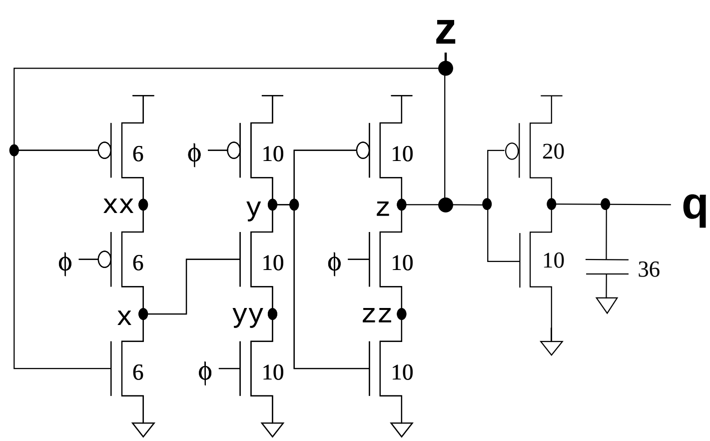
Brockett Ring at z

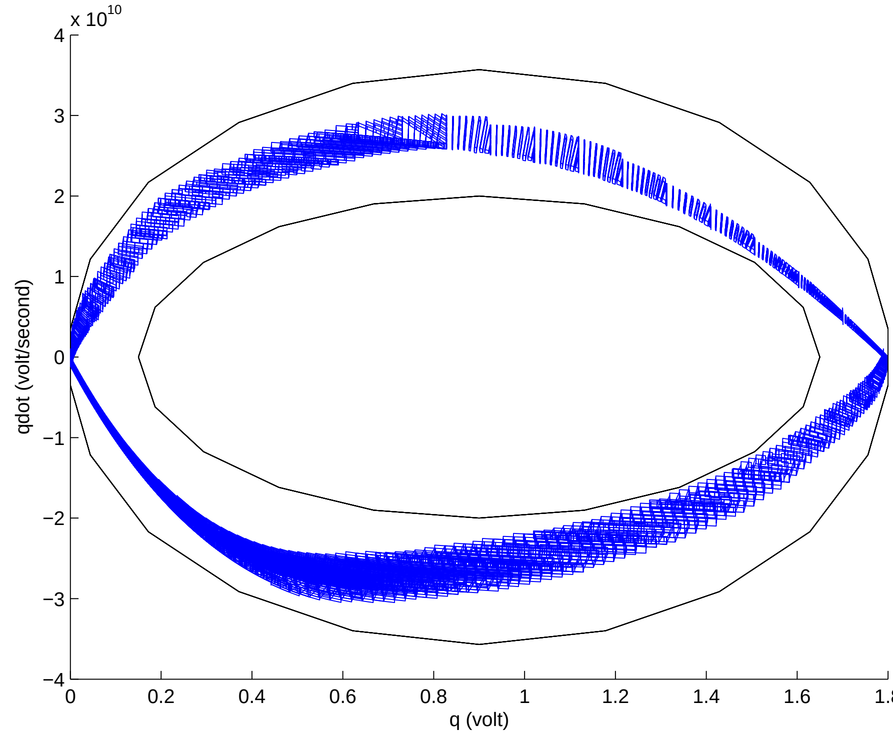




- Construct the brockett annulus for z, ignoring the inverter
- Perform a separate reachability analysis for the output inverter
- Arbitrary ripple counter

Brockett Ring at q



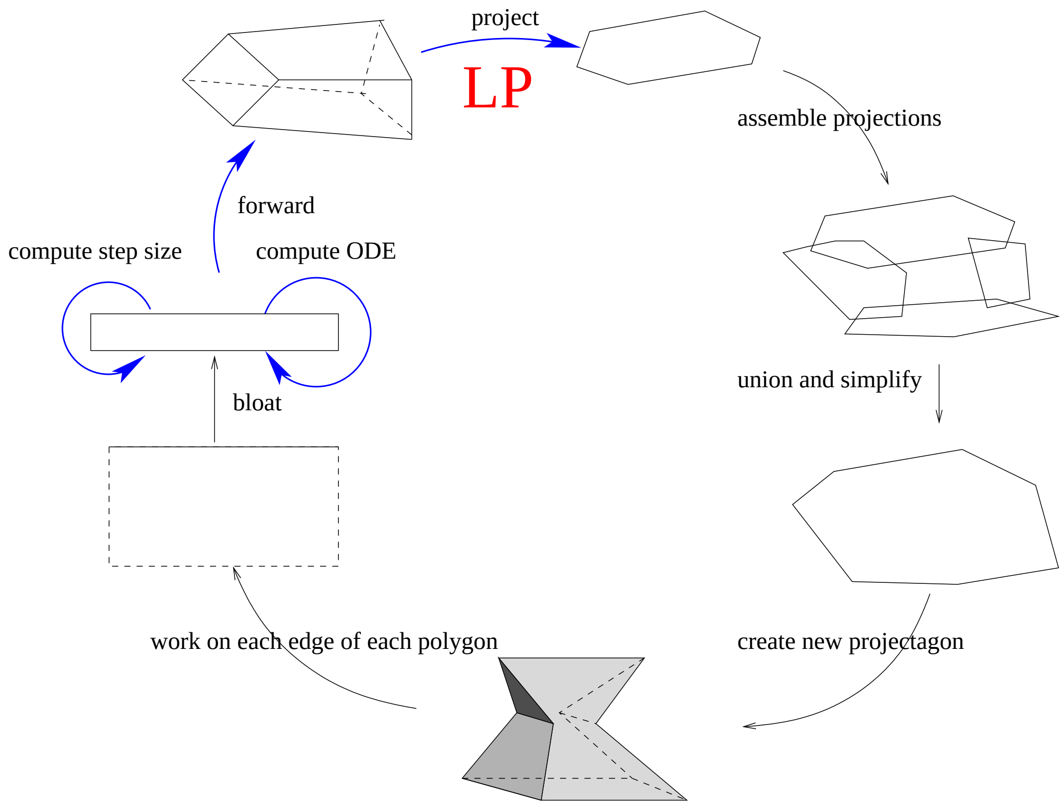


- Construct the brockett annulus for z, ignoring the inverter
- Perform a separate reachability analysis for the output inverter
- Arbitrary ripple counter

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 projectagon $Ax \leq b$ down onto $(\hat{x}, \hat{y})$ subspace

The basic idea is to solve LPs of the form

$$\max_{v \in \mathbb{R}^n} (\hat{x}\cos\theta + \hat{y}\sin\theta) \cdot v \text{ s.t. } Av \leq b$$

for all θ that are the normal of polygon edges.

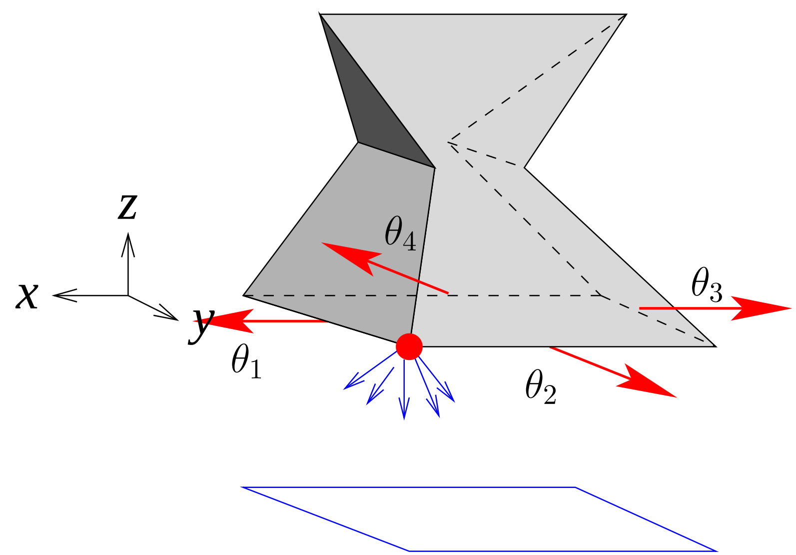
Given θ_c of current edge, the optimal basis $\mathcal{B}$ is computed by solving the LP.

COHO solves the dual of the LP

$$\min_{u \in \mathbb{R}^{+m}} b \cdot u \text{ s.t. } A^T u = \hat{x}\cos\theta + \hat{y}\sin\theta$$

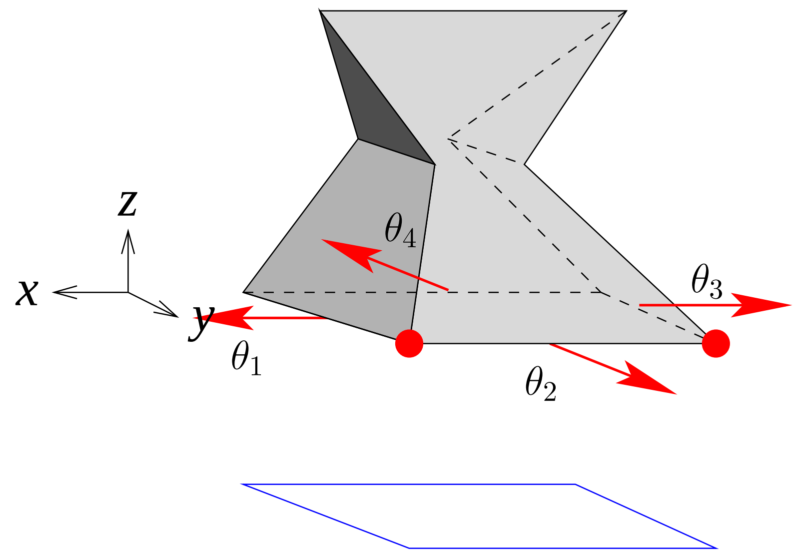
as $u = A_{\mathcal{B}}^{-T}(\hat{x}\cos\theta + \hat{y}\sin\theta)$.

θ_n for next edge is the critical value at which u acquires a negative element.



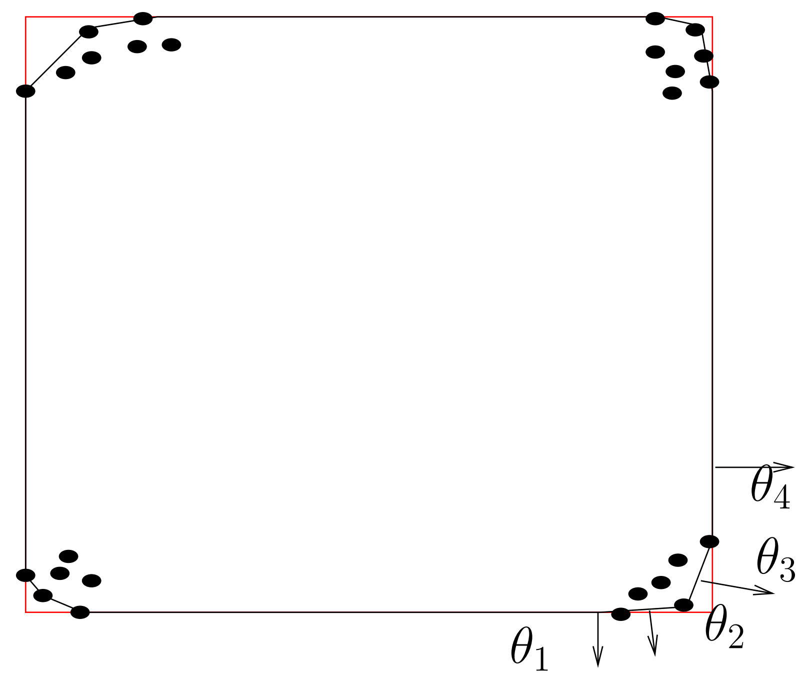
Faster Projection Algorithm

- O(n) time linear program solver
  - A single pivot distance between adjacent optimal basis
  - Increase angle of optimization direction until current vertex is infeasible
  - Remove infeasible column and find new column to bring in
  - It works for about 80% of the time
- Approximated projection algorithm



Faster Projection Algorithm

- O(n) time 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 two steps by enforcing a lower bound on the change of θ
    - The number of LPs to solve is decreased by 50%

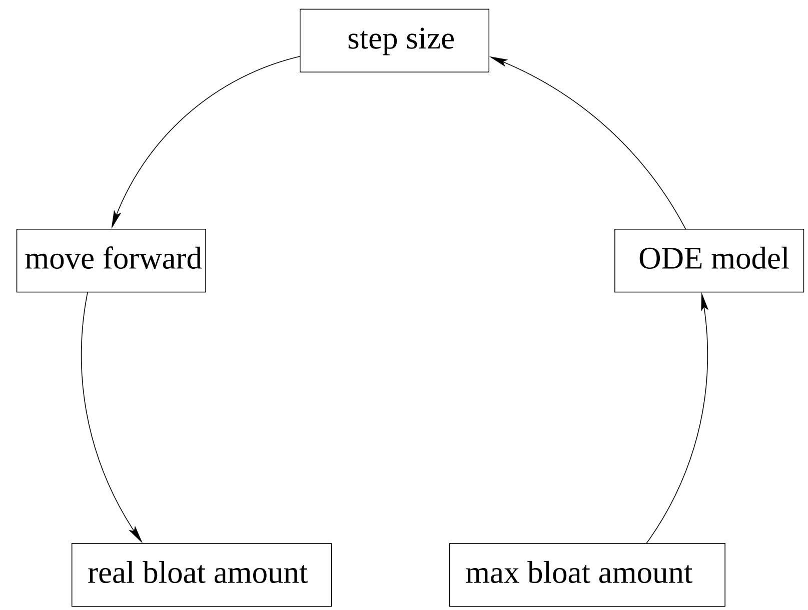


Faster Projection Algorithm

- O(n) time linear program solver
- Approximated projection algorithm
- 2.4x speed-up

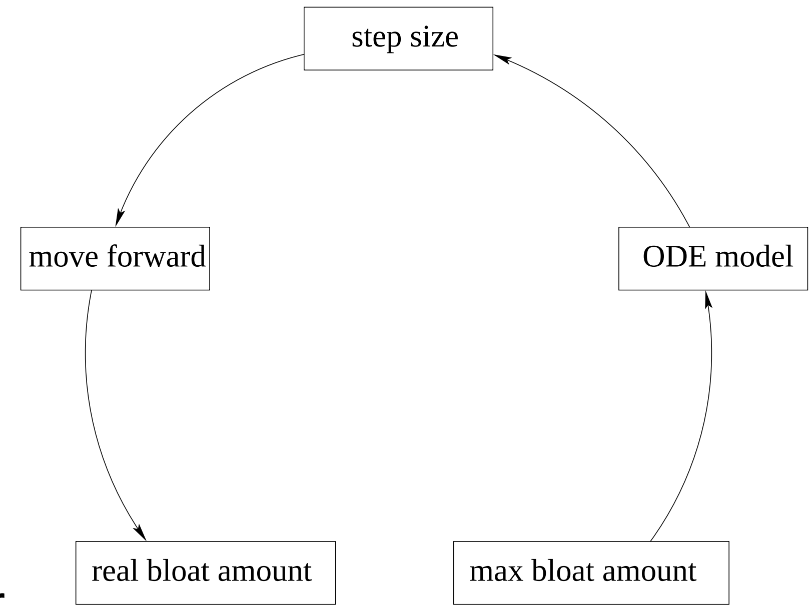
Improved Bloating and Time-Step

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



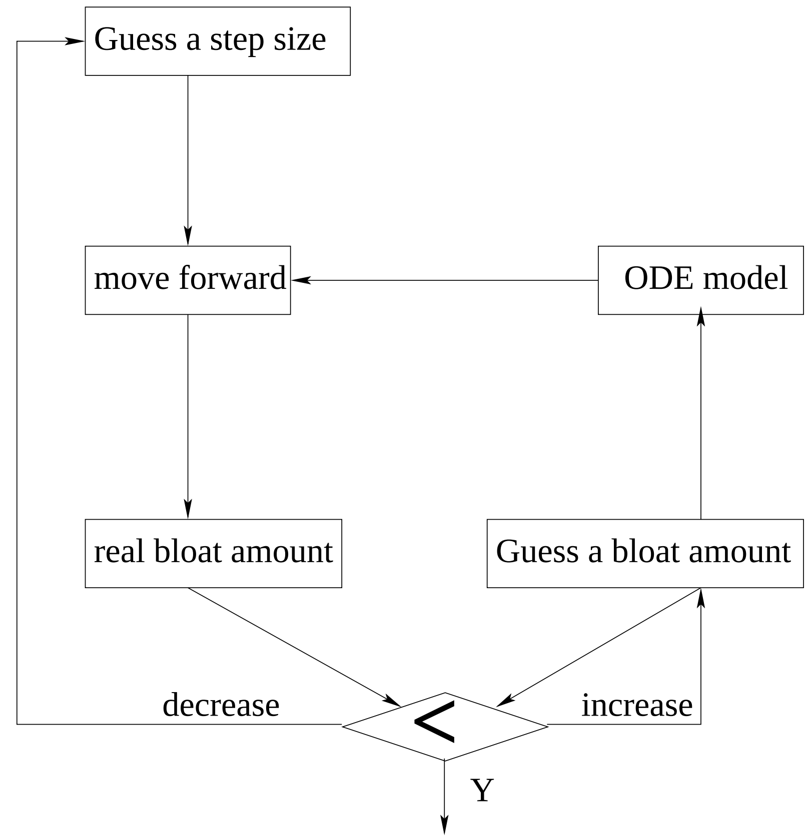
Improved Bloating and Time-Step

- 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.