The Timed and Hybrid Group at Verimag

Oded Maler

CNRS - VERIMAG
Grenoble, France

December 2007

Summary

- General presentation of Verimag
- The Group
- Hybrid Systems
- Timed Systems
- SAT

Verimag

- Academic research laboratory situated in Grenoble, France
- Affiliated with CNRS (national research institute) UJF (scientific university) and INPG (engineering school). French administration is a very technical topic that cannot be elaborated in a short presentation
- Around 30 researchers, 20 engineers and post-docs, 30 PhD students
- Involvement in many European, national and industrial projects
- Growing collaboration with ST microelectronics

Verimag: Claims to Fame

- Pioneering work on Model-Checking, theory and tools (Sifakis 81)
- Development of the data-flow programming language Lustre (Caspi and Halbwachs), the basis of the SCADE environment for programming safety-critical applications (flight control of Airbus, subways, nuclear plants, automobiles)
- More recent:
- Pioneering work on verification of timed and hybrid (discrete-continuous) systems
- Other topics (not covered here): verification of infinite-state systems (abstraction, decidability) and of security protocols
- Work on SystemC, on code generation from components, performance aware compilation, systems biology, etc.

Timed and Hybrid Systems Group

- Around 10 persons, headed by Oded Maler
- Traditionally closer to the R than to the D in the R&D spectrum/pipeline, investigation of new domains less mature for industrial transfer
- But this is slowly changing
- The focus: adapting verification-like techniques to systems defined at a more refined level of abstraction than traditionally (discrete finite-state systems)
- In particular: models that involve dense time and models that involve real-valued variables

Continuous and Hybrid Systems: Motivation

- System with real-valued state variables and dynamics defined by differential equations

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

  or their discrete-time counterparts

$$x_{t+1} = f(x_t, u_t)$$

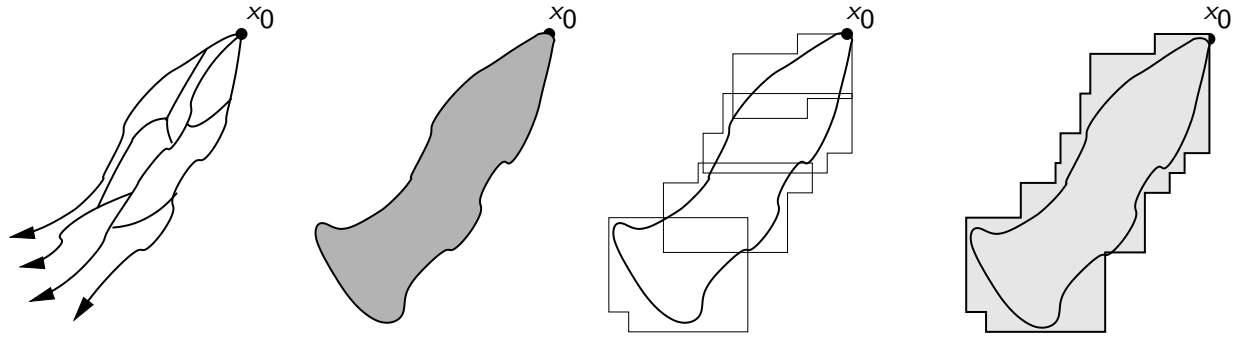
- Motivation: models of the external physical environment that a computer has to control
- Motivation: models of analog circuits (voltage, current) and of digital circuits at the physical level
- Goal: answering questions about the behavior of a system in the presence of *any* input signal u

Continuous and Hybrid Systems: Techniques

- Reachability-based techniques: exporting the ideas of symbolic discrete (digital) verification by computing/approximating the set of reachable states
- Trajectory-based techniques: proving system correctness by finitely-many simulations, finding input stimuli with good coverage
- Monitoring: expressing desired properties of analog signals via an extended real-time temporal logic (PSL-AMS) and generating monitors that detect violation of these properties by simulation traces (lightweight verification)

Reachability Computation (T. Dang, G. Frehse, C. LeGuernic, A. Girard)

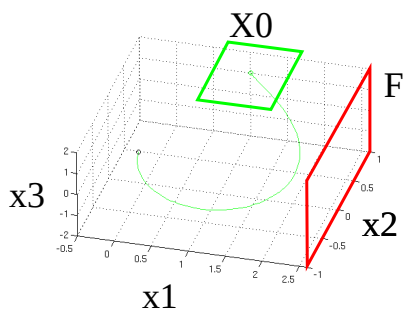
- Compute all reachable states by a "set-based" numerical integration



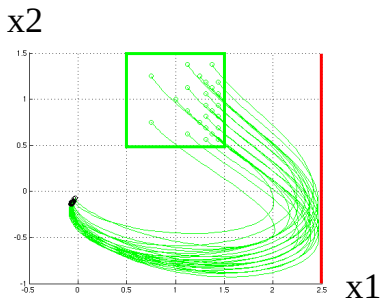
- Compute states reachable at time interval $[kt, (k+1)t]$ from states reachable at $[(k-1)t, kt]$ (under *all* inputs u)
- Main issues: how to represent sets efficiently (polytopes, ellipsoids, zonotopes)
- State of the art: for linear systems $\dot{x} = Ax + Bu$ we can treat systems with *hundreds* of state variables
- Current focus: hybrid systems (mode switching) and nonlinear systems

Trajectory-based Methods: Sensitivity (A.Donze)

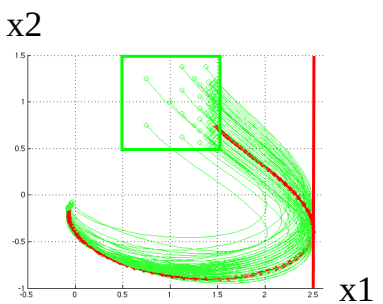
- How to use finitely-many simulations to prove a safety property?
- Sensitivity analysis and sample refinement to prove/disprove intersection with bad states
- Example: 50 dimensional linear systems with 2-dim initial set:



$d = 2.6$
safe
1 trajectory



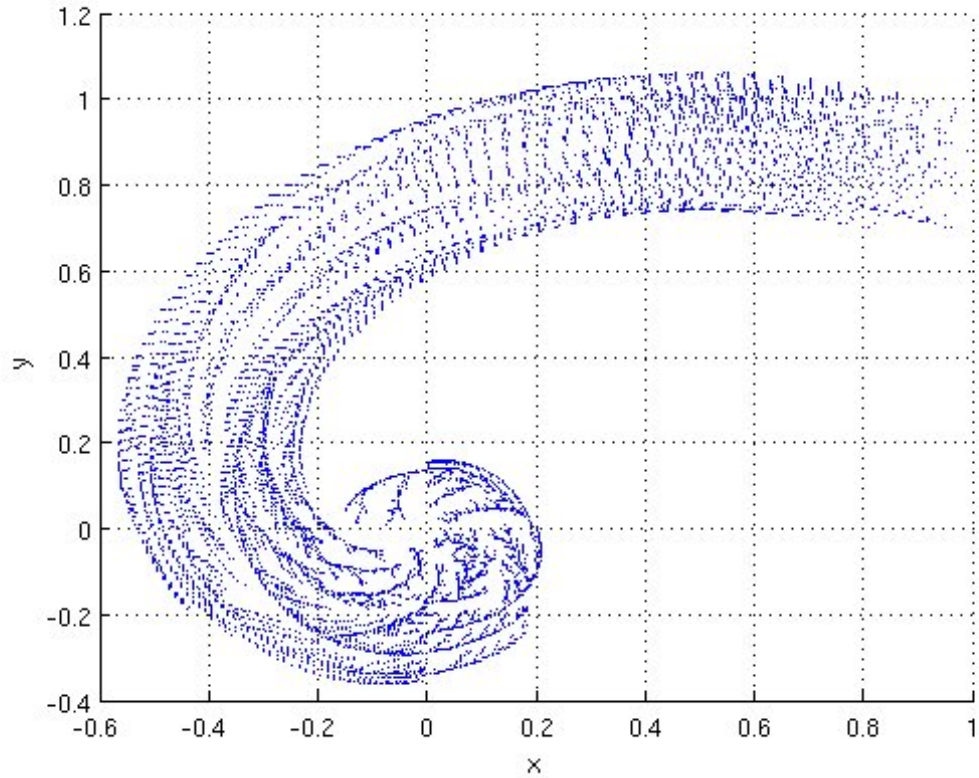
$d = 2.5$
uncertain, $\delta = 0.1$
25 trajectories



$d = 2.5$
unsafe, $\delta = 0.01$
63 trajectories

Trajectory-based Methods: Random Search (T. Dang)

- Use RRT techniques from robotics path planning in order to give a good coverage of the reachable part of the state space
- Connection with SPICE simulator



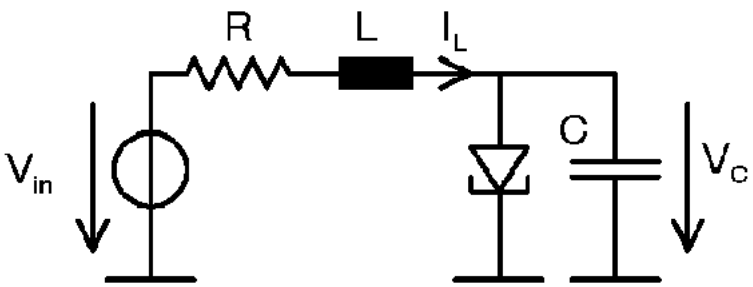
Monitoring (D. Nickovic)

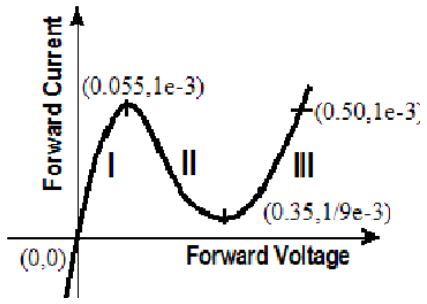
- Whatever we achieve in "formal" verification, simulation is here to stay, especially for the analog parts
- Analog simulations are long and tedious
- We can at least liberate the designer from observing their output
- We developed a temporal logic for defining (sequential) properties of analog (or mixed) signals (PSL/AMS)
- From these specification we derive property testers that can be plugged into simulators and automatically detect violation
- Connection to simulators can be either online (during simulation) or offline
- The AMT tool (separate slides)

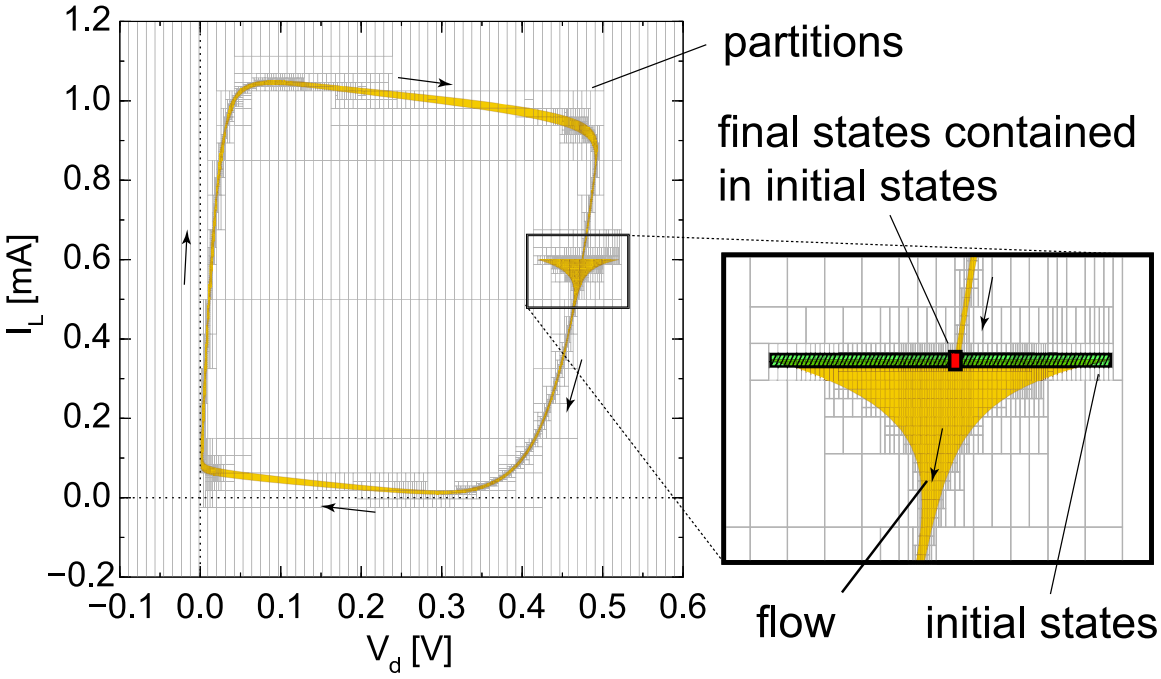
Analog Case Studies

- Verification
  - Tunnel diode oscillator
  - Differential VCO oscillator
  - Biquad filter
  - Sigma-Delta modulator
  - AC-DC rectifier
  - Buck converter
- Monitoring
  - FLASH memory
  - DDR2: alignment between data DQ and data strobe DQS signals

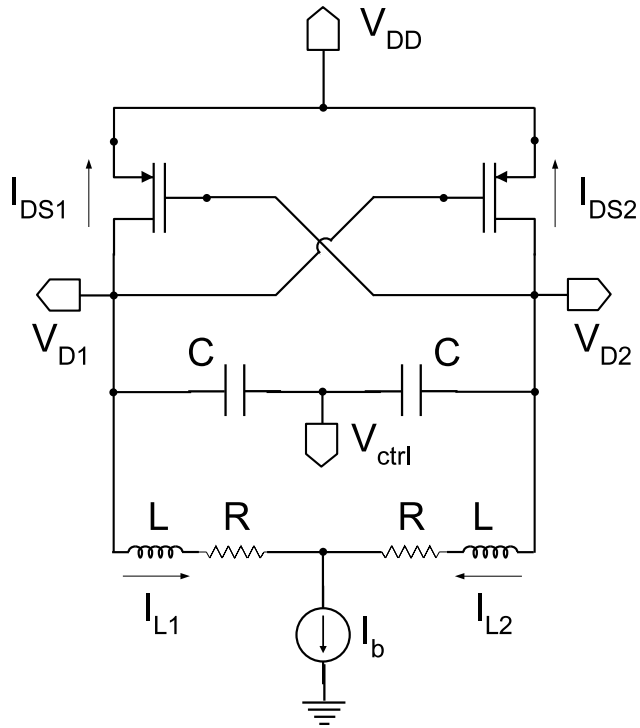
Example: Tunnel Diode Oscillator (G. Frehse)

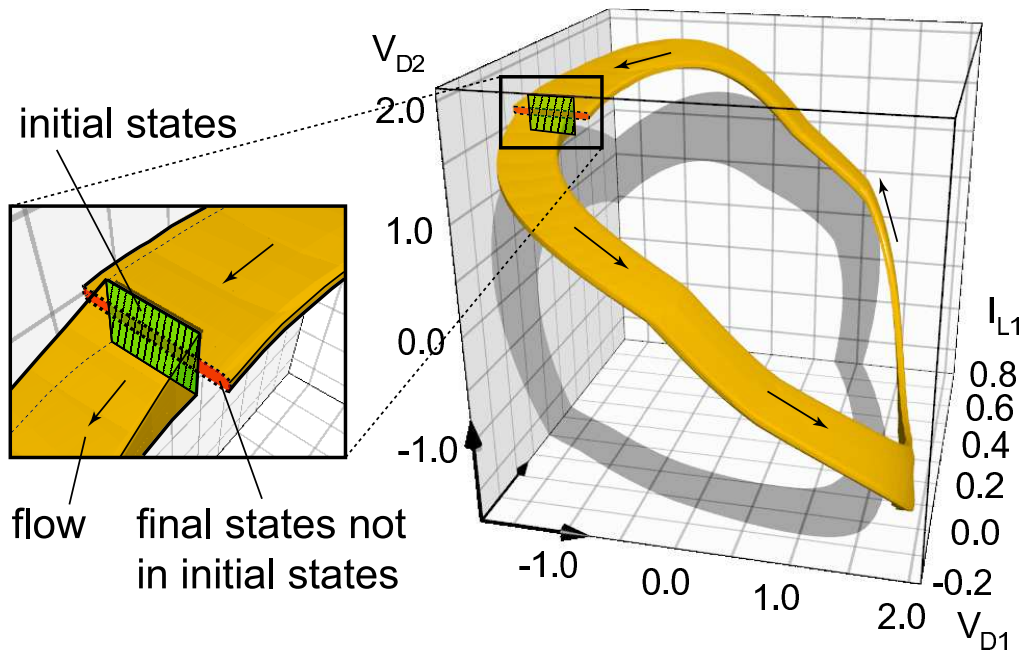






Example: Differential VCO Circuit (G. Frehse)



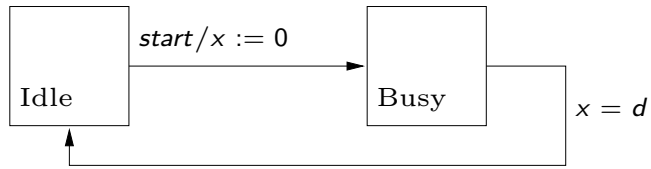


Timed Systems: Introduction

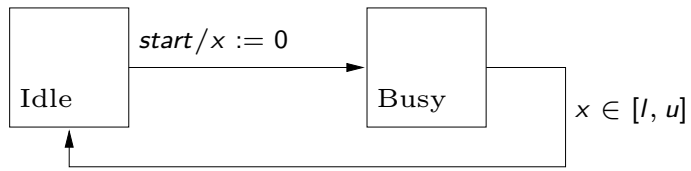
- Modeling system behavior at a level of abstraction which consists of discrete transitions separated by time duration (Boolean signals)
- The computational model is the *timed automaton*, a finite-state machine augmented with real-valued clocks
- The clocks measure the time since certain events and their values enable/disable further events
- Appropriate for modeling any process that takes time:
  - The time it takes a logical gate or a transistor to switch after its input has changed
  - The time is takes a block of code to execute
  - The time it takes a message to propagate in a network
  - Timing assumptions on the environment of the system
  - Response time requirements

The Simplest Example

- A process that takes d time between initiation and termination



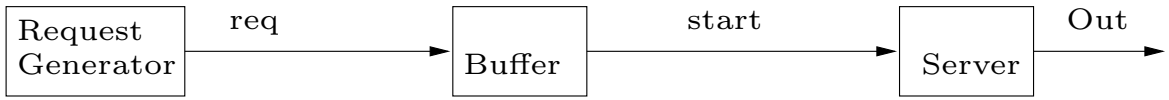
- A process that takes between l and u time between initiation and termination (set-theoretical non determinism)



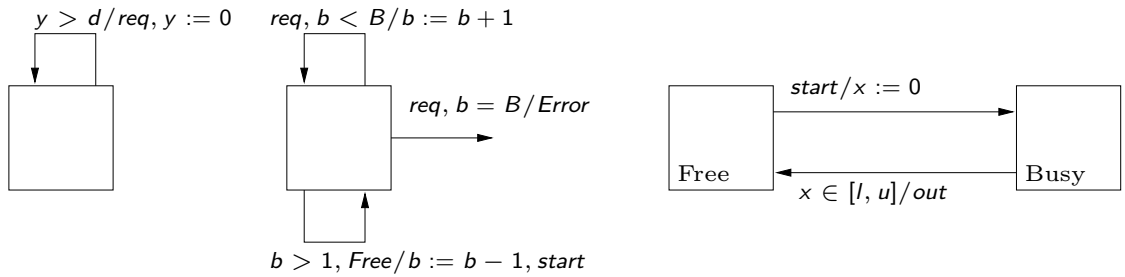
- No a priori commitment to the size of the time step (unlike clock cycles)

Getting more Complex

- Imagine a process that generates requests, puts them on a bounded buffer which sends them to a server.

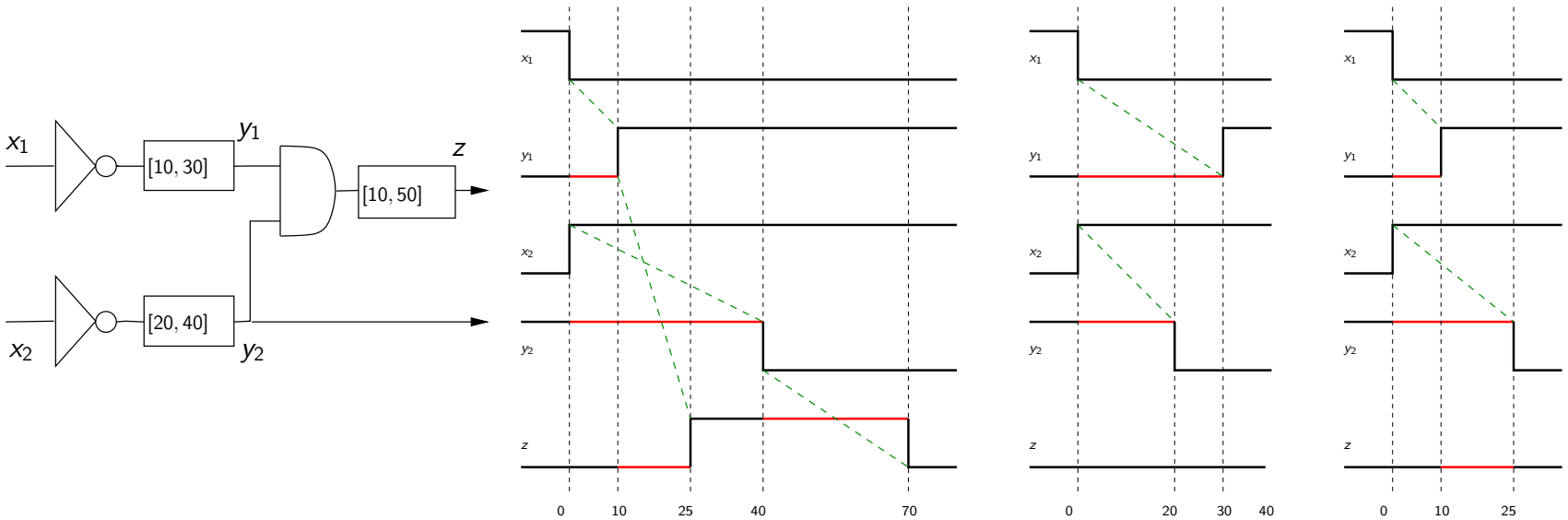


- Assume every two requests are separted by at least d time, that buffer size is B and service time is between l and u



- Composing them together we have a timed automaton which represent all the possible behaviors of the system
- More complex situations with various resource generators, different server types, schedulers, admission controllers, etc.

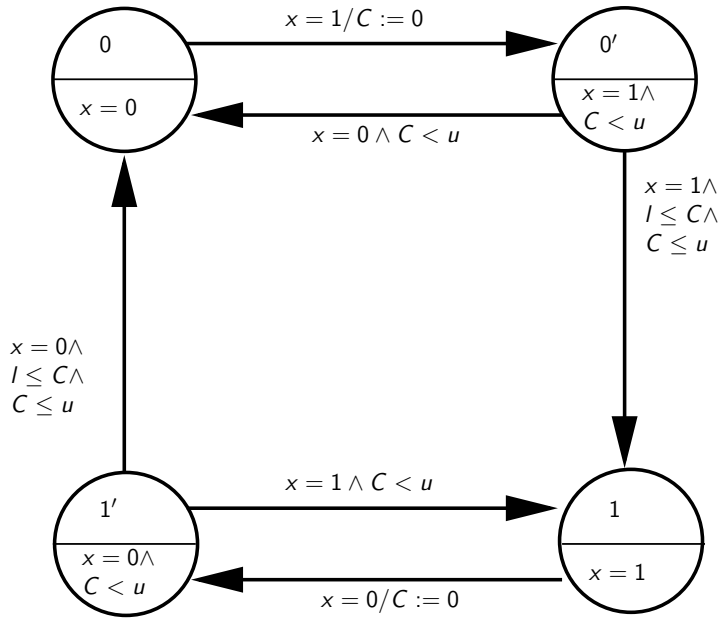
Example: Circuits with Bi-bounded Inertial Delays



- A circuit may have infinitely-many different behaviors depending on the "choice" of the delay of each gate

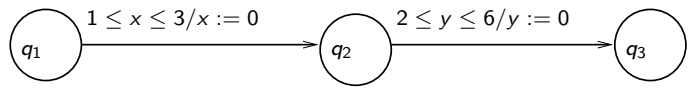
Modeling Delays with Timed Automata

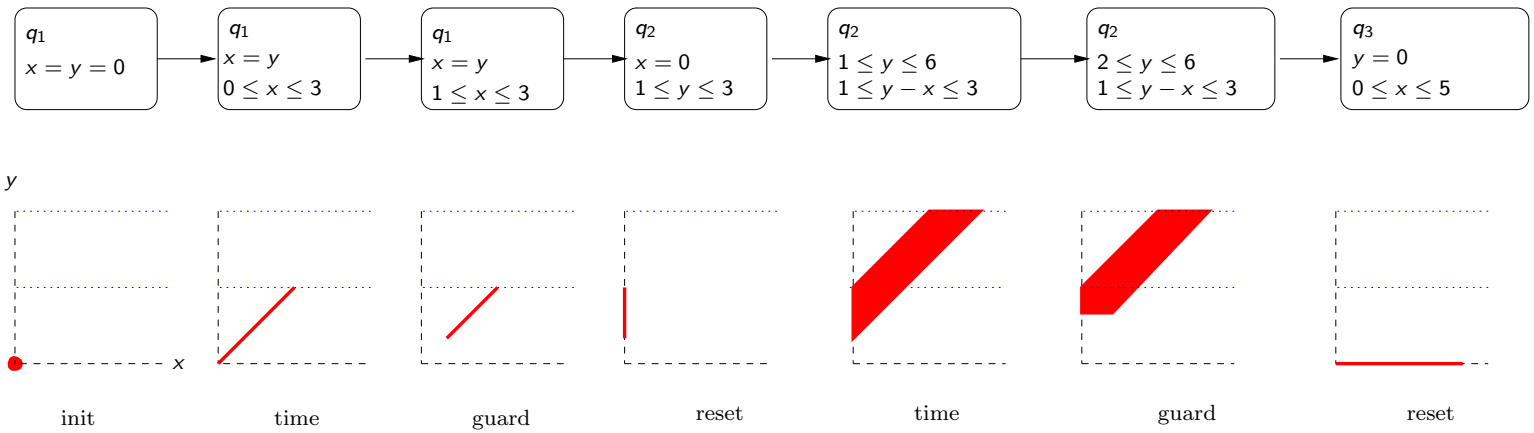
- Each gate is decomposed into an instantaneous Boolean function and a bi-bounded (non-deterministic) delay element
- Each delay element is modeled as a timed automaton with 4 states and 1 clock
- Composing all these automata we obtain a timed automaton with $O(2^n)$ states and n clocks, which captures *all* the behaviors of the circuit



Timed Automata: Analysis

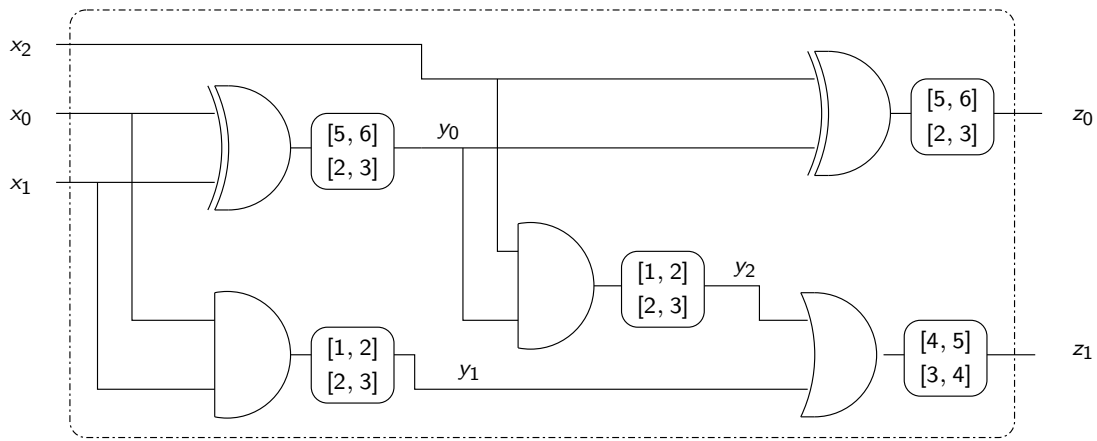
- Forward reachability computation in an extended state space (discrete states and sets of clock valuations)

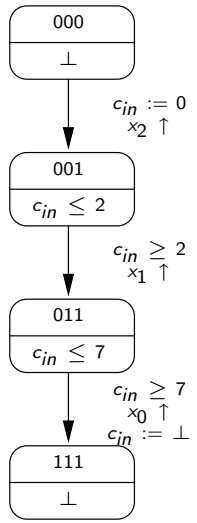




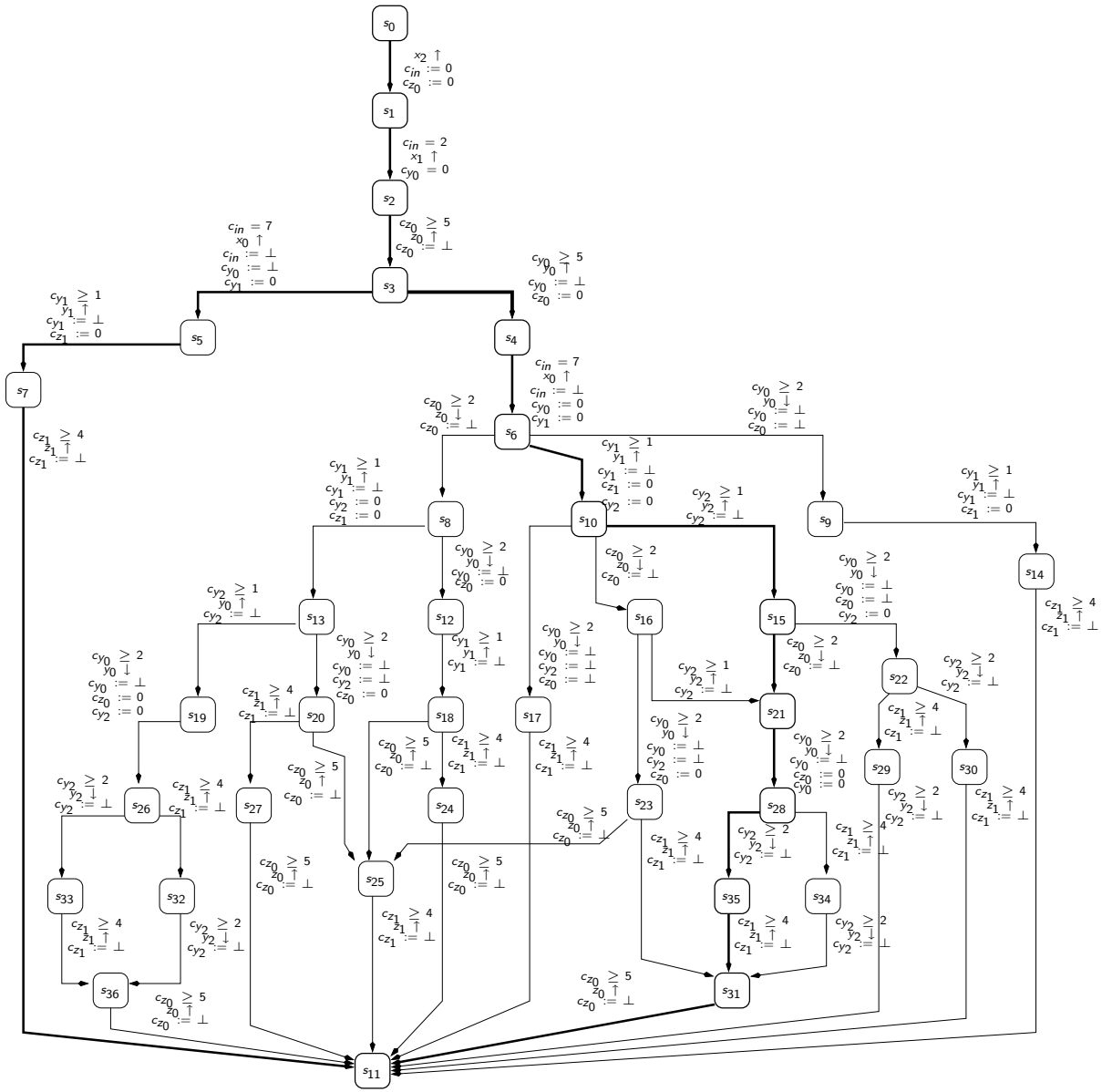
- Decidable but costly. Current tools cannot handle more than $10 - 20$ timed components

Circuit Example: Full Adder and its Input Generator



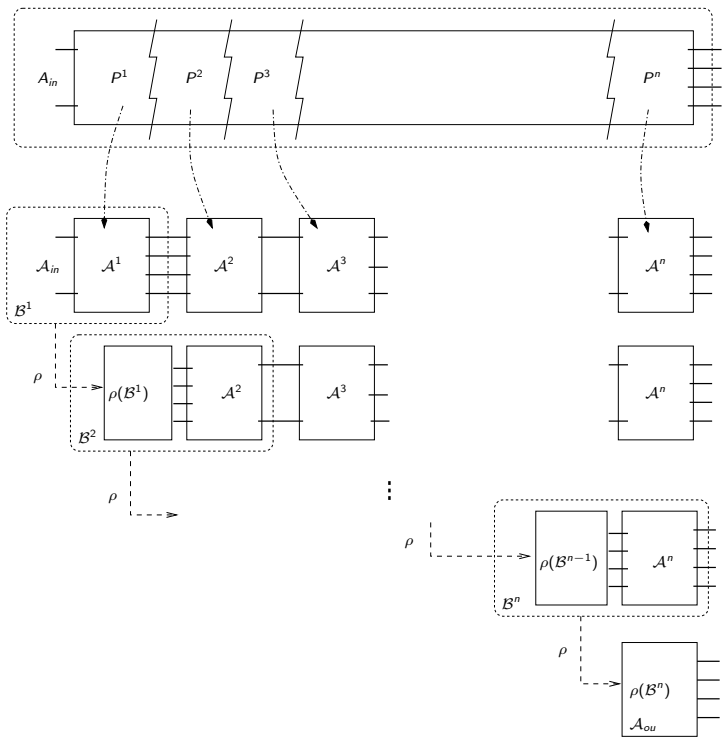


Full Adder: Analysis Results



Analyzing Large Systems

- A divide-and-conquer methodology: cut the circuit into pieces, analyze a piece, create an abstraction of its I/O behavior and plug it into the rest of the circuit

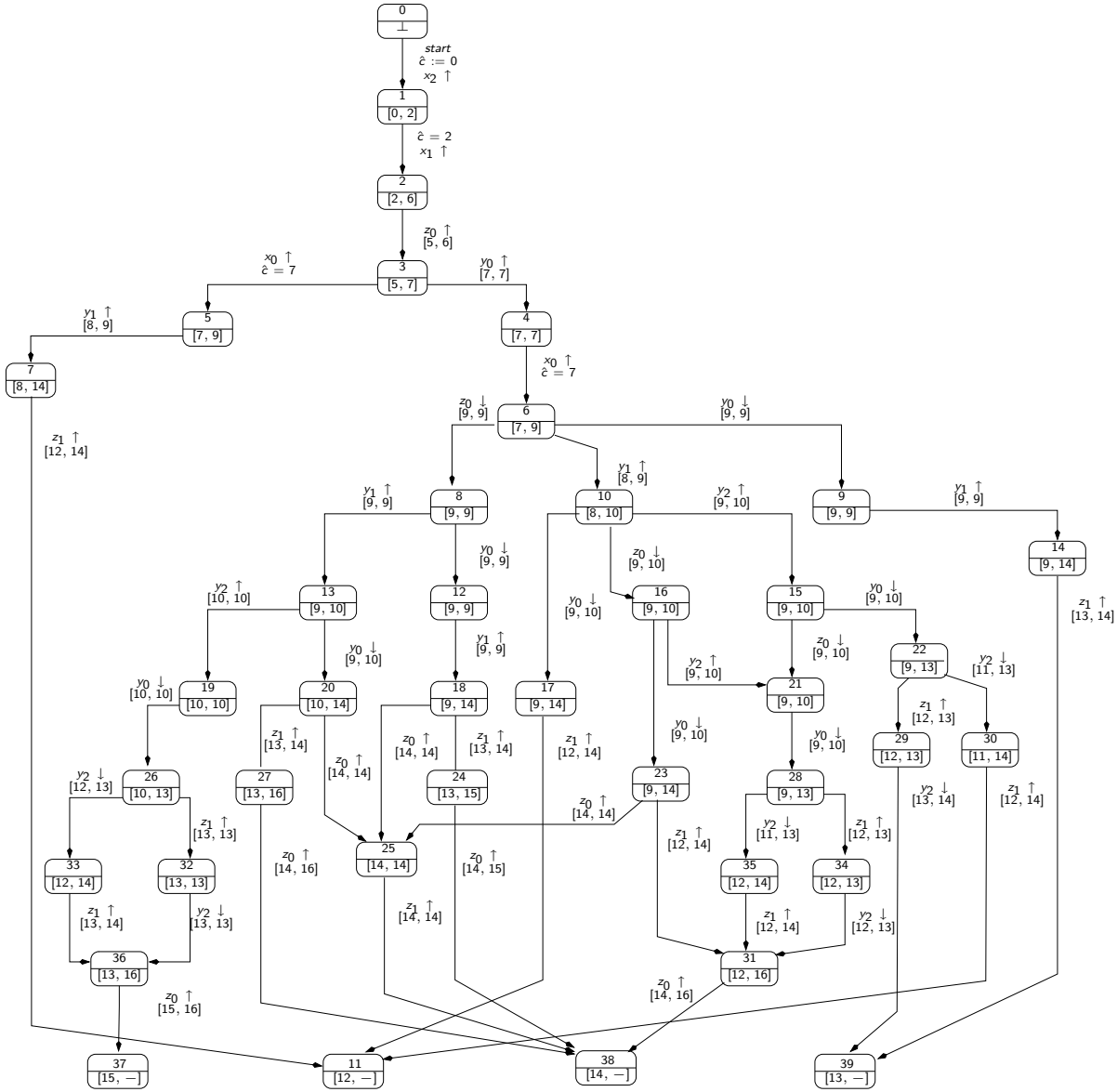


The Abstraction Technique (M. Bozga, R. Ben Salah)

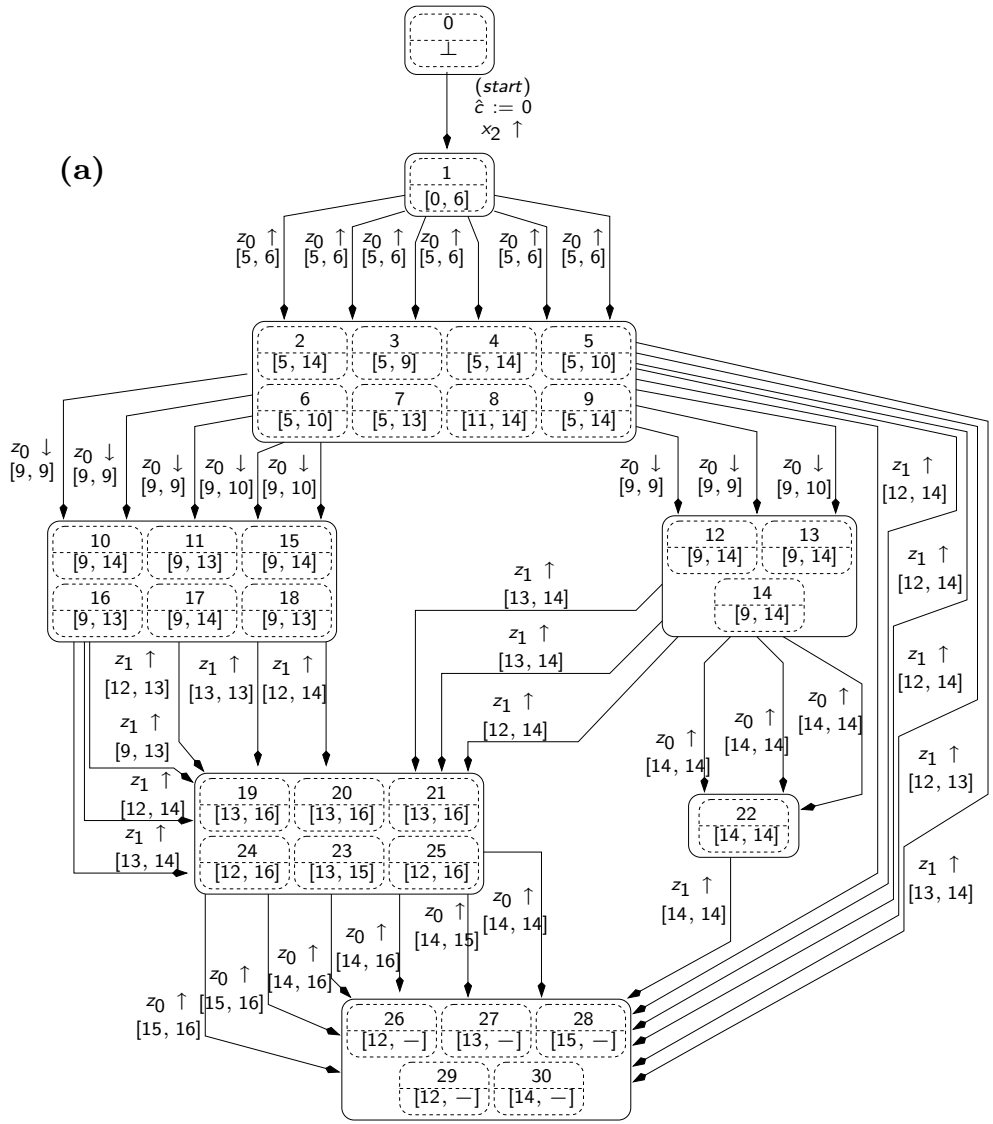
- Add an auxiliary clock that measures absolute time
- Perform reachability analysis
- Project the timing constraints on absolute time; the obtained automaton over-approximates the behavior, some correlations between the timing of certain events may be lost
- Hide internal transitions and minimize the automaton

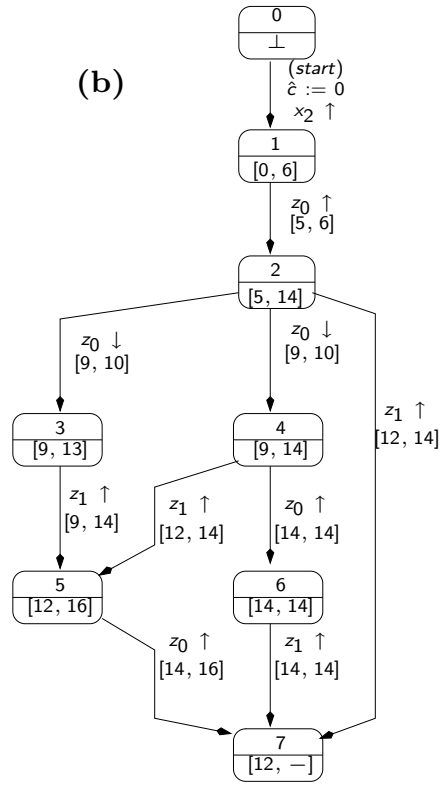
The Abstraction: Example

The automaton after projection



The Abstraction: the Reduced Model





0={0}
1={1}
2={2,3,4,5,6,7,8,9}
3={10,11,15,16,17,18}
4={12,13,14}
5={19,20,21,23,24,25}
6={22}
7={26,27,28,29,30}

Abstraction: State of the Art

- For circuits that have inputs that change only once, we can analyze by divide-and-conquer circuits with around 100 gates (will be improved, but against one input scenario)
- We have developed a very promising technique for circuits/systems with inputs that keep on arriving
- No absolute time to project on: each input events generates its own clock which is killed when the event leaves the system
- The abstract model relates the timing of each output event to the input event that triggered it
- Very promising for generating small size abstraction of components (not necessarily at the circuit level)
- Still work to be done

Other Efforts: SAT Modulo Theories (S. Cotton)

- Motivated by timed automata we developed a powerful SMT solver for the theory of difference constraints ($x - y < c$)
- Can be used, for example, in combination with static timing analysis to eliminate false paths
- As a side effect we have developed open source SAT and QBF solvers in Java
- Currently trying to use them to find invariants