Toward Formal Verification of AMS Circuits

Oded Maler, Thao Dang, Antoine Girard, Goran Frehse, Alexandre Donze, Tarik Nahhal, Dejan Nickovic, Colas Le Guernic, Rajarshi Ray, Romain Testylier, Noa Shalev ...

CNRS - VERIMAG
Grenoble, France

Munich 23/11/2011

Disclaimer

- I am **not** a hard core circuit (or even EDA) person
- My background is in the theory and practice of formal verification, traditionally restricted to digital systems
- Our group has been working for 20 years on extending verification to **hybrid systems**:
- Systems that mix **discrete** and **continuous** dynamics: finite-state machines and differential equations
- We developed complementary techniques and tools for validating such systems: **test generation**, **assertion language**, **parameter-space exploration** and **formal verification**
- We realized that analog circuits is perhaps the best application domain for these techniques
- This talk is a survey of some of the problems, some of our solutions and case-studies

Academic Landscape (the Push Side)

- Major verification conferences (CAV, FMCAD, TACAS) are dominated by the discrete view (digital hardware and software)
- The hybrid systems conference (HSCC) is mainly driven by control applications
- There are some slots in circuit and EDA conferences
- We started a series of workshops
- **FAC: Formal Verification of Analog Circuits**
- Edinburgh 2005, Princeton 2008, Grenoble 2009

Academic Landscape (the Push Side)

- In Salt Lake City 2011 the name changed to **Frontiers in Analog Circuits** to cover additional concerns other than verification
- Steering committe: M. Greenstreet (UBC), L. Hedrich (Frankfurt), M. Horowitz (Stanford and Rambus), O. Maler (Verimag), C. Myers (Utah) and R. Rutenbar (UIUC)
- Additional 2011 speakers: C. Grimm (TU Wien), R. Hum (Mentor), M. Marcu (Agilent) and G. Taylor (Intel)
- Next workshop will be held in February 2013, San Francisco (with ISSCC)

Industrial Needs (the Pull Side)

- The **verification bottleneck**: our ability to understand complex systems grows slower than our ability to assemble them
- I think this holds also for administrative constructions
- The particularity of **analog** circuits:
- Boundary between inherently different levels of abstraction:
- Moving between RTL and gate level you change scale but the nature of the (dynamical) system is the same
- Moving between Boolean gates to transistors you change the nature of the dynamics

Industrial Needs (the Pull Side)

- The particularity of **analog** circuits:
- Cultural gaps: digital designers, EDA providers and even theoreticians share common concepts: Boolean functions, sequential machines
- Analog designers come from other cultures, e.g. signal processing, physical sciences
- Analog design is still considered as a handcraft artistic activity compared to the formalization and bureaucratization in the digital design process
- Analog devices are small but may cause a lot of problems: the mythical 20% - 80% ratio

The Scope of AMS Thinking

- Underlying models are, this way or another, continuous (and hybrid) dynamical systems with state variables indicating mostly voltages
- Reasoning at this level is needed in:
- Purely analog functions that interact with the **physical** world (RF, MEMS, etc.)
- Interface technology between digital components: memory (FLASH, DRAM, etc.) communication (ETHERNET)
- D/A and A/D converters
- PLLs for oscillators/clocks in digital circuits
- And finally: voltage-level analysis of digital circuits, for example, for power and noise analysis

The Major Verification Question

- We build an analog device, say a PLL
- This device will be embedded in **different environments**, physically and logically:
  - It can be realized in different technologies
  - It can be realized in different fabs
  - It can be subject to in-die variations
  - It can be embedded into different SoCs, each providing it with a specific set of **stimuli** and requiring a specific set of **constraints** on the **response**
- We would like to know how **robust** the device is to all these variations
- To characterize the range of environments in which it functions **correctly**

Functioning Correctly

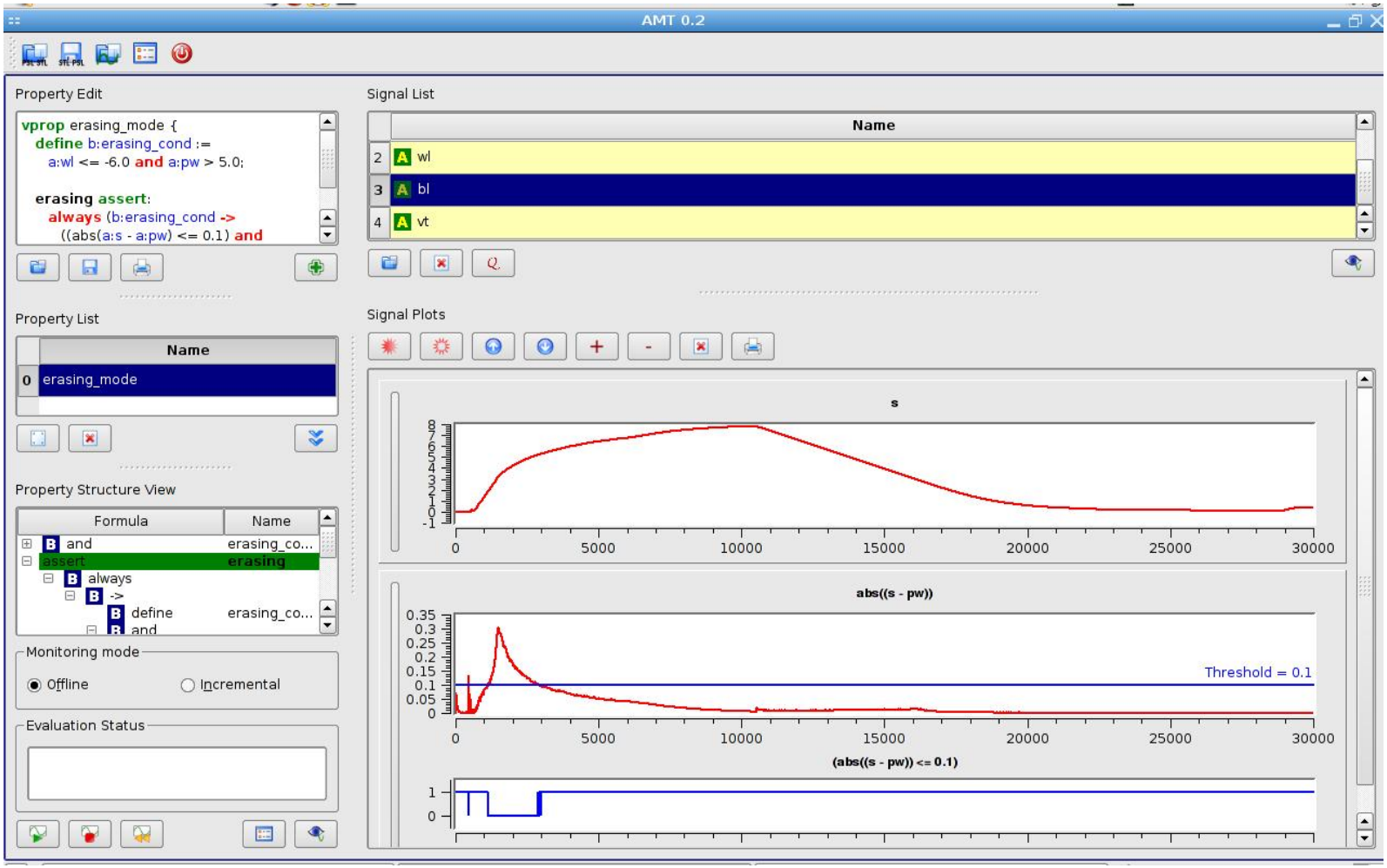
- What relations over time should hold between input and output signals?
- In the EU project PROSYD (ST, IBM, Infineon) we developed an AMS extension of PSL
- It is called **STL** (**signal temporal logic**) and can express such properties/assertions/requirments very elegantly[1]
- **Whenever the voltage of** x **is above** c_x **then within** t_1 **to** t_2 **milliseconds the voltage of** y **will drop below** c_y
- A natural extension for time-domain "sequential" properties used in digital toward dense time and real-valued variables
- Ideal for specifying interfaces between digital to analog
- Extensions to frequency domain properties are under way

[1] Maybe too elegantly for engineers...

The AMT Tool

- Gets as input STL specifications and automatically generates a **monitoring program** ("dynamic" verification)
- It can then read simulation traces, **detect violation** of properties and explain them
- Two working modes:
  - **Offline**: reading traces from a file
  - **Online**: getting the traces from a concurrently working simulator. Can abort (expensive) simulation upon property violation
- Has been applied to circuit case studies: FLASH Memory writing/erasing (ST), DDR interface (Rambus)
- Was discussed within the SVA-AMS working group
- Seems that practitioners still prefer to hack their assertions in the language of the simulator...

The AMT Tool


AMT 0.2
Property Edit
vprop erasing_mode {
define b:erasing_cond :=
a:wl <= -6.0 and a:pw > 5.0;
erasing assert:
always (b:erasing_cond ->
((abs(a:s - a:pw) <= 0.1) and
Signal List
Name
2 A wl
3 A bl
4 A vt
Property List
Name
0 erasing_mode
Property Structure View
Formula
Name
and
erasing_co...
assert
erasing
always
->
define
erasing_co...
and
Signal Plots
s
abs((s - pw))
Threshold = 0.1
(abs((s - pw)) <= 0.1)
Monitoring mode
Offline
Incremental
Evaluation Status

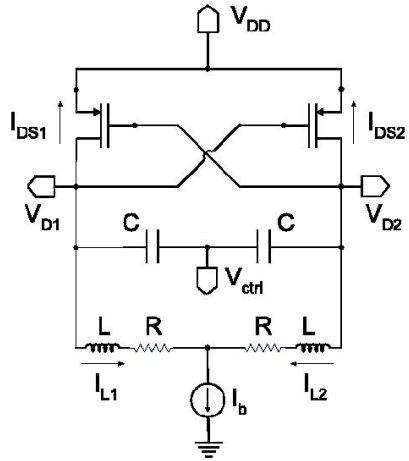
Coverage

- How do we **cover**, using simulation all the possible variations in the external environment of the circuit:
  - Different transistor parameters at the IP level
  - Different initial conditions (that can get an oscillator stuck)
  - Different input signals from other subsystems at both IP and behavioral level
  - Other external variations such as temperature
- Remark: although from an abstract perspective these are all **inputs**, their nature and importance may be quite different
- What is the most efficient way to spend a given simulation time budget?

Parameter-Space Exploration

- Models may depend on **parameters**
- Some parameters are under our control and some are not
- Fixing a **nominal value** for a parameter we can run a simulation but what do we learn about other values?
- We developed an intelligent simulation-based procedure to explore the parameter space
- It can, in principle, prove certain properties based on a **finite** number of simulations
- It can trace (an approximation of) the **boundary** between parameter values that yield some desired behavior and those that do not

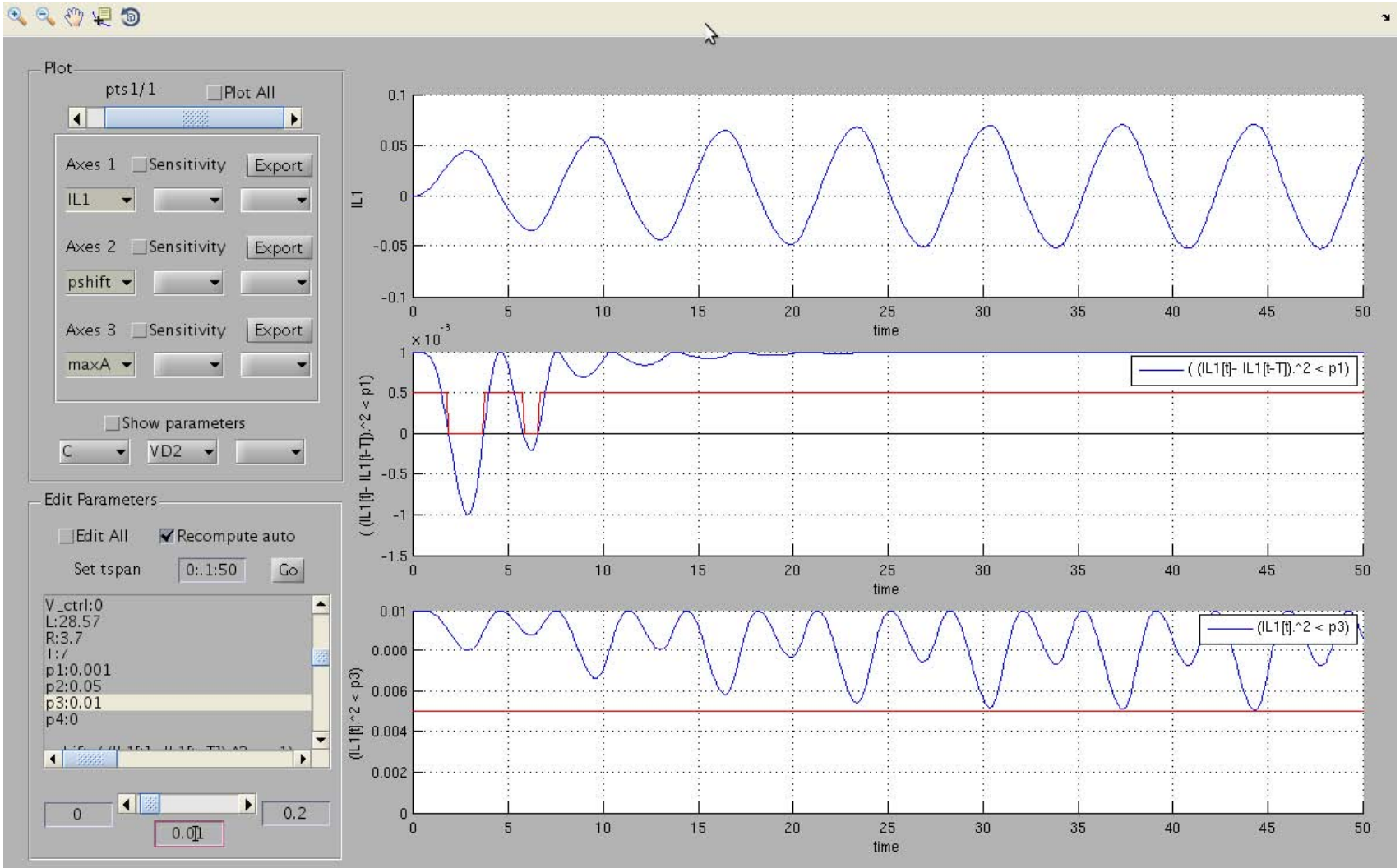
Example: a voltage-controlled oscillator (VCO)



- A nonlinear circuit, 3 state variables and $\sim$ 10 parameters
- Which range of parameters produces good oscillations?
- First we formalize good oscillations in STL:
- Alternating above and below a minimum amplitude:
  `(ev_[0,T] (IL1[t]>Amin)) and (ev_[0,T] (IL1[t]<-Amin))`
- Holding strict periodicity:
  `alw_[0,4*T] ( (IL1[t] - IL1[t-T])^2 ) < epsi)`
- ...

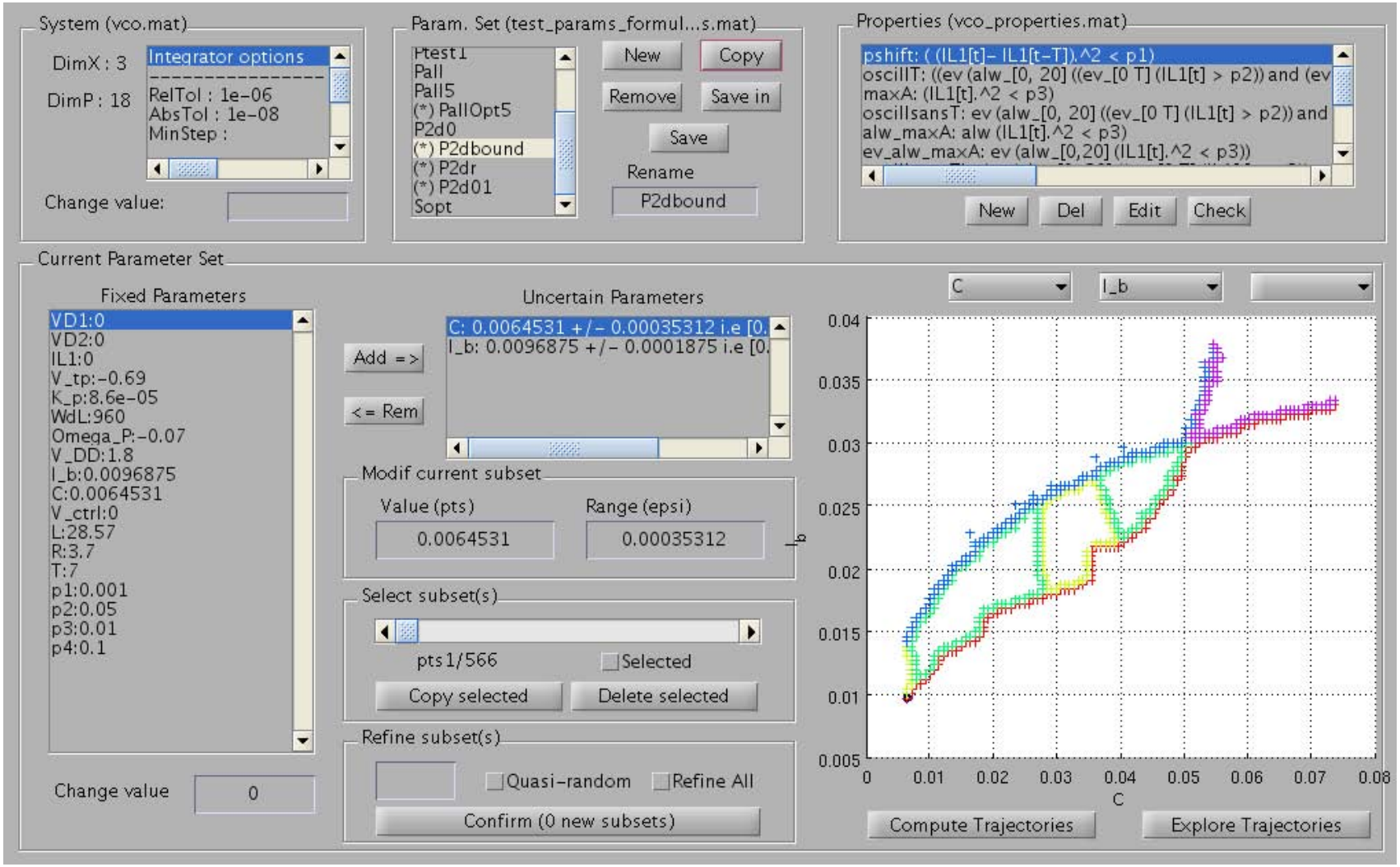
VCO and the BREACH Tool

- For each choice of parameter value we simulate and detect satisfaction/violation of the property



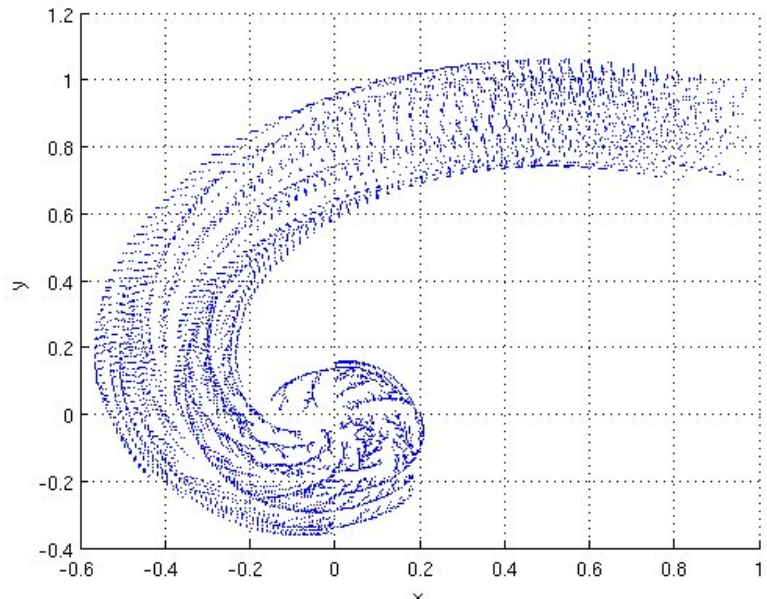
VCO and the BREACH Tool

- At the end we trace the boundaries between satisfaction and violation for each property



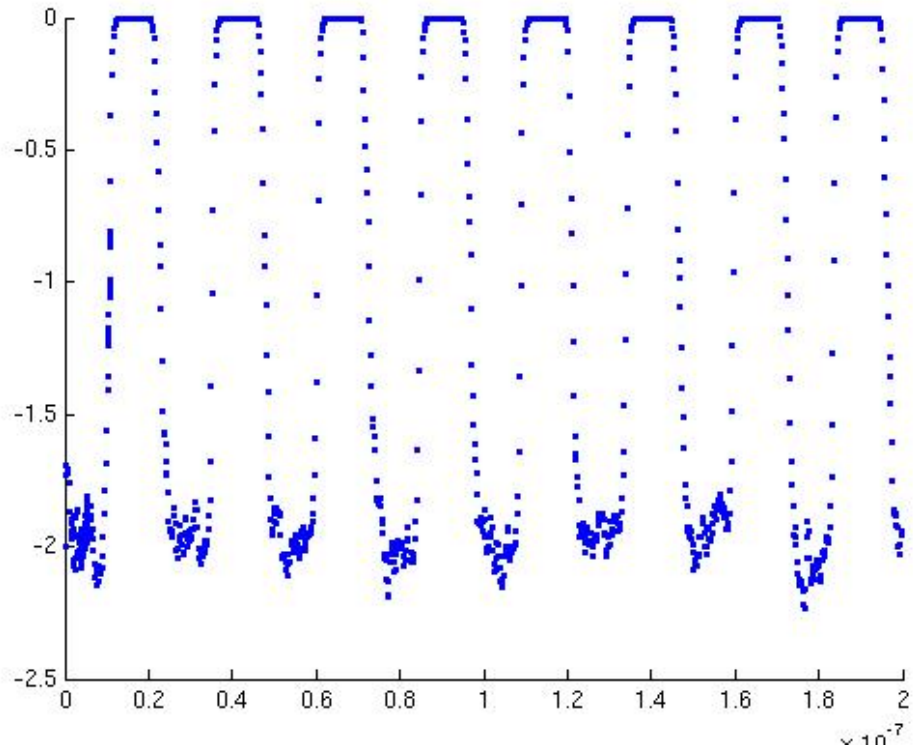
High-Coverage Test Generation

- How to generate **stimuli** that induce good **coverage** of the possible system behaviors?
- Coverage here is more "semantic": covering the **reachable state space** of the system, rather than covering the syntax of circuit description
- The principle: treat stimuli and their induced behaviors as trees which are quasi-randomly generated with statistical coverage as a bias
- Inspired from ideas in robotics motion planning (RRT)



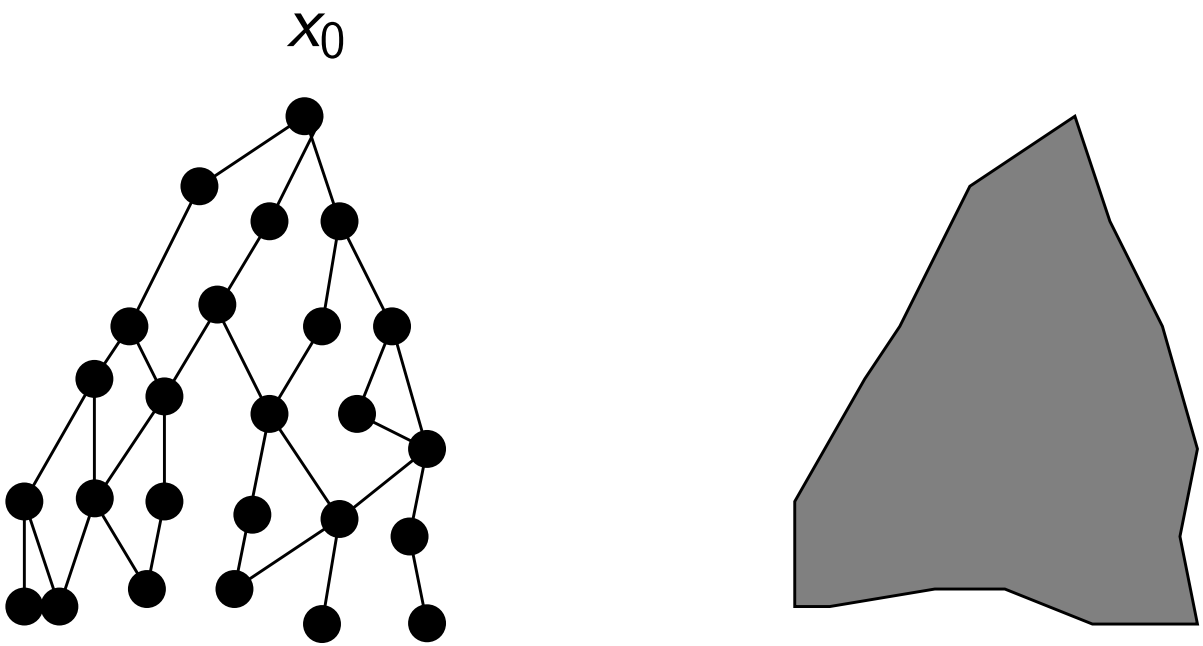
Test Generation: the HTG Tool

- Developed in the French VAL-AMS project with the SICONOS team at INRIA
- Takes SPICE netlists or hybrid automata as input
- Generates inputs in a coverage-guided way
- Applied to many circuits: Sigma-Delta A/D converter, VCO (55 continuous variables), etc.
- Example: a ring oscillator, the input is the source voltage



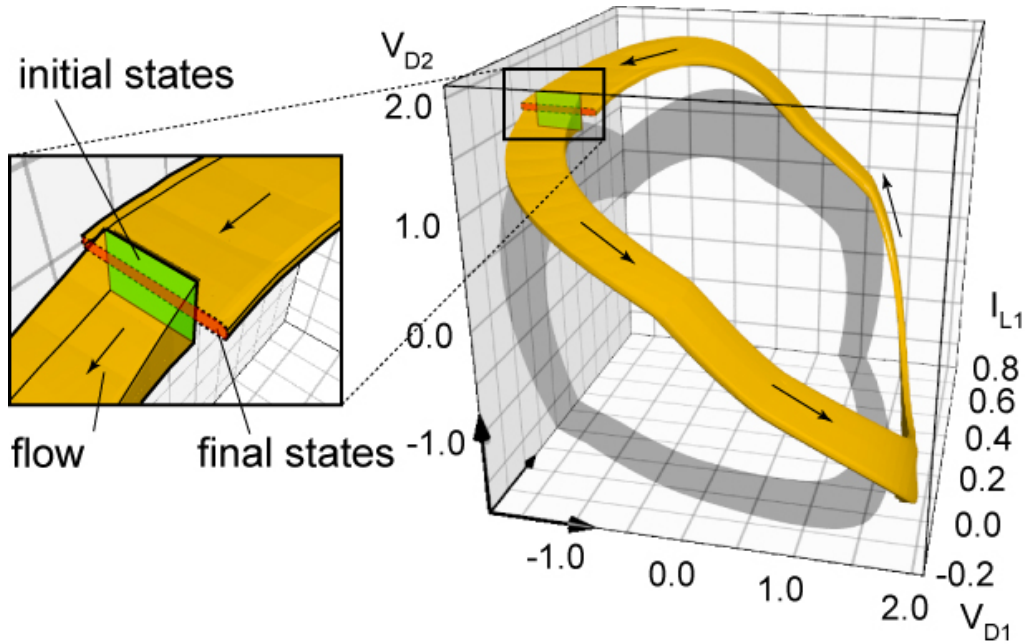
Formal Verification

- This is the most challenging (and somewhat romantic) goal: replace simulation by verification
- This means compute **"tubes"** or **"pipes"** of trajectories in the state space
- A **set-based** simulation that covers **all** variations in parameters, initial states and dynamic inputs
- Breadth-first rather than depth-first exploration



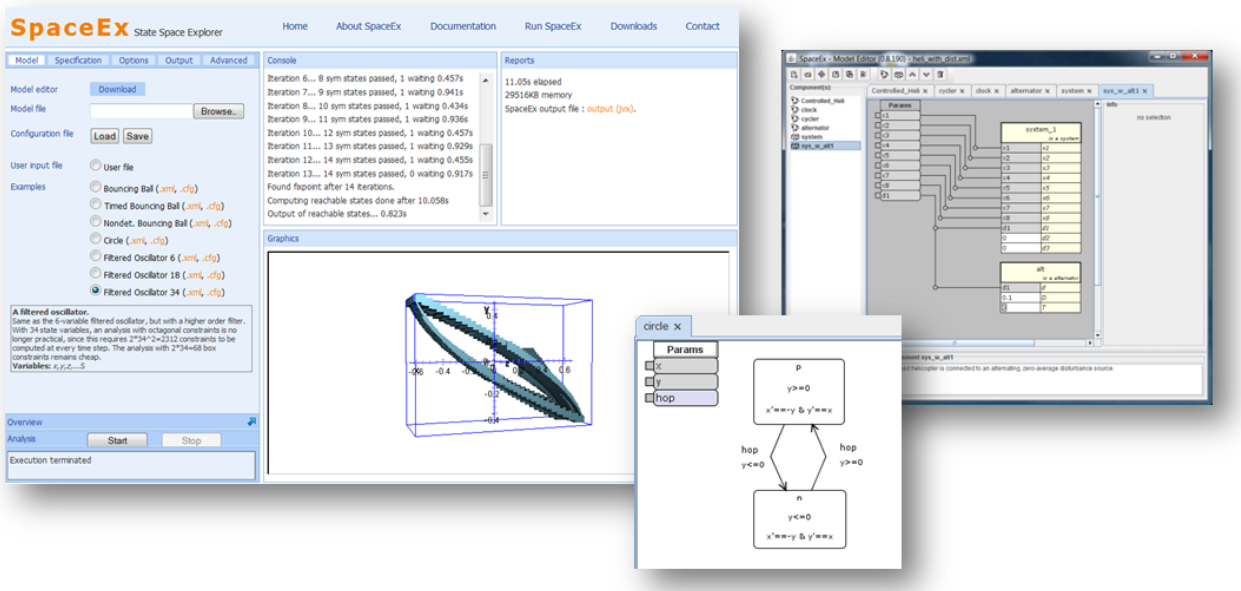
Computing Reachable States

- It is more difficult than simulation because we need to represent and store **sets** in $\mathbb{R}^n$ rather than **points**
- Uses graph algorithms, numerical analysis and **computational geometry** in **high dimension**
- This limits the size of systems that can be treated - small tricky systems at the behavioral level



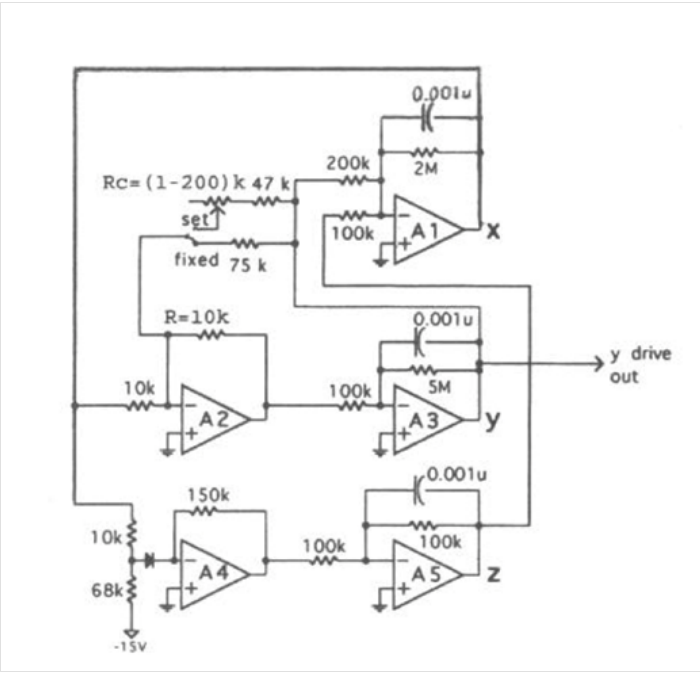
Computing Reachable States: State of the Art

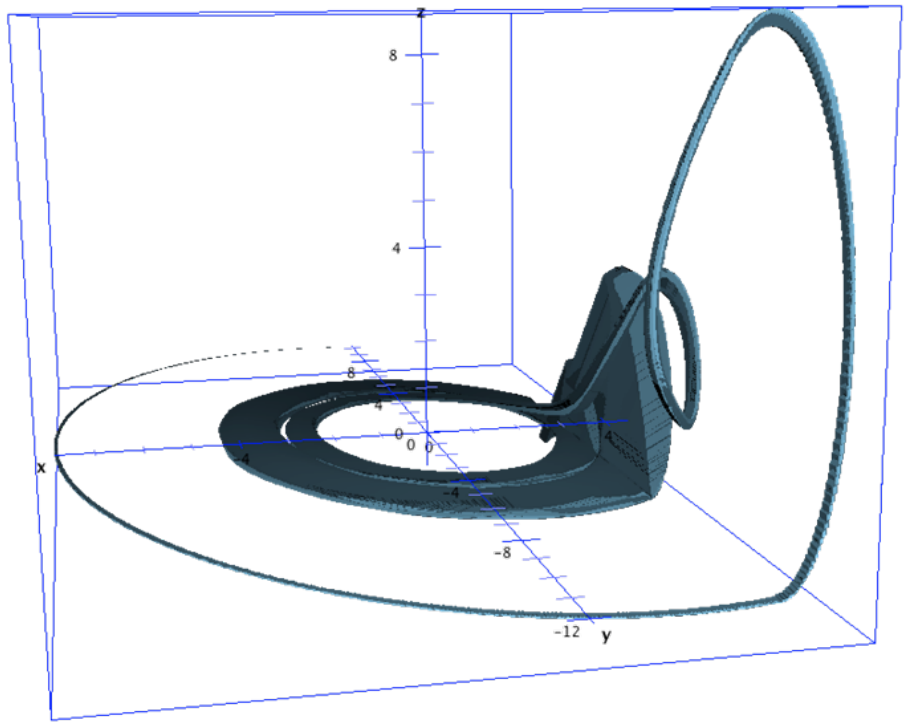
- New algorithms and data structures can handle linear and piecewise-linear systems with 100-200 state variables
- Small nonlinear systems (under development)
- Integrated in a tool, **SpaceEx: The State-Space Explorer**
- Has a model editor and web interface and is available for download at `http://spaceex.imag.fr`
- Applied to examples in control systems, biology and circuits



The State-Space Explorer (SpaceEx)

- Example: a chaotic circuit





Relation between IP and Behavioral Models

- An interesting research question
- Motivation for behavioral models seems to be twofold:
  - Keep IP confidential
  - Export models that can be integrated in higher-level simulation without delaying it
- How to **define** and **establish** the relationship between IP and behavioral level models of the **same** device?
- Can this be done automatically by **abstraction** methods used elsewhere or by **black-box** identification?
- Can the assertion language be used to summarize the input-output behavior of the device?
- Since such a specification is under-determined by nature, how to use such an abstract model in a simulation?

Conclusions

- The verification of AMS circuits is only in its infancy - like digital verification 20 years ago
- Some of the problems solved by researchers are **auto-generated**, some are coming from **sporadic** interactions with circuit and EDA people
- A better interaction is needed between providers of verification techniques and their potential users
- Most urgent task: identify some issues which
  - Are very important for designers
  - Can benefit from a systematic validation methodology
  - Can be handled by already existing verification techniques or their immediate extensions
- Thank you