On Timing Analysis of (Huge) Acyclic Circuits

Ramzi Ben Salah, Marius Bozga, Oded Maler

VERIMAG

Grenoble, France

FORMATS Workshop, Marseille, September 2003

Nutrition Facts

1) Circuits and timing

2) Timed automata for circuits

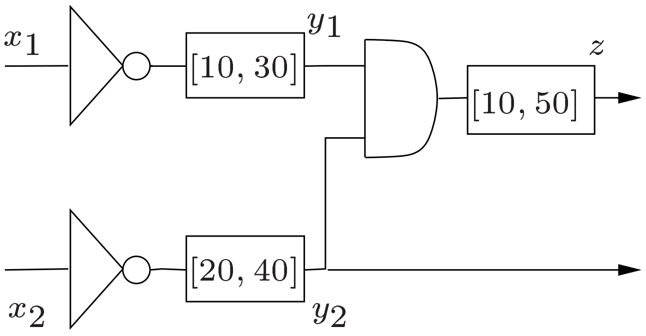
3) Timed Automata Verification Basics

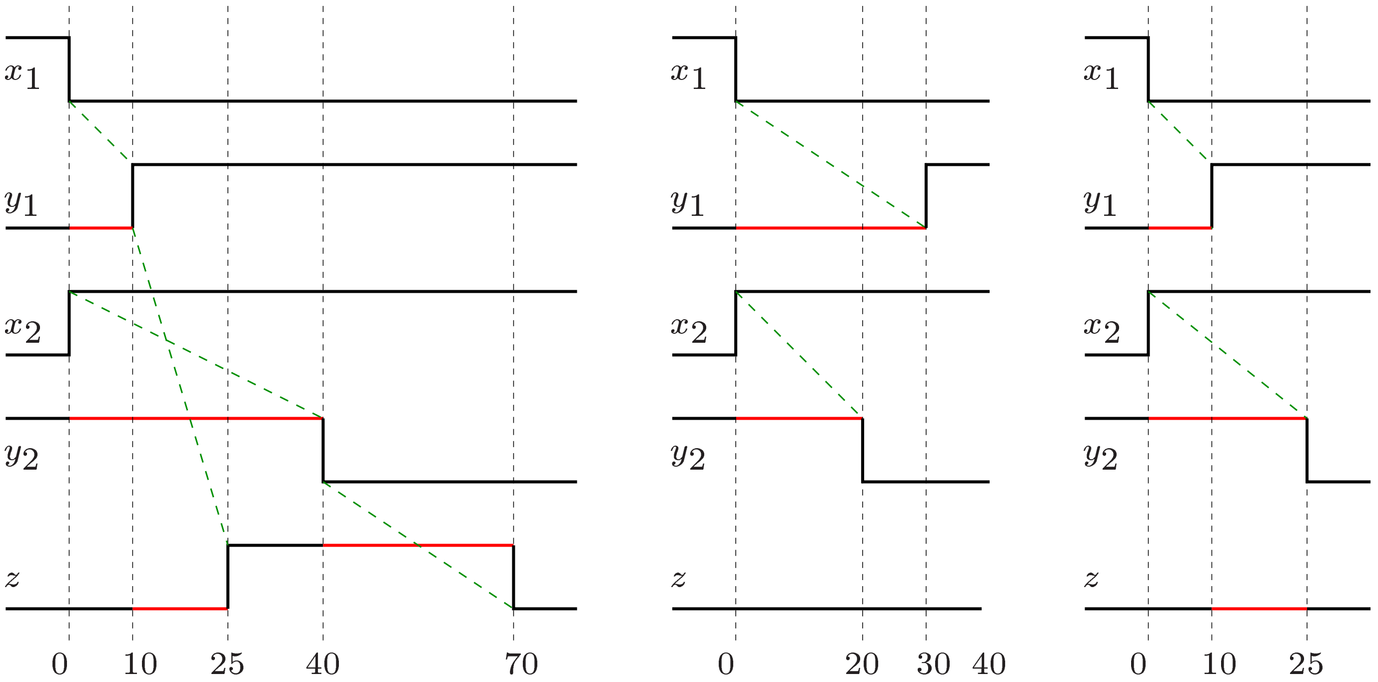
4) Abstraction of acyclic circuits and automata

5) Experimental results

6) Conclusions and future work

Circuits with Bi-bounded Inertial Delays





Acyclic Circuits

Start with a stable states. Primary inputs change only once at the beginning

This induces a non-countable number of possible behaviors

Each behavior admits a finite number of changes and goes to a stable state within a bounded amount of time (sum of the upper bounds on the delays along the longets path)

We want to compute the maximal stabilization time, i.e. the stabilization time for the worst behavior

Two sources of complexity: the exponential number of input pairs and the non-countable number of behaviors that each of these pairs induces. We focus on the second.

Common Approaches

1) Simulation with many choices of delays (non-exhaustive).

2) Static timing analysis: computing the longest path.

This is like abstracting away all the logic and assuming that every change in every input to every gate will trigger a change in the output.

In reality this is not the case and there are various techniques for detecting "false paths".

We want to do it with timed automata, an ideal model for expressing the interaction between logic and timing (this was the motivation, by the way, in [Dill 1989]).

Modeling Circuits with Timed Automata

Our modeling approach, based on [Maler and Pnueli 95], is the following:

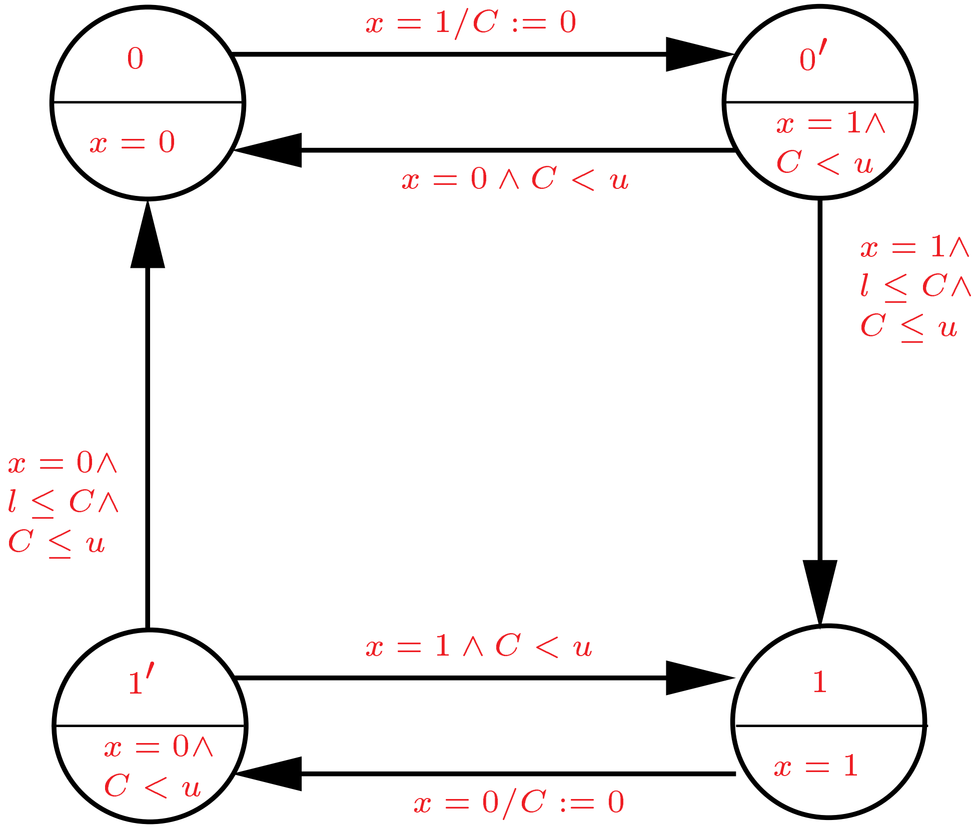
Decompose any gate into an instantaneous Boolean function and a bi-bounded (non-deterministic) inertial delay element.

Model every delay element 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 whose set of runs corresponds exactly to the set of all possible behaviors of the circuit (under all possible combinations of choices of delays).

Already applied to some MOS devices, asynchronous circuits (Sergio Yovine, Marius Bozga, Stavros Tripakis) and Domino circuits (Nishant Sinha). Mostly still toy problems due to clock and state explosion.

The Delay Automaton



Timed Automata are n-Tuples...

A timed automaton is $\mathcal{A} = (Q, C, I, \Delta)$

Q: a set of states, C: a set of clocks,

I: staying condition (invariant), assigning to every q a conjunction I_q of inequalities of the form $c \leq u$, for some clock c and integer u

Δ: a transition relation consisting of tuples (q, ϕ, ρ, q') where q and q' are states,

$\rho \subseteq C$ is the set of clocks reset by the transition, and

ϕ (the transition guard) is a conjunction of formulae of the form $c \geq l$ for some clock c and integer l.

A clock valuation is a function $\mathbf{v} : C \to \mathbb{R}_+ \cup \{0\}$ and a configuration is a pair $(q, \mathbf{v})$ consisting of a discrete state (location) and a clock valuation.

Runs of Timed Automata

A *step* of the automaton is one of the following:

- A discrete step: $(q, \mathbf{v}) \xrightarrow{\delta} (q', \mathbf{v}')$, for some transition $\delta = (q, \phi, \rho, q') \in \Delta$, such that $\mathbf{v}$ satisfies ϕ and $\mathbf{v}' = \mathrm{Reset}_\rho(\mathbf{v})$.

- A time step: $(q, \mathbf{v}) \xrightarrow{t} (q, \mathbf{v} + t\mathbf{1})$, $t \in \mathbb{R}_+$ such that $\mathbf{v} + t\mathbf{1}$ satisfies I_q.

A run of the automaton starting from a configuration $(q_0, \mathbf{v}_0)$ is a finite sequence of steps

$$\xi : \quad (q_0, \mathbf{v}_0) \xrightarrow{t_1} (q_1, \mathbf{v}_1) \xrightarrow{t_2} \cdots \xrightarrow{t_n} (q_n, \mathbf{v}_n).$$

Symbolic Reachability Computation

A *symbolic state* is (q, Z) where q is a discrete state and Z is a zone, a set of clock valuations satisfying a conjunction of inequalities $c_i - c_j \geq d$ or $c_i \geq d$.

Symbolic states are closed under the following operations:

The *time successor* of (q, Z), the configurations reachable from (q, Z) by letting time progress without violating the staying condition of q:

$$Post^t(q, Z) = \{(q, \mathbf{z} + r\mathbf{1}) : \mathbf{z} \in Z, r \geq 0, \mathbf{z} + r\mathbf{1} \in I_q\}.$$

The δ*-transition successor* of (q, Z) is the configurations reachable from (q, Z) by taking the transition $\delta = (q, \phi, \rho, q') \in \Delta$:

$$Post^\delta(q, Z) = \{(q', \mathrm{Reset}_\rho(\mathbf{z})) : \mathbf{z} \in Z \cap \phi\}.$$

The Reachability Graph

The δ-*successor* of a time-closed symbolic state (q, Z) is the set of configurations reachable by a δ-transition followed by passage of time:

$$Succ^{\delta}(q, Z) = Post^{t}(Post^{\delta}(q, Z)).$$

Starting from a state s, the forward reachability algorithm for TA generates the reachability (simulation) graph, a directed graph $S = (N, \rightarrow)$ obtained by starting at $Post^{t}(s, \{\mathbf{0}\})$ and applying $Succ^{\delta}$ until termination.

There is a path from (q, Z) to (q', Z') in S iff for every $\mathbf{v}' \in Z'$ there exists $\mathbf{v} \in Z$ and a run of $\mathcal{A}$ from $(q, \mathbf{v})$ to $(q', \mathbf{v}')$.

Hence the union of all symbolic states in S is exactly the set of reachable configurations. This is the basic algorithm in tools such as Kronos and Uppaal.

Finding the Maximal Stabilization Time

A global state of an automaton $\mathcal{A}$ that corresponds to a circuit is called unstable if at least one of the components is in an excited state ($0'$ or $1'$).

We augment $\mathcal{A}$ with an additional clock T which is never reset to zero and hence it measures absolute time. For acyclic circuits the value of T is bounded in any unstable state.

Doing forward reachability on the augmanted automaton we find the maximal stabilization time as the maximal value of T in any reachable symbolic state (q, Z) with q unstable.

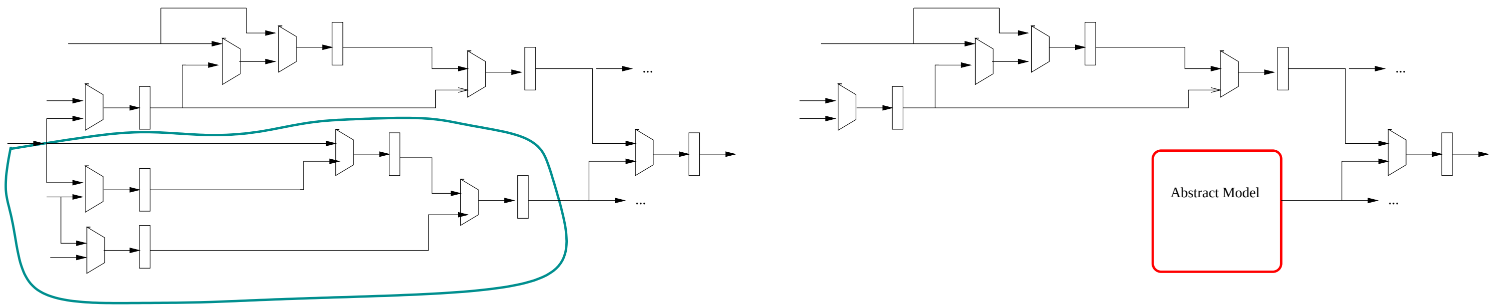
This solves the problem, but only for small circuits (less than 10 gates) so we need to reinvent something new.

Divide, Approximate and, hopefully, Conquer

The basic idea: take a sub-circuit on the left, use TA technology to generate an approximate timed model of its output.

The approximation should be safe (a super set of the observable behaviors of the sub-circuit) and much smaller then its exact model.

It is then plugged as an input model to the rest of the circuit.



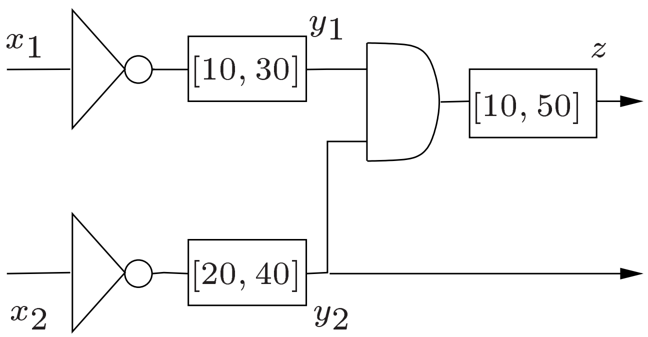
The Nature of the Abstraction

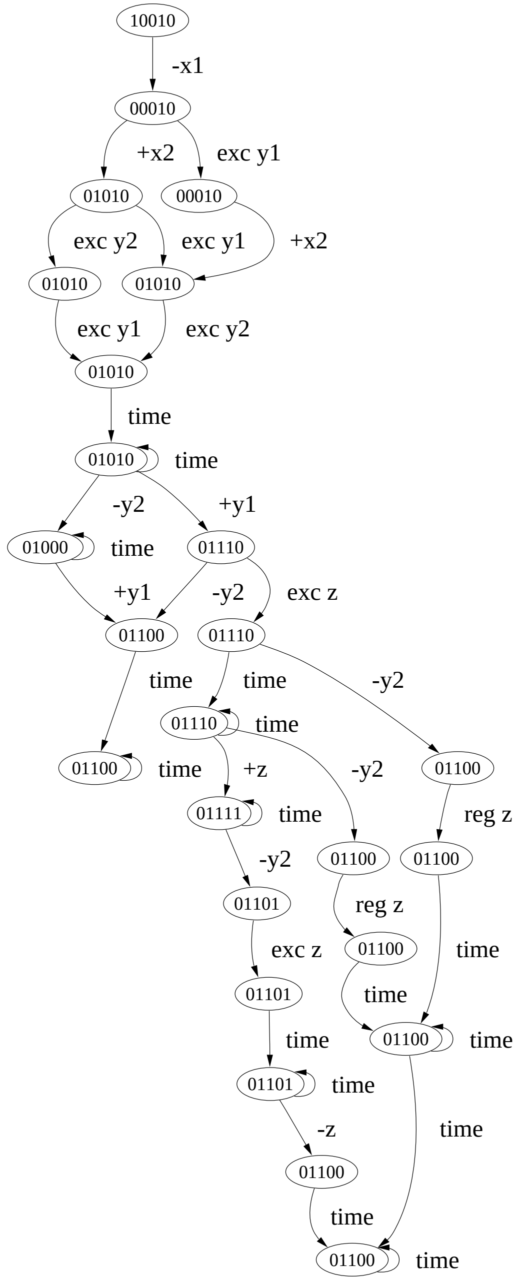
First, the obvious thing: hiding internal actions such as excitation and “regrets” of the outputs and all transitions of internal wires.

Relaxation of timing constraints by allowing things to happen at impossible times (but not in impossible orders!)

The reachability graph can be viewed as an expansion of the TA, on one hand splitting some discrete states according to clock values, and removing transitions that are infeasible due to timing constraints.

By associating with each symoblic state (q, Z) the staying condition Z and with each outgoing transition the intersection of Z with the guard we obtain a TA equivalent to the original one.

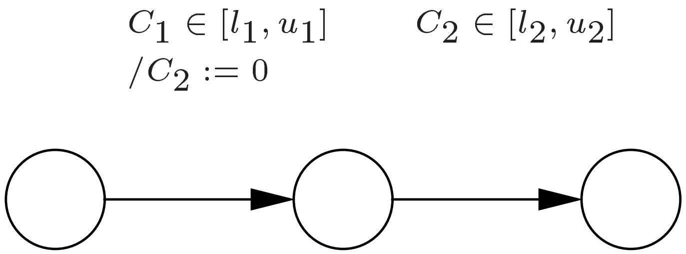

x_1
$[10, 30]$
y_1
z
$[10, 50]$
$[20, 40]$
x_2
y_2

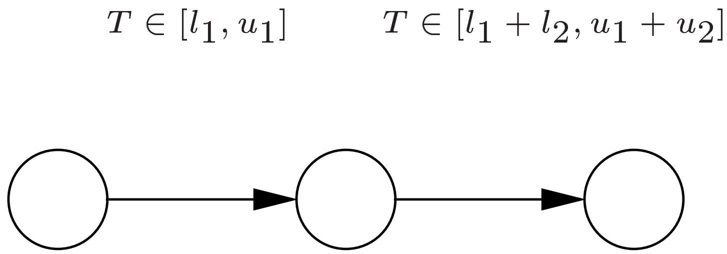

10010
-x1
00010
+x2
exc y1
01010
00010
exc y2
exc y1
+x2
01010
01010
exc y1
exc y2
01010
time
01010
time
-y2
+y1
01000
time
01110
+y1
-y2
exc z
01100
01110
time
time
-y2
01110
time
01100
time
+z
-y2
01100
01111
time
reg z
-y2
01100
01100
01101
reg z
exc z
01100
time
01101
time
time
01100
time
01101
time
time
-z
01100
time
01100
time

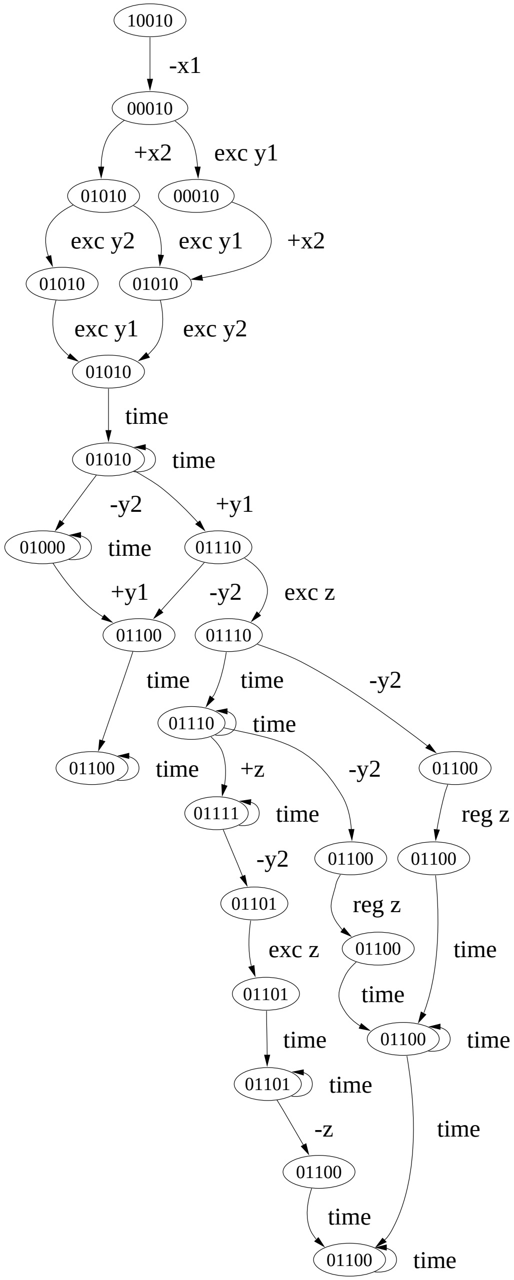
Relaxing Timing Constraints

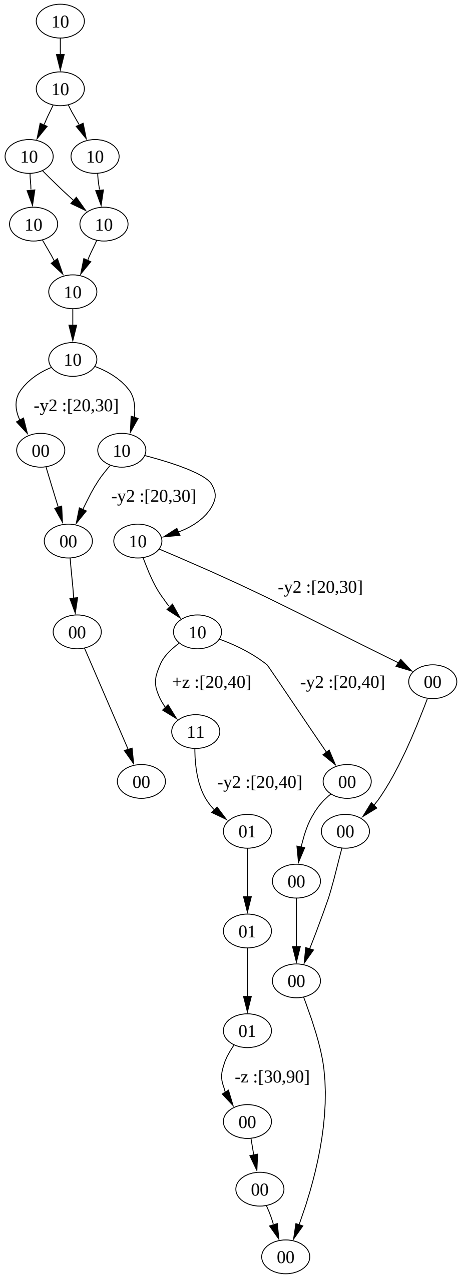
We take the TA obtained from the reachability graph and project on a subset of the clocks. The constraints related to the other clocks are removed.

For acyclic circuits it is natural to project only on the auxiliary clock T that measures absolute time. This way we keep the information about the time each transition can be taekn (but lose some inter-dependence information).





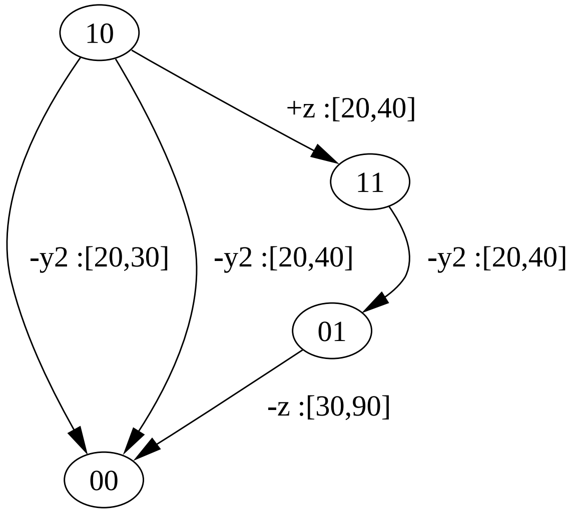

10010
-x1
00010
+x2
exc y1
01010
00010
exc y2
exc y1
+x2
01010
01010
exc y1
exc y2
01010
time
01010
time
-y2
+y1
01000
time
01110
+y1
-y2
exc z
01100
01110
time
time
-y2
01110
time
01100
time
+z
-y2
01100
01111
time
reg z
-y2
01100
01100
01101
reg z
exc z
01100
time
01101
time
time
01100
time
01101
time
-z
time
01100
time
01100
time


-y2 :[20,30]
-y2 :[20,30]
-y2 :[20,30]
+z :[20,40]
-y2 :[20,40]
-y2 :[20,40]
-z :[30,90]

Hiding and Minimization

Unobservable transitions are replaced by τ and the automaton is minimized by merging states that are equivalent according the following relation:

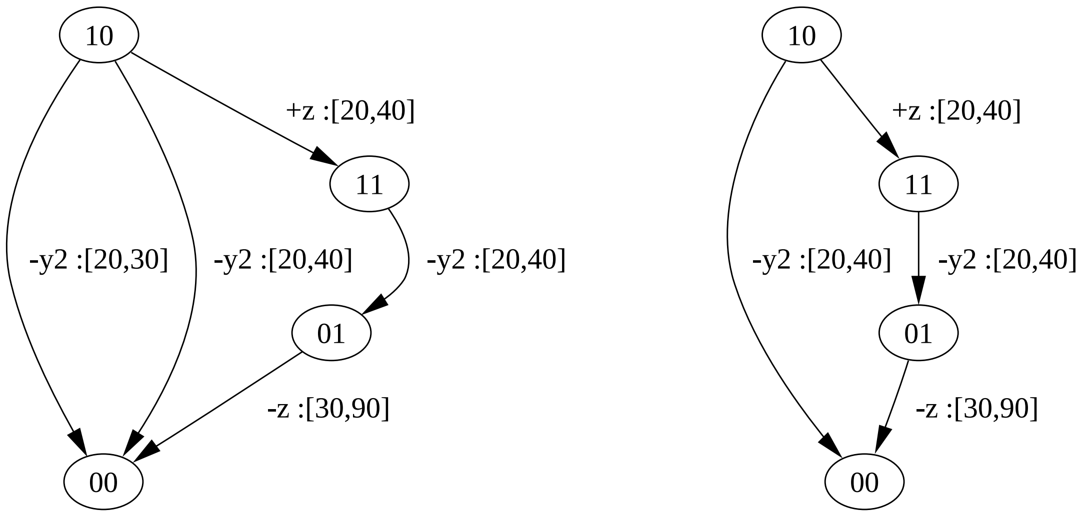
$$q_1 \sim q_2 \ \text{ iff } \ \forall \delta, I \ \ q_1 \stackrel{\tau^* \cdot (\delta, I)}{\longrightarrow} q_1' \Rightarrow (\exists q_2' \ \text{ s.t. } \ q_2 \stackrel{\tau^* \cdot (\delta, I)}{\longrightarrow} q_2' \ \wedge \ q_1' \sim q_2')$$



More Aggressive Minimization

The previous relation treats intervals syntactically. The following relation merges them

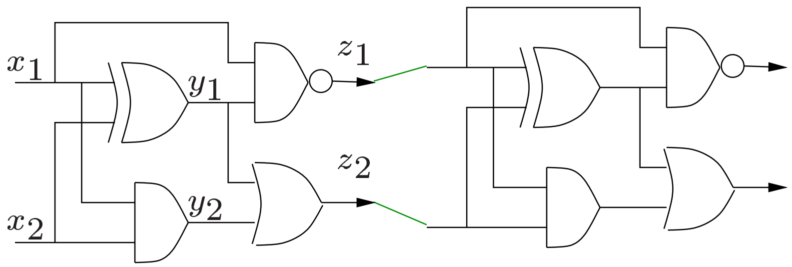
$$q_1 \sim' q_2 \ \text{ iff } \ \forall \delta, I \ \ q_1 \overset{\tau^* \cdot (\delta, I)}{\longrightarrow} q_1' \Rightarrow (\exists q_2', I' \ \text{ s.t. } \ q_2 \overset{\tau^* \cdot (\delta, I')}{\longrightarrow} q_2' \ \wedge \ q_1' \sim' q_2')$$



Experimental Results

We took a circuit with 2 inputs, 4 gates and 2 outputs and applied our techniqUe to n concatenation of the circuit with itself assuming x_1 may rise anywhere in $[10, 35]$ and x_2 in $[15, 63]$.

We successively abstract the circuits $1 \ldots n-1$ to be used as input to circuit n



The results are sensitive to $[l, u]$: the larger is $(u-l)/l$, more types of behaviors are possible.

$[l, u]$	$[1, 2]$			$[10, 12]$			$[100, 102]$		
no.	states	min	time	states	min	time	states	min	time
1	71	4	0:01	65	3	0:01	65	3	0:01
2	934	12	0:02	270	7	0:02	270	7	0:02
3	–	–	–	2690	11	0:03	2690	11	0:04
4				5397	23	0:05	4080	16	0:06
5				217951	144	9:44	21498	30	0:12
6				–	–	–	50543	39	0:30
7							73502	48	1:01
8							95619	57	1:54
9							117736	66	3:12
10							139853	75	5:08
11							161970	84	7:32
12							184087	93	10:05
13							206204	102	14:42
14							228321	111	20:39
15							250438	120	28:15
16							272555	129	36:46
17							117736	138	49:36
18							316789	147	1:04:04
19							338906	156	1:21:48
20							361023	165	1:42:59
21							383140	174	1:58:56
22							405257	183	2:30:31
23							–	–	–

Discussion

We pushed the capabilities of TA-based circuit verification toward 100 gates (clocks and state variables)

Bottlneck is memory usage during reachability computation – can be improved by exploiting acyclicity.

Hard to compare to TA-analysis without abstraction because it is not feasible

We don't know what to do with the explosion in the number of input pairs combination except by enumeration when their number is small. Probably the application domain will be control circuits with a restricted number of admissible inputs — not adders and multipliers.

Variations

How to partition a given circuit? Which variables to group together in the same abstract model?

Projecting on absolute time is only one possibility. Other clocks (or even new auxliary clocks) can be kept in the abstraction according to designers' intuition concerning their importance.

In general, the more abstract (and small) is the model in stage n, the more complex behaviors it generates in stage $n+1$ and fine-tuning is needed to find the best trade-off.

Future and Related Work

In this work we always worked with a closed system, a system containing the model of its input.

Extending the abstraction technique to open system is a major challenge. It will allow "hierarchical" verification and timing analysis and a timed characterization of "components" in general.

Some related work has been done by Tasiran, Brayton, Alur and Kurshan [96, 97, manual abstraction, assume-guarantee] and recently by Zheng, Mercer and Myers [2003, automatic safe transformation of timed Petri nets, trace theory]

We are working on it

Credits

The theory and the tools are based on the Verimag toolset IF and its two predecessors Kronos (for timed automata) and CADP (for enumerative varification and minimization).

These tools incorporate algorithms and software developed by numerous people over almost two decades, including J. Sifakis, S. Graf, J.-C. Fernendez, A. Bouajjani, C. Rodrigez, H. Garavel, L. Mounier, A. Kerbrat, T. Henzinger, X. Nicollin, S. Yovine, A. Olivero, C. Daws, S. Tripakis and M. Bozga.

N. Sinha, F. Gariet, B. Diop and O. Ben Sik Ali worked on previous versions of the abstraction technique.

This research was supported by the EU project AMETIST (advanced methods for timed systems) and by a grant from Intel (combining formal methods and timing analysis) supervised by Avi Efrati and Ken Stevens.