

Verification of Asynchronous Circuits using Timed Automata

Marius Bozga, Hou Jianmin, Oded Maler and Sergio Yovine

VERIMAG, Centre Equation, 2, av. de Vignate, 38610 Gières, France

Abstract

In this work we apply the timing verification tool OpenKronos, which is based on timed automata, to verify correctness of numerous asynchronous circuits. The desired behavior of these circuits is specified in terms of signal transition graphs (STG) and we check whether the synthesized circuits behave correctly under the assumption that the inputs satisfy the STG conventions and that the gate delays are bounded between two given numbers. Our results demonstrate the viability of the timed automaton approach for timing analysis of certain classes of circuits.

1 Introduction

Today most of circuit verification and analysis is done while maintaining a separation between the *logical* functionalities of a circuit and the *delay* properties of its components. For clocked synchronous circuits, the size of the clock cycle can be determined by computing the accumulated delays along the longest path from inputs to latches. Assuming that the cycle time is sufficiently large, the functional verification of the circuit can proceed by ignoring gate and wire delays and by treating the whole circuit at the abstraction level of an untimed sequential machine. While this division of labor makes circuit design and verification a more tractable process, it makes it more difficult to satisfy the ever-growing demands for more performance. The reason is that in reality *logic* and *timing* have complex mutual interactions, and two different realizations of the same combinational function, having the same path length can differ significantly in their maximal stabilization times. The path length only gives an upper-approximation of the propagation delay, taking into account worst-cases which are, more often than not, impossible when logic is taken into account (“false paths”).

A lot of *asynchronous circuits* [MB59,U69,KKTV93,BS94] design has been done within the *speed-independent* paradigm. The desired behavior of a circuit

¹ Email: {bozga, maler, yovine}@imag.fr

is specified as a kind of “protocol” between the circuit and its environment. This protocol does not assume two distinct phases in every operation cycle (arrival of inputs and computation of next-state and output) and hence the circuit specification cannot be decomposed naturally into a combinational function and a memory.² The major burden in asynchronous design is to detect occurrences of certain subsets of events (which may appear in various orders) which are sufficient for triggering further events in the circuit. This approach requires a large silicon investment in event-detection mechanisms and it has been observed, e.g. [MM93,CKK⁺98], that by taking delay information into account, many behaviors anticipated by the speed-independent design cannot actually happen and the size of the circuit can be reduced significantly by putting such behaviors in the “don’t-care” category.

These and other observations call for a formal model in which the interaction between logic and delays can be expressed naturally, and which can serve as a basis for design and validation tools that take advantage of this expressive power. Timed automata [AD94] constitute such a model. These are automata augmented with auxiliary clock variables whose role in the model is to measure the time elapsed since the occurrence of certain events. Using these clocks, the phenomenon of *uncertain but bounded delay* between two or more events can be expressed in a very natural manner. Of course, timed automata (henceforth TA) inherit from automata the capability to model any complex discrete dynamics and hence they are more expressive than models based on timed marked graphs whose analysis can be performed using the Max-Plus algebra. Indeed, it was shown [D89,L89,MP95] that circuits with bi-bounded gate or wire delays can be transformed into networks of timed automata which can serve as a basis for simulation, verification and automatic design. Several tools for TA verification have been implemented [LPY97,DOTY96] and applied to various problems, including timing analysis of circuits [MY96,BMPY97,TB97,TKB97,TKY⁺98,BMT99]. Alternative models which are used to address the same class of problem are based on some variants of timed Petri nets [BD91,HB95,BM98,SY95,YR99,KB99,ZM00] and it will be interesting to compare them with the TA-based approach both in terms of modeling and expressivity and in terms of underlying computational difficulty.

This work describes the application of the TA-based verification methodology and the tool OpenKronos [BDM⁺98] to the verification of asynchronous circuits. We take two dozens of typical asynchronous circuits realized by gates having bi-bounded delays. Using standard TA reachability methods we attempt to verify that these circuits behave according to their specifications. Our performance results indicate how far one can go by applying brute-force verification to the rich TA model (we were able to verify circuits with up to 17 gates) and from where you need to augment verification with a compo-

² This is not the case in burst-mode circuits which are out of the scope of this paper.

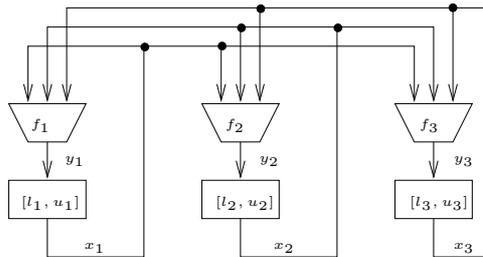


Fig. 1. A circuit with delays.

sitional methodology and with specialized techniques that take advantage of the special structure of the sub-class of TA that correspond to circuits.

The rest of the paper is organized as follows: in Section 2 we describe how we model bi-bounded delays using timed automata and how timing verification is applied to these models. In Section 3 we illustrate, using an example, how the joint behavior of the circuit and of its STG specification are converted into a timed automaton and analyzed by OpenKronos. Finally, the verification results for the benchmark examples are reported in Section 4.

2 Modeling Delays with Timed Automata

In this section we sketch informally our approach for modeling circuits with bi-bounded delays using timed automata [MP95,MY96,BMT99]. We view a circuit as a network consisting of Boolean gates and (non-deterministic) delay elements as in Figure 1. A Boolean gate can be viewed as a memoryless function from signals to signals. Each delay element is characterized by an interval $[l, u]$ of lower- and upper-bounds on the propagation times of events from the input to the output (wire delays can be modeled as a special case where the Boolean function is the identity). We assume that the delays are *inertial*: changes that do not persist for l time are filtered away. More refined delay models can be defined at the price of more complex analysis. Due to uncertainty a delay element can transform an input signal into uncountably-many different output signals, as demonstrated in Figure 2, and hence the corresponding operator $D_{[l,u]}$ is non-deterministic, i.e. set-valued. The semantics of the circuit is the set of all solutions of a system of equations and inclusions on signals of the form:

$$y_i = f_i(x_1, \dots, x_n) \quad x_i \in D_{[l_i, u_i]}(y_i)$$

We translate every equation into a timed automaton whose set of behaviors coincides with the set of solutions of the equation and the composition of all these automata generates exactly all the possible behaviors of the circuit under all possible choices of delays. The automaton for a Boolean gate $y_i = f_i(x_1, \dots, x_n)$ is simply a one-state automaton which generates all the tuples satisfying the equation. Each delay element of the form $x \in D_{[l,u]}(y)$ is

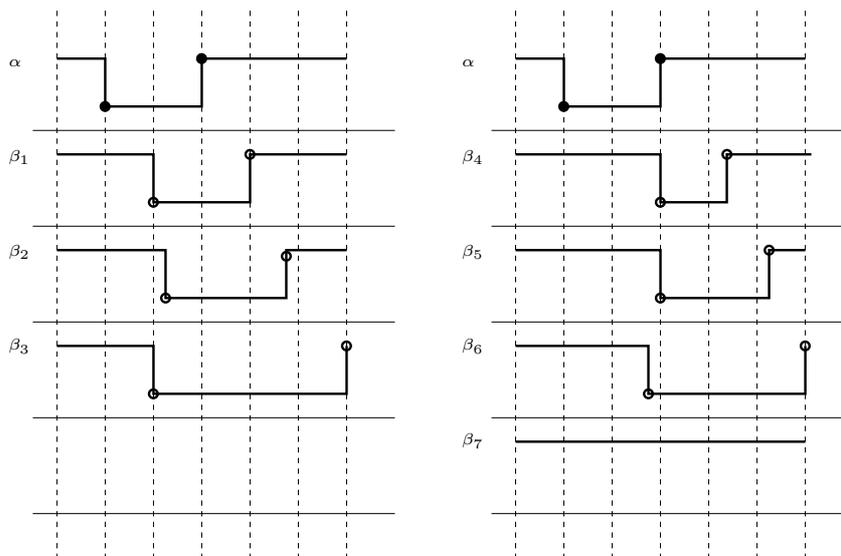


Fig. 2. An input signal α and a sample $\{\beta_1, \dots, \beta_7\}$ of the set $D_{[1,3]}(\alpha)$ of its delayed outputs.

modeled by one timed automaton with 4 states and one clock as depicted in Figure 3. State $(0, 0)$ is a *stable* state where the input y and the output x are both 0. As soon as the input y changes to 1, a transition to the *excited* state $(1, 0)$ is made and a clock C is reset to zero and starts measuring the time since the event. The transition from $(1, 0)$ back to $(0, 0)$ signifies a “regret” of the input *before* the propagation of the event to the output. Such regret transitions can be avoided in certain models which assume that the input behaves according to some protocol, or be replaced by an “error” transition if the design methodology disallows such phenomena.³ When at state $(1, 0)$, if the clock value crosses the lower bound l , the output *can* change to 1 and the automaton moves to the stable state $(1, 1)$. However, as long as the upper bound u has not been reached, the automaton may stay in $(1, 0)$. The ability to express and analyze this temporal uncertainty is the main feature of TA. Unlike deterministic models used in SPICE simulation, a circuit modeled using such bi-bounded delay elements and their corresponding TA will have *many* behaviors, even in the presence of a single input signal. However *all* these behaviors can be captured using geometric methods based on the possible ranges of the values of clock variables. The generators of input signals can also be modeled as timed automata, expressing various restrictions on the inputs such as timing bounds on their frequency or some protocols of interaction with the circuit that they are assumed to follow. By combining these automata with those that model the circuit, it is possible, in principle, to simulate *all the possible behaviors of the circuit, in the presence of all admissible inputs and choices of delays* and hence lift formal verification methodology from untimed

³ A more realistic model of inertial delays might require more states and clocks and is currently under investigation.

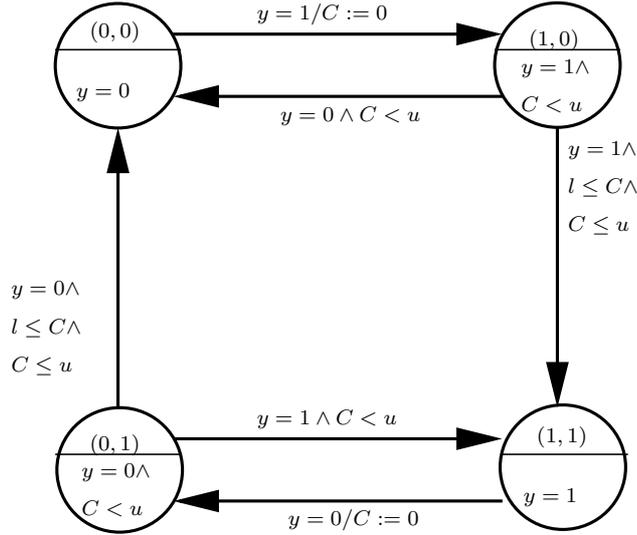


Fig. 3. The timed automaton for a delay element. The runs of the automaton are exactly those satisfying $y \in D_{[l,u]}(x)$.

to timed circuit models.

As an illustrative example consider the two independent oscillators appearing in Figure 4. Suppose that initially they are both in state 0 and hence the reachability analysis starts at global state $(0, 0)$ with clocks at $(0, 0)$. The product automaton may stay at $(0, 0)$ as long as none of the clocks has crossed its corresponding upper-bound. In this example, where $u_1 < u_2$, the set of clock values reachable via time passage at state $(0, 0)$ is $\{(x_1, x_2) : x_1 = x_2 \leq u_1\}$. By intersecting this set with the transition guard $C_1 \geq l_1$ we obtain the set $\{(x_1, x_2) : l_1 \leq x_1 = x_2 \leq u_1\}$ which denotes all the clock valuations in which the transition from $(0, 0)$ to $(1, 0)$ is enabled. Since this transition resets C_1 we may reach $(1, 0)$ at any point in the clock space belonging to $\{(0, x_2) : l_1 \leq x_2 \leq u_1\}$. From there, by time passage, we may reach the set $\{(x_1, x_2) : l_1 \leq x_2 \leq u_2 \wedge l_1 \leq x_2 - x_1 \leq u_1\}$, and this set, in turn, can be intersected with the condition $C_2 \geq l_2$ for moving to $(1, 1)$ etc. The reader can find formal definitions of TA reachability analysis in [A99, Y98].

From a theoretical standpoint all the interesting problems concerning TA (and circuits modeled by them) can be solved algorithmically. These problems include absence of hazards, bounded response properties, absence of shortcuts in transistor models, conformance with communication protocols and many other properties currently classified under different sub-topics in circuit design. Other problems which can be formulated and theoretically solved are the controller synthesis problem (the automatic derivation of delay parameters and transition conditions in order to guarantee satisfaction of certain properties) and the time-optimal controller synthesis problem (choosing parameters and conditions that will lead the automaton into a set of states as soon as possible, e.g. into the set of stable states in a combinational circuit). However, due to the complexity of TA analysis, many researchers and practitioners prefer less

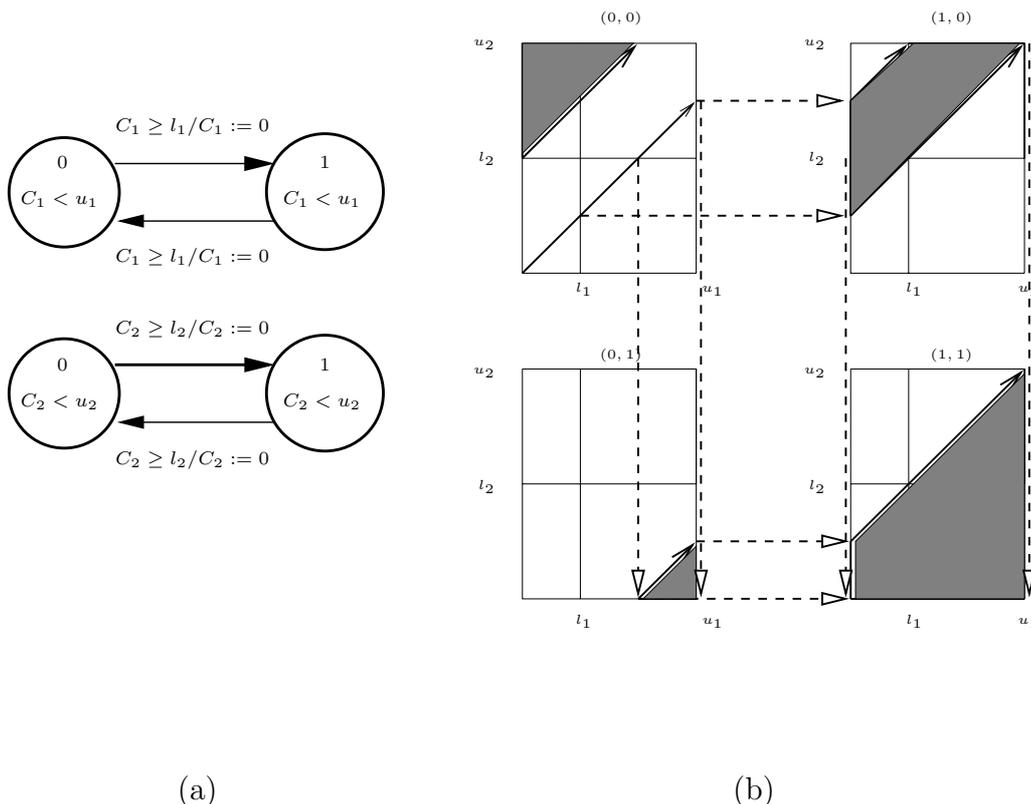


Fig. 4. (a) Two TA representing two independent oscillators. (b) The first steps in computing all their possible behaviors. Dashed lines indicate discrete transitions.

expressive but more tractable models. We believe that in the long run it is better to separate considerations of modeling adequacy from more pragmatic considerations related to tool performance. In other words, it is better to have first a general model which describes the phenomenon in question in a faithful manner and only later to devise various techniques in order to overcome verification complexity. Our strategy is thus to use the full TA model and see what is the largest chunk of circuitry that can be wholly analyzed using TA technology, before resorting to abstraction and approximation techniques.

3 Modeling and Verification of Asynchronous Circuits

We have applied OpenKronos to several benchmark examples of asynchronous circuits taken from [PCKP00]. The intended behaviors of these circuits were specified using *signal transition graphs* (STGs), which are essentially Petri nets whose transitions are labeled by events corresponding to rising and falling of signals. An STG represents a “protocol” of interaction between a component and its environment. As an example, consider the circuit `half` which realizes a half handshake between two adjacent stages in a pipeline. The circuit has two input signals `Ri` and `Ai` and two output signals `Ro` and `Ao`. The behavior is specified by the STG of Figure 5-(b). This specification defines only a partial-

order among events and is indifferent, for example, to the order between $Ao+$ and $Ai+$. The marking graph of this specification is the automaton of Figure 5-(c) which accepts all the linearizations of the partial-order. It is assumed that the environment respects the specification (e.g. Ai will not rise before Ro goes up). We want to verify whether the circuit implementation behaves properly, that is, the Ao and Ro events take place only when they are allowed by the STG.

The circuits realizing the specifications were synthesized as follows. Initially the asynchronous synthesis tool Petrify [CKK⁺97] was used to transform the STG specifications into circuits built from complex gates with arbitrary fan-in. These circuits are speed-independent by construction and hence do not need verification. These circuits were then transformed using the tool SIS [SSL⁺92] into circuits realized by gates from standard cell libraries with fan-in 2. The circuit synthesized for the `half` specification is depicted in Figure 5-(d) and it has five internal variables in addition to inputs and outputs.

This transformation does not take into account potential hazards that could be generated by internal signals. Our goal is to prove that they behave, nevertheless, correctly, under certain assumptions of gate delays (assumed to be in the interval [27, 33]) and timing constraints for the external environment.

According to the principles described in the previous section the circuit is modeled as a product of timed automata with a clock for each gate – in this case 7 clocks. This timed automaton description is generated automatically from the circuits. The STG specification is translated automatically into an untimed automaton isomorphic to the marking graph, with error transitions added for every output event and a state in which it is not enabled (e.g. event $Ro-$ induces an error transition from state 3 in the automaton of Figure 5-(c)). Additional timing constraints on the inter-arrival times of the input events are modeled using an additional automaton and a clock for each input signal.

The verification problem that we pose is whether the set of all time-constrained behaviors of the circuit contains a behavior not included in the semantics of the STG. Technically this question is equivalent to whether an error transition is reachable in the composition of all the abovementioned automata. For the `half` circuit, if we assume no timing restrictions on the inputs, we find the following error trace:

$$\begin{array}{l} 27 \text{ Ro+ } Ai+ \ 27 \text{ g3- } g2+ \text{ Ao+ } Ri- \ 27 \text{ g3+ } 27 \text{ g4+} \\ \text{Ro- } Ai- \ 27 \text{ g2- } g1- \ 27 \text{ Ao- } Ri+ \text{ Ro+ } Ai+ \text{ g2+ } 27 \text{ Ro-} \end{array}$$

In this trace, Ro goes up after 27 time units and this is followed immediately by rising of Ai . Then after more 27 time the output of gate 3 falls and that of gate 2 rises, and so on, until finally $Ro-$ occurs before being enabled by $Ao+$. On the other hand, if we assume further that the any two changes of an input variable are separated by some time in [900, 1111], the circuit is proved correct (similar results under this last assumption were obtained in [PCKP00]).

all which might be reachable). The analysis is performed on the product of this TA with the automata for the STG specification and the automata that model the time-constrained inputs (OpenKronos generates the product “on-the-fly”). For each circuit we have tried to compute the “simulation graph” (see [Y98]) whose states are pairs of the form (q, F) where q is a discrete state and F is a polyhedral subset of the clock space. Depending on the temporal complexity of the automaton, the size of this graph might be significantly larger than the number of discrete states. Computing the simulation graph amounts to computing all the reachable states of the TA, and this computation is needed to prove that the circuit is correct. For incorrect circuits bugs can usually be found much before the completion of this computation. As table 1 shows, we were able to perform this exhaustive analysis to 15 circuits out of 21. For the remaining 6, we were able to compute around 500000 symbolic states in about 10 minutes with the available memory (all the results were obtained on a SUN UltraSparc 10 with 2GB of memory). Among these we found, nevertheless, bugs in two, namely `tsend-bm` and `mr1`. These results were obtained using the standard reachability analysis algorithm for timed automata, unlike the approach of [PCKP00], which inspired our work, where a special heuristic which alternates between timed and untimed analysis is applied. Note that our verification results differ from those of [PCKP00] because they consider any “regret” transition in internal gates as an error transition, while we accept such behaviors as long as the STG specifications concerning observable behaviors are respected.

The ability of OpenKronos to treat such non-trivial asynchronous circuits is a source of optimism concerning the future applicability of TA analysis to timing verification. We believe that if these results could be achieved without any heuristic, much larger circuits could be verified by combining the verification engine of OpenKronos with general and circuit-specific abstraction and approximation techniques [B96,AIKY95,WD94,TAKB96,ZM00], combination of timed and untimed verification [PCKP00], relative timing [SGR99,KB99], partial-order methods [BM98] and other techniques reported in the literature. **Acknowledgment:** We thank Jordi Cortadella and Marco Pena for providing us with the benchmarks and for many discussions. Ken Stevens, Mike Kishinevski and Luciano Lavagno answered various questions concerning asynchronous circuits.

References

- [A99] R. Alur, Timed Automata, *Proc. CAV'99* LNCS 1633, 8-22, Springer, 1999.
- [AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.
- [AIKY95] R. Alur, A. Itai, R.P. Kurshan and M. Yanakakis, Timing Verification by

no.	name	gates	states	transitions	time(sec)	correct
1	alloc_outbound	11	313	366	0.09	Y
2	chu133	9	2580	3390	0.63	Y
3	converta	12	891	1129	0.19	Y
4	dff	6	753	1160	0.19	N
5	ebergen	9	661	846	0.14	Y
6	half	7	1990	3041	0.41	Y
7	mp_forward_pkt	10	807	1076	0.24	Y
8	nowick	10	208	245	0.05	Y
9	rcv_setup	6	1213	1469	0.22	Y
10	rpdf	8	10934	13554	2.93	Y
11	sbuf_ram_write	17	50510	83313	31.77	Y
12	sbuf_read_ctl	10	451	572	0.13	Y
13	sbuf_send_ctl	13	1741	2300	0.49	Y
14	sbuf_send_pkt2	13	115	138	0.07	N
15	vme	12	2209	2519	0.39	Y
16	mr1	16	490938	638558	607.43	N
17	tsend_bm	12	503406	765214	589.56	N
18	mmu	22	475228	710353	595.09	?
19	mr0	20	545022	662768	593.24	?
20	ram_read_sbuf	17	647890	911249	678.48	?
21	trimos_send	24	516149	693547	580.33	?

Table 1

The performance results for the benchmark asynchronous circuits. The number of states and transition are those of the simulation graph and the time figures correspond to the duration of computing this graph.

Successive Approximation, *Information and Computation* 118, 142-157, 1995.

[AMP98] E. Asarin, O. Maler and A. Pnueli, On the Discretization of Delays in Timed Automata and Digital Circuits, in R. de Simone and D. Sangiorgi (Eds), *Proc. Concur'98*, LNCS 1466, 470-484, Springer, 1998.

[B96] F. Balarin, Approximate Reachability Analysis of Timed Automata, *Proc. RTSS'96*, 52-61, IEEE, 1996.

- [BD91] B. Berthomieu and M. Diaz, Modeling and Verification of Time Dependent Systems using Time Petri Nets, *IEEE Trans. on Software Engineering* 17, 259-273, 1991.
- [BM98] W. Belluomini and C.J. Myers, Verification of Timed Systems Using POSETs, in A.J. Hu and M.Y. Vardi (Eds.), *Proc. CAV'98*, 403-415, LNCS 1427, Springer, 1997.
- [BDM⁺98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine, Kronos: a Model-Checking Tool for Real-Time Systems, *Proc. CAV'98*, LNCS 1427, Springer, 1998.
- [BMPY97] M. Bozga, O. Maler, A. Pnueli, S. Yovine, Some Progress in the Symbolic Verification of Timed Automata, in O. Grumberg (Ed.) *Proc. CAV'97*, 179-190, LNCS 1254, Springer, 1997.
- [BMT99] M. Bozga, O. Maler and S. Tripakis, Efficient Verification of Timed Automata using Dense and Discrete Time Semantics, in L. Pierre and T. Kropf (Eds.), *Proc. CHARME'99*, 125-141, LNCS 1703, Springer, 1999.
- [BS94] J.A. Brzozowski and C-J.H. Seger, *Asynchronous Circuits*, Springer, 1994.
- [CKK⁺97] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno and A. Yakovlev, Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers, *IEICE Transactions on Information and Systems*, Vol. E80-D, No. 3, March 1997, pages 315-325.
- [CKK⁺98] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Taubin and A. Yakovlev, Lazy Transition Systems: Application to Timing Optimization of Asynchronous Circuits, in *Proc. ICCAD'98*, 324-331, 1998.
- [D89] D.L. Dill, Timing Assumptions and Verification of Finite-State Concurrent Systems, in J. Sifakis (Ed.), *Automatic Verification Methods for Finite State Systems*, LNCS 407, 197-212, Springer, 1989.
- [DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, The tool Kronos, in "Hybrid Systems III, Verification and Control", LNCS 1066, Springer, 1996.
- [HB95] H. Hulgaard and S.M. Burns, Efficient Timing Analysis of a Class of Petri Nets, *Proc. CAV'95*, 1995.
- [KB99] H. Kim and P.A. Beerel, Relative Timing Based Verification of Timed Circuits and Systems, *Proc. IWLS'99*, June 1999.
- [KKTV93] M. Kishinevsky, A. Kondratyev, A. Taubin and V. Varshavsky, *Concurrent Hardware: The Theory and Practice of Self-Timed Design*, Wiley, 1993.

- [LPY97] K.G. Larsen, P. Pettersson and W. Yi, UPPAAL in a Nutshell, *Software Tools for Technology Transfer* 1/2, 1997.
- [L89] H.R. Lewis, Finite-state Analysis of Asynchronous Circuits with Bounded Temporal Uncertainty, TR15-89, Harvard University, 1989.
- [MP95] O. Maler and A. Pnueli, Timing Analysis of Asynchronous Circuits using Timed Automata, in P.E. Camurati, H. Evekking (Eds.), *Proc. CHARME'95*, LNCS 987, 189-205, Springer, 1995.
- [MY96] O. Maler and S. Yovine, Hardware Timing Verification using KRONOS, In *Proc. 7th Israeli Conference on Computer Systems and Software Engineering*, Herzliya, Israel, June 1996.
- [MM93] C.J. Myers and T. H.-Y. Meng, Synthesis of Timed Asynchronous Circuits, in *IEEE Transactions on VLSI Systems* 1, 106-119, 1993.
- [MB59] D.E. Muller and W.S. Bartky, A theory of Asynchronous Circuits, in *Proc. of an Int. Symposium on the Theory of Switching*, 204-243, Harvard University Press, 1959.
- [PCKP00] M.A. Pena, J. Cortadella, A. Kondratyev and E. Pastor, Formal Verification of Safety Properties in Timed Circuits, *Proc. Async'00*, 2-11, IEEE Press, 2000.
- [RM94] T.G. Rokicki and C.J. Myers, Automatic Verification of Timed Circuits, *Proc. CAV'94*, June, 1994.
- [SY95] A. Semenov and A. Yakovlev, Verification of Asynchronous Circuits based on Timed Petri Net Unfolding, *Proc. TAU'95*, 199-210, 1995.
- [SGR99] K.S. Stevens, R. Ginosar, and S. Rotem, Relative Timing, *Proc. Async'99*, 1999.
- [SSL⁺92] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton and A. Sangiovanni-Vincentelli, SIS: A System for Sequential Circuit Synthesis, Technical Report UCB/ERL M92/41, University of California, Berkeley, May 1992.
- [TAKB96] S. Tasiran, R. Alur, R.P. Kurshan and R. Brayton, Verifying Abstractions of Timed Systems, in *Proc. CONCUR'96*, 546-562, Springer, 1996.
- [TB97] S. Tasiran and R.K. Brayton, STARI: A Case Study in Compositional and Hierarchical Timing Verification, in O. Grumberg (Ed.) *Proc. CAV'97*, 191-201, LNCS 1254, Springer, 1997.
- [TKB97] S. Tasiran, Y. Kukimoto and R.K. Brayton, Computing Delay with Coupling using Timed Automata, *Proc. TAU'97*, 1997.

- [TKY⁺98] S. Tasiran, S. P. Khatri, S. Yovine, R.K. Brayton and A. Sangiovanni-Vincentelli, A Timed Automaton-Based Method for Accurate Computation of Circuit Delay in the Presence of Cross-Talk, *FMCAD'98*, 1998.
- [U69] S.H. Unger, *Asynchronous Sequential Switching Circuits*, Wiley, 1969.
- [WD94] H. Wong-Toi and D.L. Dill, Approximations for Verifying Timing Properties, in T. Rus and C. Rattray (Eds.), *Theories and Experiences for Real-Time System Development*, World Scientific Publishing, 1994.
- [YR99] T. Yoneda and H. Ryu, Timed Trace Theoretic Verification using Partial Order Reduction, *Proc. Async'99*, 108-121, IEEE, 1999.
- [Y98] S. Yovine, Model-checking timed automata, in G. Rozenberg and F. Vaandrager (Eds.), *Lectures on Embedded Systems*, LNCS 1494, Springer, 1998.
- [ZM00] H. Zheng and C.J. Myers, Automatic Abstraction for Synthesis and Verification of Deterministic Timed Systems, *Proc. TAU'2000*, December, 2000.