Approximate Algorithms for Time Separation of Events *

Supratik Chakraborty David L. Dill
Computer Systems Laboratory, Stanford University, Stanford, CA 94305

Abstract

We describe a polynomial-time approximate algorithm for computing minimum and maximum time separations between all pairs of events in systems specified by acyclic timing constraint graphs. Even for acyclic graphs, the problem is NP-complete. We propose finding an approximate solution by first approximating the non-convex feasible space with a suitable convex "envelope", and then solving the problem efficiently in the approximate convex space. Unlike previous works, our algorithm can handle both min and max type timing constraints in the same system, and has a computational complexity that is polynomial in the number of events. Although the computed separations are conservative in the worst-case, experiments indicate that our results are highly accurate in practice.

1. Introduction

Finding bounds on the time separation between events has emerged as a central problem in analysis of hardware designs. The results of such an analysis can be used for interface timing verification [7]; for design optimization of timed asynchronous circuits [8]; for determining delay padding for eliminating hazards in asynchronous circuits [6]; for obtaining optimal clock schedules in synchronous circuits [9]; or for performance analysis of concurrent systems.

In this paper, we propose a new *polynomial-time approximate algorithm* for finding the minimum and maximum time separations between *all pairs* of events in a concurrent system, given bounded propagation delays between events, and the type of timing constraint (min or max) associated with each event. Unlike several previous works [1, 7, 10, 2, 5], our algorithm has a worst-case time complexity that is *polynomial* in the number of events, *and* can handle *both* min and max type constraints in the same system. A limitation of our current approach is that it applies only to systems representable by *acyclic* timing constraint graphs. However, this still encompasses a large and important class of applications.

It has been shown by previous researchers [7] that the general form of the problem is NP-complete. The computational difficulty stems from the non-convex shape of the feasible space of a system of constraints involving min and max functions. We, therefore, propose to first approximate the non-convex feasible space with a convex "envelope". Since every point in the feasible space is contained in the approximate convex space, maximizing the separation between two events in the approximate space yields an upper bound on the maximum achievable separation in the original feasible space. Ideally, we would like to find the smallest convex space (convex hull) that contains the feasible space. Unfortunately, finding a convex hull in n dimensions is itself intractable. So, we propose a new approximation methodology for obtaining a small convex envelope of the non-convex feasible space.

Our method proceeds by representing each min (or max) constraint with an *equivalent* combination of linear inequalities among the variables. We then successively derive other linear inequalities from this equivalent combination, ensuring that at each step, the feasible space of the derived system of inequalities contains the feasible space of the set of inequalities from which it is derived. The resulting system of linear inequalities, with a feasible space that is necessarily convex, then constitutes an approximation to the original min (or max) constraint. We then use a polynomial-time algorithm to find bounds on the separation between every pair of events in the approximate convex space. Although our computed bounds represent conservative approximations to the true bounds in the worst-case scenario, experiments on several large systems have shown that our results are highly accurate in practice.

1.1. Related work

Previous work in this area comes from a wide spectrum of contexts. McMillan and Dill [7] proposed a cubic-time algorithm for the acyclic max-only constraint problem. A branch-and-bound algorithm for systems with both min and max constraints was also proposed in [7]. However, this has a worst-case exponential time complexity. Vanbekbergen et al. proposed an algorithm for systems with both min and max constraints in [10]. Unfortunately, their algorithm is incorrect. The work of [8] computes loose bounds on the time separation of events in cyclic graphs containing only max type events. Burks and Sakallah formulated the problem as a *min-max linear program* (mmLP) and proposed two exact algorithms in [2]. Unfortunately, both algorithms have worst-case exponential time complexity. The work of Hulgaard et al. [5] proposed an interesting algorithm for computing the exact maximum separation between two *specific*

* This work was supported by a grant from the Semiconductor Research Corporation (Contract No. 96-DJ-389) and a gift from Sun Microsystems

events in a cyclic process graph with max type events. Their algorithm, however, has a worst-case exponential time complexity. Extensions of this algorithm to deal with extended free choice and unique choice have been proposed in [4]. Lavagno and Sangiovanni-Vincentelli used a variant of the present problem to find optimum delay paddings to eliminate hazards in asynchronous circuits [6]. Several other timing analyzers have been developed that either cater to specific applications [1] or deal with fixed delays between events [3].

2. The timing analysis problem

The setup for our timing analysis problem is as follows: We have a concurrent system where event i occurs at time t_i, and $\delta_{i,j}$ represents the delay in the propagation of event i to event j. The delays $\delta_{i,j}$ are constrained to lie within fixed upper and lower bounds $d_{i,j}$ and $D_{i,j}$. Each event i is also associated with a min or max operator that determines how its time of occurrence relates to those of other events.

Such a system of constraints can be represented by a timing constraint graph where nodes represent events and directed edges represent dependencies between the occurrence times of events. A directed edge from event i to event j is labeled $[d_{i,j}, D_{i,j}]$, representing the delay $\delta_{i,j}$. If an event f is associated with a max (min) operator, the node corresponding to f is labeled max (min). As an example, Figure 1 shows the timing constraint graph for the following system of timing constraints:

$$\begin{array}{lcl}
t_a & = & t_a + \delta_{s,a}; \\
t_b & = & \min(t_s + \delta_{s,b},\ t_a + \delta_{a,b}); \\
t_c & = & \max(t_a + \delta_{a,c},\ t_b + \delta_{b,c}) \\
t_d & = & \min(t_a + \delta_{a,d},\ t_b + \delta_{b,d}) \\
0 & \leq & \delta_{s,a},\ \delta_{a,b},\ \delta_{a,c},\ \delta_{b,c},\ \delta_{a,d},\ \delta_{b,d}\ \leq\ 1 \\
1 & \leq & \delta_{s,b} \leq 3
\end{array}$$

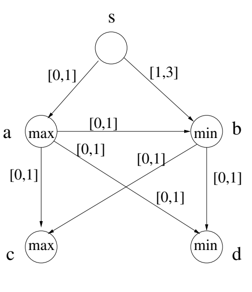

Figure 1. Example of a timing constraint graph

Given such a timing constraint graph, our goal is to determine bounds on the minimum and maximum achievable values of $(t_i - t_j)$ for every pair of events i and j. Since $\min(t_i - t_j) = -\max(t_j - t_i)$, it suffices to determine the maximum value of $(t_i - t_j)$ for every *ordered pair* of events (i, j). We restrict our analysis to acyclic timing constraint graphs, which can, however, have multiple *source* events (events with no incoming edges), provided bounds on the separations between the source events are given.

2.1. An approximation strategy

The feasible space of a system of constraints involving min and max functions is non-convex. This renders standard convex optimization techniques inapplicable. In this section, we describe a method to compute a convex envelope of the non-convex feasible space.

Mathematically, each non-linear equation of the form:

$$t_f \quad = \quad \max(t_i + \delta_{i,f},\ \ldots\ t_k + \delta_{k,f}) \qquad (1)$$

can be represented using the following *equivalent* system of linear inequalities:

$$\begin{array}{c}
(t_f \geq t_i + d_{i,f}) \qquad \wedge \\
\vdots \qquad \wedge \\
(t_f \geq t_k + d_{k,f}) \qquad \wedge \\
\left(\begin{array}{ll} (t_f \leq t_i + D_{i,f}) & \vee \\ \vdots & \vee \\ (t_f \leq t_k + D_{k,f}) & \end{array} \right)
\end{array} \qquad (2)$$

where $\wedge$ represents conjunction and $\vee$ represents disjunction of the truth or falsity of each linear inequality.

Now, let s be an arbitrary event in the system, and let $\Delta_{s,i}$ represent an upper bound (achievable or not) of the separation $(t_i - t_s)$. It follows that $t_i \leq t_s + \Delta_{s,i}$. Combining this with $t_f \leq t_i + D_{i,f}$ (see inequalities (2)), we obtain $t_f \leq t_s + \Delta_{s,i} + D_{i,f}$.

Reasoning analogously, we can relate t_s to the occurrence times of the other predecessors of f. This gives us the following system of linear inequalities derived from (2):

$$\begin{array}{c}
(t_f \geq t_i + d_{i,f}) \qquad \wedge \\
\vdots \qquad \wedge \\
(t_f \geq t_k + d_{k,f}) \qquad \wedge \\
\left(\begin{array}{ll} (t_f \leq t_s + \Delta_{s,i} + D_{i,f}) & \vee \\ \vdots & \vee \\ (t_f \leq t_s + \Delta_{s,k} + D_{k,f}), & \end{array} \right)
\end{array} \qquad (3)$$

The disjunction of inequalities in (3) can be further approximated to give:

$$t_f \quad \leq \quad t_s + \max_{l\ \in\ preds(f)} (\Delta_{s,l} + D_{l,f}), \qquad (4)$$

where we have used the notation $preds(f)$ to denote the set of events l that have a directed edge to f in the timing constraint graph. Since the event s was arbitrarily chosen, inequality (4) holds for *all* events in the system. However, choosing s to be the same as f does not yield any useful information, since t_f appears on both sides of inequality (4). Therefore, applying inequality (4) to all events other than f, we get the following system of linear inequalities as an approximation to the non-linear equation (1).

$$\begin{array}{c}
(\wedge_{i\ \in\ preds(f)}\ (t_f\ \geq\ t_i + d_{i,f})) \qquad \bigwedge \\
(\wedge_{all\ s \neq f}\ (t_f\ \leq\ t_s + \max_{l\ \in\ preds(f)} (\Delta_{s,l} + D_{l,f}))).
\end{array} \qquad (5)$$

This leads to the following important observation.

Observation 1 *: $-d_{i,f}$ is an upper bound of $\Delta_{f,i}$, for all $i \in preds(f)$. Similarly,* $\max_{l \in preds(f)} (\Delta_{s,l} + D_{l,f})$ *is an upper bound of $\Delta_{s,f}$, for all $s \neq f$.*

A min constraint $t_f = \min(t_i + \delta_{i,f}, \ldots t_k + \delta_{k,f})$ can be similarly approximated, leading to the following observation:

Observation 2 *: $D_{i,f}$ is an upper bound of $\Delta_{i,f}$ for all $i \in preds(f)$. Similarly,* $\max_{l \in preds(f)} (\Delta_{l,s} - d_{l,f})$ *is an upper bound of $\Delta_{f,s}$ for all events $s \neq f$.*

Note that any solution of the original non-linear equation (e.g., equation (1)) is also a solution of the corresponding approximate system of linear inequalities (e.g., inequalities (5)). Since the feasible space of the latter is necessarily convex, we have effectively determined a convex envelope of the non-convex feasible space of the non-linear equation. For example, consider the following system of of timing constraints:

$$\begin{array}{lcl} t_3 & = & \max(t_1 + \delta_{1,3},\ t_2 + \delta_{2,3}) \\ 1 & \leq & \delta_{1,3}, \delta_{2,3} \ \leq \ 3 \\ \Delta_{1,2} & = & \Delta_{2,1} \ = \ 1 \end{array}$$

Figure 2(a) shows the feasible space of this system. The volume contained within the solid V-shaped entity and extending infinitely in both directions along the turning-corner of the V represents points where this system of constraints is satisfied. Our approximation gives the following system of linear inequalities for this system:

$$t_3 \geq t_1 + 1; \quad t_3 \leq t_1 + 4; \quad t_3 \geq t_2 + 1;$$
$$t_3 \leq t_2 + 4; \quad -1 \leq t_2 - t_1 \leq 1$$

The feasible space of this system of inequalities is shown in Figure 2(b). Clearly this includes the non-convex space of Figure 2(a).

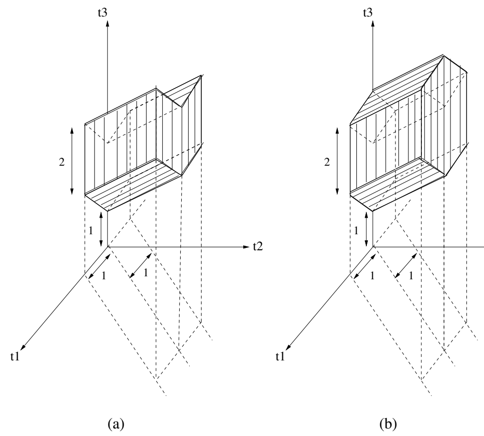

Figure 2. Illustrating the inclusion of the original feasible space in the approximate convex space.

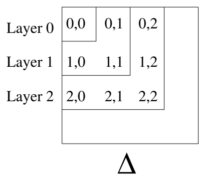

Figure 3. Layer wise construction of the Δ matrix

2.2. A polynomial-time approximate algorithm

In this section, we use observations 1 and 2 of Section 2.1 to design a polynomial-time algorithm for computing bounds on the separation between every pair of events. Our algorithm accepts as inputs a finite, acyclic timing constraint graph with n events, and bounds on the separation between the source events. The events are topologically sorted with event 0 being a source event. The output of our algorithm is an $n \times n$ matrix Δ, where $\Delta_{i,j}$ contains an upper bound on $(t_j - t_i)$, i.e., $t_j - t_i \leq \Delta_{i,j}$.

The Δ matrix is computed in an incremental fashion as follows: We start off with $\Delta_{0,0}$ which trivially has the value 0. We then compute $\Delta_{1,0}, \Delta_{1,1}$ and $\Delta_{0,1}$ (see Figure 3). Next, we compute $\Delta_{2,0}, \Delta_{2,1}, \Delta_{2,2}, \Delta_{0,2}$ and $\Delta_{1,2}$ (see Figure 3). This process is repeated for increasing values of the topological index, effectively constructing the Δ matrix "layer-by-layer" as shown in Figure 3. The i^{th} such "layer" of the Δ matrix is comprised of the entries $\Delta_{j,i}$ and $\Delta_{i,j}$, where $0 \leq j \leq i$.

The rationale behind this layer-wise construction is as follows: All events in $preds(i)$ have topological indices less than i. It therefore follows from observations 1 and 2 in Section 2.1 that initial estimates of $\Delta_{s,i}$ and $\Delta_{i,s}$, for all $s \in \{0, \ldots, i-1\}$, can be computed using the entries in the $(i \times i)$ sub-matrix of Δ comprised of rows 0 through $(i-1)$ and columns 0 through $(i-1)$. Moreover, since the timing constraint graph is acyclic, the timing of event i cannot affect the maximum separation between events with indices less than i. Consequently, the entries in the $(i \times i)$ sub-matrix mentioned above are not affected by the timing of event i. This suggests that we first compute the entries in the $(i \times i)$ sub-matrix comprised of rows 0 through $(i-1)$ and columns 0 through $(i-1)$, and then use the computed entries in this sub-matrix to compute the i^{th} layer of the Δ matrix. Applying this method for increasing values of i from 1 through $(n-1)$ yields the Δ matrix layer-by-layer.

Figure 4 shows the steps of our timing analysis algorithm. We initialize the Δ matrix and then compute upper bounds on the maximum separation times a layer at a time. For each layer i, we first compute initial estimates of as many entries as possible using observation 1 or 2 of Section 2.1 (depending on the type of operator associated with event i). Thereafter, we use these estimates and previously computed entries in layers $\leq i$ to compute the other entries in layer i. This computation is similar to finding the shortest path to (from) event i from (to) other events $< i$. If, during this computation, we find that the value of an entry can be further refined using observation 1 or 2 of Section 2.1, we perform this refinement in the steps marked **(*)** in Figure 4.

Initialization:
for each i in 0 to $(n-1)$
  for each j in 0 to $(n-1)$
    if $(i == j)$ $\Delta_{i,i} = 0$;
    else if (i, j are source events)
      $\Delta_{i,j} =$ given value;
    else $\Delta_{i,j} = +\infty$;

Main Algorithm:
for each i in 1 to $(n-1)$
  1. Compute_Initial_Estimates(i);
  2. **if** (i is max type)
    for each j in 0 to $(i-1)$
      $\Delta_{i,j} = \min_{k \in preds(i)}(\Delta_{k,j} - d_{k,i})$;
      if (j is max type)
(*) $\Delta_{i,j} = \min(\Delta_{i,j},\ \max_{k \in preds(j)}(\Delta_{i,k} + D_{k,j})$;
    else if (i is min type)
    for each j in 0 to $(i-1)$
      $\Delta_{j,i} = \min_{k \in preds(i)}(\Delta_{j,k} + D_{k,i})$;
      if (j is min type)
(*) $\Delta_{j,i} = \min(\Delta_{j,i}, \max_{k \in preds(j)}(\Delta_{k,i} - d_{k,j})$;

Compute_Initial_Estimates(i: index)
if (i is max type)
  for each j in $preds(i)$ $\Delta_{i,j} = -d_{j,i}$;
  for each j in 0 to $(i-1)$
    $\Delta_{j,i} = \max_{k \in preds(i)}(\Delta_{j,k} + D_{k,i})$
else
  for each j in $preds(i)$ $\Delta_{j,i} = D_{j,i}$;
  for each j in 0 to $(i-1)$
    $\Delta_{i,j} = \max_{k \in preds(i)}(\Delta_{k,j} - d_{k,i})$

Figure 4. Time separation of events algorithm

The complexity of our algorithm is $O(n^2.p)$, where $p = \max_{all\ events\ i} |preds(i)|$. In practice, p is usually bounded by a small constant. Therefore, our algorithm has $O(n^2)$ complexity for all practical purposes.

2.3. An example

We now apply our algorithm to the timing constraint graph of Figure 1. The Δ matrix generated is shown in Table 1. It may be verified that all our computed bounds are exact.

	s	a	b	c	d
s	0	1	2	3	2
a	0	0	1	2	1
b	0	0	0	1	1
c	0	0	0	0	1
d	0	0	1	2	0

Table 1. Δ matrix for our example.

In Section 2.1, we saw that each min or max constraint can be equivalently represented by the conjunction and disjunction of simple linear inequalities (see inequalities (2)). As a simple approximation, one may choose to disregard the disjunctive constraints, since taking these into account turns the feasible space non-convex. Applying this simple approximation to the system represented in Figure 1 gives:

$$t_a \geq t_s + 0; \quad t_a \leq t_s + 1; \quad t_b \leq t_a + 1; \quad t_b \leq t_s + 3;$$
$$t_c \geq t_a + 0; \quad t_c \geq t_b + 0; \quad t_d \leq t_a + 1; \quad t_d \leq t_b + 1.$$

This system admits the solution: $t_s = t_a = t_b = t_c =$ any large positive number, and $t_d = 0$. Therefore, an upper bound on $(t_b - t_d)$ subject to this system of constraints is $+\infty$. In reality, however, the achievable upper bound of $(t_b - t_d)$ in Figure 1 is 1.

To understand what went amiss, observe that $t_d = min(t_a + [0, 1],\ t_b + [0, 1])$ yields the system of inequalities: $(t_d \leq t_a + 1) \wedge (t_d \leq t_b + 1) \wedge ((t_d \geq t_b + 0) \vee (t_d \geq t_b + 0))$. With $t_d = 0$ and $t_b = t_a =$ a large positive number, both the disjunctive clauses are violated. Therefore, this cannot be a solution to the system. This shows that it is important not to disregard the disjunctive constraints in inequations (2). Our algorithm takes these into account by means of the approximations described in Section 2.1.

3. Experimental results

We have implemented our timing analysis algorithm and applied it to several concurrent system specifications. Our benchmarks include cyclic constraint graphs from [5] and [8] unfolded to varying degrees. These are named "hr-n" and "mr-n" in Table 2, where different values of r indicate different cyclic graphs, and n denotes the degree of unfolding. The benchmark"mcmill" was obtained from [7], while "diffeq" was obtained from an analysis of an asynchronous differential equation solver chip [11]. The other benchmarks were obtained as systems of timing constraints during timing simulation of ISCAS85 circuits. Table 2 shows some of our analysis results. For each benchmark, we report the number of events (n), and the number of nontrivial min/max constraints (with $\geq$ 2 arguments) in the system. This gives us a basis for comparing our run times with those of [2].

In order to quantify the accuracy of our results, we define the accuracy of our algorithm by computing the fraction of all time separations that we are able to compute exactly. For problems involving only max constraints, we obtain the exact separations using the algorithm of [7]. Comparing these with the bounds computed by our algorithm shows that our algorithm produces exact results for all benchmarks involving only max constraints. Unfortunately, we currently don't have an implementation of an exact algorithm for solving systems with both min and max functions. Consequently, we are unable to report the accuracy of our results for benchmarks involving both min and max functions (an exception is "mcmill", which was small enough to allow manual checking). However, we believe that our results are highly accurate in these cases as well, since our algorithm does not treat max-only systems in any special way.

Table 2 also shows the time taken by our algorithm to analyze each benchmark. These times are on a MIPS R5000 150 MHz processor with 128 MBytes of main memory, and do not include the time to read in the timing constraint graph and topologically sort the events. We find that for the max-only problem, our algorithm is approximately 3 times faster than the algorithm of [7]. For systems with both min and max functions, our run times may be compared with those of [2]. Table 3 lists the run times for some of the largest systems analyzed in [2]. Since the computing platforms are not the same, a direct comparison would be unfair. Nevertheless, an inspection of the CPU times in Tables 2 and 3 indicates that for systems of comparable sizes, our algorithm computes **all $\mathbf{n^2}$** separations in much less time than that required by the algorithm of [2] to compute the maximum value of **only one objective function**. Unfortunately, the exact benchmarks used in [2] were not available for analysis.

Bench-mark Name	Evts. (n)	Min/max fns.	t_1 (s)	t_2 (s)	Accu-racy
h1-64	321	316	0.49	1.48	1.0
h1-256	1281	1276	8.65	26.31	1.0
h1-512	2561	2556	36.69	113.77	1.0
h2-64	385	379	0.75	2.21	1.0
h2-256	1537	1531	13.17	40.31	1.0
h2-512	3073	3067	55.63	171.86	1.0
m1-64	641	255	1.55	4.97	1.0
m1-256	2561	1023	27.87	92.05	1.0
m1-512	5121	2047	120.69	417.89	1.0
mcmill	13	5	0.00	N/A	1.0
diffeq	258	48	0.21	N/A	N/A
c1355	948	537	2.78	N/A	N/A
c7552	5258	2343	133.14	N/A	N/A
c6288	4189	3356	65.38	N/A	N/A

Evts. (n) : # events,
Min/max fns. : # min/max functions with $\geq$ 2 arguments,
t_1 : Time for $\mathbf{n^2}$ separations by our method on R5000,
t_2 : Time for $\mathbf{n^2}$ separations by [7] on R5000

Table 2. Computing all pairs of separations

Benchmark Name	Min/max functions	Time (s) for **one objective function** on DEC5000 ([2])
s444	30	0.574
s1423	74	15.5
s9234	227	105.9
s38584	1452	1965.7

Table 3. Optimizing one linear function

In summary, our experiments indicate that our algorithm produces results with a high degree of accuracy while remaining efficient. The power of our approach lies in the ability to achieve high degrees of accuracy and efficiency even when analyzing large systems containing *both* min and max type timing constraints.

4. Conclusion

We have proposed a new polynomial-time approximate algorithm for finding minimum and maximum separations between events in concurrent systems. Unlike several previous approaches, our algorithm handles *both* min and max type constraints in the same system and has a complexity that is polynomial in the number of events. While the separations computed by our algorithm are conservative in the worst-case, experiments show that our results achieve a high degree of accuracy. The results of such an analysis can be used in the analysis, optimization and verification of concurrent and asynchronous systems.

5. Acknowledgments

We are thankful to Kenneth Y. Yun and Julio Arceo of the University of California, San Diego for providing us with the details of the "diffeq" benchmark.

References

[1] T. Amon and G. Borriello. An approach to symbolic timing verification. In *Proceedings of the 29th DAC*, June 1992.

[2] T. M. Burks and K. A. Sakallah. Min-max linear programming and the timing analysis of digital circuits. In *Proceedings of the ICCAD*, Nov. 1993.

[3] S. M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits*. PhD thesis, California Institute of Technology, 1991.

[4] H. Hulgaard and S. M. Burns. Bounded delay timing analysis of a class of CSP programs with choice. In *Proceedings of the 1st ASYNC*, Nov. 1994.

[5] H. Hulgaard, S. M. Burns, T. Amon, and G. Borriello. An algorithm for exact bounds on time separation of events in concurrent systems. Technical report, Department of Computer Science and Engineering, University of Washington, Feb. 1994. Tech. Report No. UW-CSE-94-02-02.

[6] L. Lavagno and A. Sangiovanni-Vincentelli. Linear programming for optimum hazard elimination in asynchronous circuits. *Journal of VLSI Signal Processing*, 7(1):137–60, Feb. 1994.

[7] K. L. McMillan and D. L. Dill. Algorithms for Interface Timing Verification. In *Proceedings of the ICCD*, Oct. 1992.

[8] C. J. Myers and T. H.-Y. Meng. Synthesis of timed asynchronous circuits. *IEEE Transactions on VLSI Systems*, 1(2):106–119, June 1993.

[9] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun. checkTc and minTc: Timing verification and optimal clocking of digital circuits. In *Proceedings of the ICCAD*, Nov. 1990.

[10] P. Vanbekbergen, G. Goossens, and H. D. Man. Specification and analysis of timing constraints in signal transition graphs. In *Proceedings of the European DAC*, Mar. 1992.

[11] K. Y. Yun, P. A. Beerel, V. Vakilotojar, A. E. Dooply, and J. Arceo. The design and verification of a high-performance low-control-overhead asynchronous differential equation solver. In *Proceedings of the 3rd ASYNC*, April 1997.