Symbolic Techniques for Performance Analysis of Timed Systems based on Average Time Separation of Events[†]

Aiguo Xie and Peter A. Beerel
University of Southern California
Los Angeles, CA 90089

Abstract

Symbolic techniques using BDDs [1] *and ADDs* [2] *are applied to the performance analysis of (asynchronous) timed systems. We model the system as a set of probabilistic finite state machines which is analyzed as a discrete time Markov chain. The stationary probability of all reachable states is obtained iteratively using ADDs. Average time separation of events is symbolically calculated to determine various performance metrics. Application to a FIFO and a differential equation solver chip demonstrates the feasibility of the technique.*

1. Introduction

A typical objective of (asynchronous) timed systems is to achieve higher average-case performance than the worst-case performance of any comparable synchronous system. Examples of such systems include the Intel AILD (asynchronous instruction length decoder) design, an asynchronous differential equation solver ASIC [17], and various pausible clocking interfaces [18]. To better design these systems, we need performance analysis tools that can quickly analyze long-run system behavior at various levels of the design hierarchy. Such tools can identify critical paths early in the design cycle, guide architectural choices such as pipeline depth, and estimate control circuit overhead. Provided with back-annotated parameters from later steps of the design, performance verification may also be carried out.

These systems contain components that have non-fixed delays due to random environment requests, data-dependent delays in functional units, and random arbitration delays. Such systems have previously been analyzed using both simulation-based and analytical approaches. Simulation-based approaches typically deal with detailed system models, however, can lead to inconclusive results because they may identify different bottlenecks when run on different data sets. To gain higher confidence more sophisticated simulation approaches have been proposed. These approaches, however, require long simulation warm-up periods [28] to get the system into steady-state before gathering statistical data. Analytical approaches, on the other hand, provide conclusive results for a given model of the system. These approaches can use powerful abstraction techniques, such as probabilistic modeling, to simplify the analysis of components with non-fixed delays. This makes analytical techniques attractive for analysis at high levels of the design hierarchy, e.g., at the register-transfer-level and higher. The principal danger of using analytical techniques, however, is that the model used may not be very accurate. The accuracy is very application dependent, but can usually be improved at the expense of longer run-times.

One common analytical model for performance analysis is generalized timed Petri-nets (GTPN) [23] that allow transitions to be annotated with conditional probabilities. Kudva et al. [12] demonstrate how asynchronous circuits can be modeled with GTPNs. Using explicit-state-based reachability analysis techniques, a GTPN model can be transformed into an equivalent Markov chain [24]. The Markov chain is then analyzed to obtain the steady-state probability distribution. This analysis, unfortunately, requires solving a system of linear equations whose size can grow *exponentially* with the size of the system. Moreover, Kudva et al. only studied the stationary probability of individual states, and in particular, they did not relate this to more typical performance metrics.

This paper proposes a new approach which addresses these issues and limitations as follows:

- We propose using symbolic techniques to facilitate the analysis of much larger systems than otherwise possible using explicit-state-based techniques. We adopt a discrete-time model of the timed system specified by a set of non-deterministic FSMs [19] and employ implicit techniques based on ordered binary decision diagrams (BDDs) [1] to compute the reachable state space. In addition, we

[†] This work is funded in part by a NSF CAREER Award MIP-9502386, a gift from the Intel Corporation, and a research seed grant from the James H. Zumberge Faculty Research and Innovation Fund at USC.

assume each component having non-fixed delay is given a discrete probability distribution over the range of possible quantized delay values. The source of this probability distribution may vary. For example, in analyzing microprocessors the distribution might be obtained by simulating typical programs. We use Algebraic Decision Diagrams (ADDs) [2] to efficiently represent the probabilistic state transition relation which defines an underlying Markov chain of the system. Using an ADD-based iterative technique, we compute the stationary probabilities of all reachable states.

- We use a notion of average time separation of events to provide a uniform framework in which many useful performance metrics (such as latency, idle period, and throughput) can be expressed. This notion extends the concept of maximum time separation of events [10,11] to stochastic models. We describe how average time separation of events can be symbolically determined from stationary probabilities. Our techniques are currently limited to *irreducible* Markov chains which encompass many timed systems.
- We describe a preliminary tool that implements the above approach. The tool is based on the SMV [4] and CUDD [2] packages. We demonstrate its effectiveness on a FIFO and a differential equation solver ASIC both having over 15,000 discrete states.

The remainder of this paper is organized as follows. Section 2 briefly reviews basic operations on BDDs and ADDs. Section 3 describes our system model and its analysis based on Markov chain theory. Section 4 describes iterative techniques for Markov chain analysis using BDDs and ADDs. Section 5 introduces our notion of average-time separation of events, presents a closed-form formula for its computation, and illustrates its role in finding many performance metrics. Section 6 compares this approach to related research in performance analysis of asynchronous systems. A description of our tool implementation is given in section 7, followed by our case studies in section 8. Section 9 concludes this paper and outlines our future work.

2. Binary and algebraic decision diagrams

Ordered binary decision diagrams (BDDs) provide a canonical representation for boolean functions. They have been extensively used in a variety of VLSI CAD applications. Given BDDs for functions f and g, efficient algorithms exist to perform many boolean operations such as negation $(\bar{f})$ and disjunction $(f \vee g)$. Another useful operation $\exists_v f$, existential quantification, can be efficiently computed using two BDD restrictions and one disjunction, i.e., $\exists_v f = f|_{v \leftarrow 0} \vee f|_{v \leftarrow 1}$.

Algebraic decision diagrams [2] are BDDs except that their terminal nodes may assume constants from any arbitrary algebraic domain $\boldsymbol{D}$ rather than be restricted to boolean constants as in BDDs. Essentially, an ADD with n variables is a function $f : \{0,1\}^n \rightarrow \boldsymbol{D}$. In the sequel, we assume the domain $\boldsymbol{D}$ is the set of all real numbers $\Re$.

A 2^n by 2^m matrix can be viewed as a boolean function M: $\{0,1\}^{n+m} \rightarrow \Re$. It can be represented by an ADD with $n+m$ variables. Operations on matrices such as summation and multiplication can be efficiently carried out by performing operations on the corresponding ADDs. For example, let $\boldsymbol{A}(X,Z)$ and $\boldsymbol{B}(Z,Y)$ be two matrices in their ADD forms. Then, $\boldsymbol{A}(X,Z)\boldsymbol{B}(Z,Y) = \exists^+_Z \boldsymbol{A}(X,Z) \bullet \boldsymbol{B}(Z,Y)$, where $\exists^+$ represents ADD arithmetic abstraction and $\bullet$ stands for *ADD times*, an operation similar to BDD *AND* except that values in corresponding leaf nodes are multiplied [2]. In case the numbers of rows and columns are not powers of two, a matrix can be padded with zeros. Note that a vector is either a one-row or one-column matrix. In practice, since isomorphic subdiagrams can be shared, ADDs are often much more efficient than other data structures for both representation and manipulation of matrices.

3. Markov chain theory of timed systems

This section describes the discrete time model of timed systems and its analysis based on markov chains.

3.1. Discrete-time models and their transition relation

The system is modeled using a set of current state variable V and next state variables V' that either represent actual signals in the design or help model the passage of time (i.e., *timers*). Each current state variable v has a counterpart next state variable v' and their range is denoted by $\tilde{V}$. A state s is a vector of the values of all state variables. Thus, the state space Q is $\tilde{V}_1 \times \tilde{V}_2 \times \ldots \times \tilde{V}_n$. $s(v)$ is the value of state variable v in state s.

The transition relation $T \subseteq Q \times Q$ consists of state transitions $(s, s') \in Q \times Q$ each modeling the evolution of the system from a state s to a state s' after *one* time step. The transition relation is derived from the set of *partial transition relations* $\{T_v \subseteq Q \times Q,\ v \in V\}$, where T_v constrains the behavior of variable v' in terms of all other state variables. T is obtained from all T_v's using *synchronous* composition semantics in which state variables change state simultaneously. Consequently, the transition relation T is the intersection of all the partial transition relations, i.e., $T = \bigcap_{v \in V} T_v$.

For the purpose of this paper, T_v is specified by a case statement which we formalize as an *ordered* set G_v of conditional assignments. Each conditional assignment *(g:o)* in G_v consists of a condition g and an associated assignment o. The condition g is a boolean predicate of all state variables excluding v', a domain we refer to as $Q \times Q_v^-$. At least one condition in the case statement must be true for every element of $Q \times Q_v^-$ (and thus for every transition (s,s')). The first true condition and its associated assignment are said to be *active.* An assignment is a set of functions from $Q \times Q_v^-$ to $\tilde{V}$. Thus, for every element in $Q \times Q_v^-$ the active assignment defines a set of values $\tilde{V}_a \subseteq \tilde{V}$. Then, T_v is the set of state transitions (s,s') such that $s'(v) \in \tilde{V}_a$.

Example 1. Figure 1 shows a simple FIFO with n stages. Each stage consists of a C-element and a delay line. All delays are quantized into time steps. The C-elements have unit delay (i.e., one time step) and the delay lines have integer delay in the range [d, D]. The left environment has an integer delay in the range $[1, +\infty)$, while the right environment has a fixed integer delay. Each of the delay lines, delay$_i$, is modeled by a timer w_i with delay range [0, D]. To model the range of delays, w_i is non-deterministically set to any integer value in [d, D] each time the input to the corresponding delay line changes, and is decremented by one each subsequent state transition. The output of the delay line is asserted when w_i reaches 0.

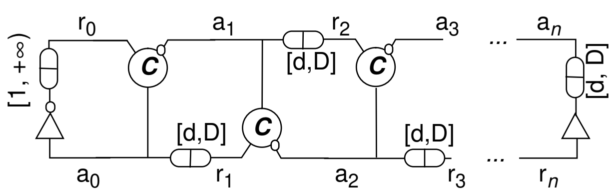

Figure 1: An *n*-stage FIFO.

The partial transition relations that model the timed behavior of the FIFO are specified using case statements. Each stage is modeled as follows:

T_{a_i} = **case**
  r_i != a_{i+1} : $\{r_i\}$;
  1 : $\{a_i\}$ for $i = 0, 2, ..., n\text{-}1$
T_{r_i} = **case**
  $w_i' = 0$ & w_i != 0 : $\{a_i\}$;
  1 : $\{r_i\}$ for $i = 1, 2, ..., n$
T_{w_i} = **case**
  a_i' != $a_{i\text{-}1}$: {d, ..., D};
  $w_i > 0$: $\{(w_i - 1)\}$;
  1 : $\{w_i\}$ for $i = 1, 2, ..., n$.

The left and right environments are modeled similarly.

3.2. Probabilistic state transition relation

A *probabilistic transition relation P* maps each state transition $t \in Q \times Q$ onto a probability such that the sum of the probabilities of the state transitions from any state is 1, i.e.,

$$\sum_{s' \in Q} P(s, s') = 1 \quad \text{for any } s \in Q \qquad (1)$$

To obtain this model we augment the conditional assignments of each variable v. We annotate the elements of each assignment of v with probabilities that add to 1. For each state transition (s,s'), this defines a function $prob : \tilde{V} \rightarrow [0, 1]$ which maps each value in $\tilde{V}_a$ to the annotated probability for that value given in the active assignment. Values of v not in $\tilde{V}_a$ are mapped to 0 probability by default. We then define a partial probabilistic transition relation $P_v : Q \times Q \rightarrow [0, 1]$ for each variable v such that $P_v(s, s') = prob(s'(v))$. The probabilistic transition relation $P : Q \times Q \rightarrow [0, 1]$ is as follows

$$P(s, s') = \prod_{v \in V} P_v(s, s') \quad \text{for all } (s, s') \in Q \times Q \qquad (2)$$

Equation 2 suggests that the probability of each state transition is defined by a set of independent choices of assignments whose probabilities multiply. This, however, is not guaranteed. Consider the example below:

T_a = **case** T_b = **case**
  b' : {1}; a' : {1};
  1 : {0} 1 : {0}

These equations define two state transitions from state (00); (00)→(00) and (00)→(11). Evaluating Equation 2, each state transition has probability 1 and their sum equals 2, a violation of Equation 1. The violation occurs because our probability assignment did not capture the non-determinism hidden in this model. Intuitively, this is because, for each partial state transition, the choice made is between two guards, not between two value assignments in one guard.

Based on this observation we can identify a class of models for which Equation 1 holds. First, we define a dependency graph in which there is a node for each current variable and a directed edge from y to x if x' is in the support set of G_y. If this graph is cycle free, we conjecture that Equation 1 is satisfied. We are currently formalizing a proof for this claim.

In addition, we believe that by adding extra state variables models that do not satisfy this property can be transformed into satisfying models without changing the performance metrics of the system .

Example 2. Consider the FIFO in Example 1 in which we assign the probability distribution for the delays of the delay lines as follows: probability(delay$_i$= 1 unit) = 0.2, probability(delay$_i$ = 2 units) = 0.6 and probability(delay$_i$ = 3 units) = 0.2. The corresponding partial transition relation for w_i is annotated as follows:

P_{w_i} = **case**
  a'_{i-1} != $a_{i\text{-}1}$: {1’0.2, 2’0.6, 3’0.2};
  $w_i > 0$: $\{w_i - 1\}$;
  1 : $\{w_i\}$

The conditional assignments that have only one element in their corresponding assignment are deterministic and thus the element is, by default, assigned probability 1. Similarly, we specify the partial probabilistic transition relation of the timer variable in the delay line of the left environment as:

P_{w_0} = **case**
  a_0' != a_0 : {1};
  $w_0 = 1$: {1'p, 0'q};
  1 : {W_0} where $p + q = 1$.

This models the delay of the left environment as having a geometric distribution with parameter p.

3.3. Markov chains

Our probabilistic transition relation P defines the one-step transition probability of a *discrete time-homogeneous Markov chain* (DTMC) $\{X_n : n \geq 0\}$ with (finite) state space Q.

A DTMC is a stochastic process for which the probability of a state transition depends only on the current state and not on the previous states visited or the current time. That is, for a DTMC, $P\{X_{n+1} = s' | X_n = s, X_{n-1}, ..., X_0\}$ equals $P\{X_{n+1} = s' | X_n = s\}$ which equals the constant $P_{ss'}$ for all $n \geq 0$ and $s, s' \in Q$.

The probability for the DTMC to transit from s to s' in n steps is referred to as $P_{ss'}^{(n)}$. A state s is *recurrent* (occurs infinitely often) if $\sum_{n=1}^{\infty} P_{ss}^{(n)} = \infty$, otherwise, it is *transient*. Two state s and s' communicate if there are some constants n, $m > 0$ such that $P_{ss'}^{(n)} > 0$ and $P_{s's}^{(m)} > 0$. A *communicating class* is a maximal set of states where every state communicates with every other state. A communicating class is a *recurrent class* if all its states are recurrent. An DTMC (with finite state space) is called *irreducible* if there is only one recurrent class, otherwise, it is called *reducible*. If starting from state s the DTMC revisits state s only at time instants that are multiples of some integer d $(d > 1)$, then state s is *periodic*, otherwise, it is *aperiodic*. States in a communicating class are either all periodic or all aperiodic.

The stationary probability of an irreducible aperiodic DTMC can be found by solving the following linear system:

$$\pi P = \pi \quad \text{and} \quad \pi 1^T = 1 \tag{3}$$

where, $\pi_{s'} = \lim_{n \to \infty} p_{ss'}^{(n)}$ is the stationary probability of state s and 1^T is an all-one vector of proper size. If the DTMC is periodic, the stationary probabilities are defined with the more general formula $\pi_{s'} = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} p_{ss'}^{(m)}$.

If the Markov chain is reducible, Equation (3) does not apply because the stationary probabilities depend not only on P but also the distribution of the initial states. This is likely to happen when many states in Q are unreachable from the initial state. Fortunately, we can restrict P to reachable states, forming a new Markov chain, which is more likely to be irreducible.

4. Symbolic analysis of state probabilities

The principal difficulty with Markov chain analysis is the computational challenge of dealing with the large state spaces of real systems. This section demonstrates how we address this problem by combining iterative techniques to find the stationary probabilities with symbolic representations using BDDs and ADDs.

4.1. Symbolic reachability analysis

The first step in our approach is to restrict P to the reachable state space and determine if the resulting Markov chain is irreducible.

We propose to use a well-known symbolic approach for reachability analysis [4], [6] to mitigate the state explosion problem. The approach is iterative and works as follows. Let Q_0 be the set of initial states and Q_k be all the states reachable from Q_0 in at most k steps from Q_0. Q_k can be computed iteratively as follows: $Q_{k+1} = Q_k \cup \{s' | \exists s, s \in Q_k - Q_{k-1}, (s, s') \in T\}$, where $Q_{-1} = \emptyset$. The iteration continues until the *frontier set* Q_k - Q_{k-1} is empty.

This fixed point is computed using BDDs. Let $\boldsymbol{T}_v(V, V')$ be the BDD corresponding to T_v. Then, the global (or monolithic) transition relation of a timed system $\boldsymbol{T}(V,V')$ is a *conjunction* of all these partial BDDs. Let $\boldsymbol{Q}_k(V)$ be the BDD corresponding to Q_k. Then, the set of reachable states is the fixed point of the following computation $\boldsymbol{Q}_{k+1}(V') = \boldsymbol{Q}_k(V') \vee \exists_{v \in V} \boldsymbol{Q}_k(V) \wedge \neg \boldsymbol{Q}_{k-1}(V) \wedge \boldsymbol{T}(V, V')$.

Once the reachable states are found, restricting P is done using BDD *AND*. Then, to determine if the restricted P defines an irreducible Markov chain, we propose a conservative approach. Assume for simplicity there exists one initial state and all annotated probabilities are non-zero. Then, we can use symbolic model checking techniques to verify that all reachable states can reach the initial state. If there are multiple recurrent classes, then the verification test will not pass (i.e., the method is safe). If there is only one recurrent class then we analyze two cases. First, if the initial state is in the recurrent class, the verification test will pass. Second, if the there is the initial state is not in the recurrent class, the verification test will fail. This second case represents a false negative which may be removed by choosing a different initial state. Extensions to multiple initial states are straight-forward. Currently, our method assumes this verification test passes.

4.2. The power method

Calculating the stationary probabilities π is equivalent to an *eigenvector* problem. It is easy to see that π is the normalized eigenvector of the transition matrix P corresponding to eigenvalue 1. Moreover, there is a unique maximum

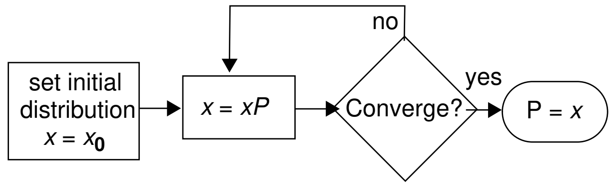

Figure 2: Diagram of the power method.

eigenvalue of P that equals 1.

Direct methods (such as Gaussian elimination) are often numerically unstable due to the subtractions involved and finite computation precision. Although they may be made numerically stable through partial or complete pivoting, they are usually unable to take advantage of the sparseness of the transition matrix. In comparison, iterative methods are numerically stable and can take advantage of the sparseness of the transition matrix.

We use the power method, illustrated in Figure 2, which is perhaps the easiest iterative method to implement. We let $\boldsymbol{x}_0$ be the normalized initial probability vector, i.e, $x_0 1^T = 1$. It follows that if we let $x_n = x_0 P^n$ that $x_n 1^T = 1$. It is easy to show that the sequence $\{x_n \mid n > 0\}$ converges to the eigenvector of P corresponding to eigenvalue 1. Moreover, let P be rearranged and decomposed as $B^{-1}DB$, where $D = diag(\lambda_1,...,\lambda_n)$ and $1 = \lambda_1 > \lambda_2 \geq \ldots \geq \lambda_n > 0$. It turns out that $\boldsymbol{x}_n\text{-}\pi = (\boldsymbol{x}_0\text{-}\pi)B^{-1}D^n B$. Thus, the power method converges at a rate of λ_2. Also, if $(\boldsymbol{x}_n\text{-}\pi)1^T$ is chosen for convergence criterion, the number of iterations required is proportional to $\log_{1/\lambda_2}\|x_0 - \pi\|$. If λ_2 is close to 1, the convergence could be slow.

One disadvantage of this iterative method for computing the eigenvector is that its convergence rate heavily depends on its eigenvalue distribution. Preconditioning techniques [26] transform the original problem by multiplying the corresponding matrix by a series of inversions of some triangular matrices so that the resulting matrix has more favorable eigenvalue distribution. In addition, rather than generating a sequence of vectors as in power method, the subspace iteration technique [25] computes a sequence of sets of n vectors which converges to the eigenvectors corresponding to the n largest eigenvalues of P. This leads to a faster convergence rate of λ_{n+1}/λ_1. Advantages of employing these sophisticated approaches are currently under investigation.

4.3. Symbolic calculation of the power method

Let $\boldsymbol{P}_v(V, V')$ represent the ADD for P_v. Then, the probabilistic transition relation can be represented by an ADD $\boldsymbol{P}(V, V')$ which can be constructed using ADD times as follows:

$$\boldsymbol{P}(V, V') = \bullet_{v \in V} \boldsymbol{P}_v(V, V') \tag{4}$$

Figure 3 illustrates the construction of the probabilistic transition relation ADD from the ADDs of the partial probabilistic transition relations.

The symbolic calculation of the power method, is basically the same as shown in Figure 2, except that matrix multiplication is replaced with ADD times. To accelerate the convergence, however, we multiply x in each iteration with P^{iter}, a higher order of P. The power number *iter* is restricted so that the size of P^{iter} does not exceed a predefined limit. This optimization yields faster convergence at the expense of more computation in each iteration step, resulting in lower run-times. Moreover, we also reduce the size of the ADD $\boldsymbol{P}(V, V')$ by *minimizing* it with respect to all unreachable states. This is done with standard algorithms and significantly reduces the overall computation run-time.

5. Average time separations of events and performance metrics

In this section, we relate performance metrics to average time separations of events which can be derived from the notion of *sojourn times* [27] of a Markov chain.

5.1. Sojourn times

Consider a time-homogeneous irreducible Markov chain $M = \{X_n : n \geq 0\}$ with finite state space Q. Suppose A is a proper subset of Q and $\bar{A}$ is the complementary subset, i.e., $Q - A$. Furthermore, assume that both A and $\bar{A}$ have at least one state in the (only) recurrent class of M. Clearly, both A and $\bar{A}$ will be visited infinitely often. A and $\bar{A}$ decompose the transition probability matrix P into four submatrices and the stationary probability vector π into two subvectors, i.e.,

$$P = \begin{bmatrix} P_{AA} & P_{A\bar{A}} \\ P_{\bar{A}A} & P_{\bar{A}\bar{A}} \end{bmatrix} \text{ and } \pi = (\pi_A, \pi_{\bar{A}}). \tag{5}$$

The time spent by M in subset A during its n^{th} visit to A is a random variable and called the n^{th} *sojourn time* of M in A. More formally, a sojourn of M in A is a sequence $X_l, ..., X_{l+m}$ where $n > 0$, $X_l, ..., X_{l+m-1} \in A$, $X_{l+m} \in \bar{A}$ and if $l > 0$ then $X_{l-1} \in \bar{A}$. This sojourn starts at time l, ends at time $l+m$, and lasts for time m. Let $N_{A,k}$ denote the k^{th} sojourn time of M in A. Rubino et al. [27] show the asymptotic behavior of $N_{A,k}$, denoted by J_A, as follows,

$$J_A = \lim_{k \to \infty} \frac{1}{k} \sum_{l=1}^{k} N_{A,l} = \frac{\pi_A 1_A^T}{\pi_A (I - P_{AA}) 1_A^T}. \tag{6}$$

$\pi_A 1_A^T$ is the stationary probability that the chain is in set

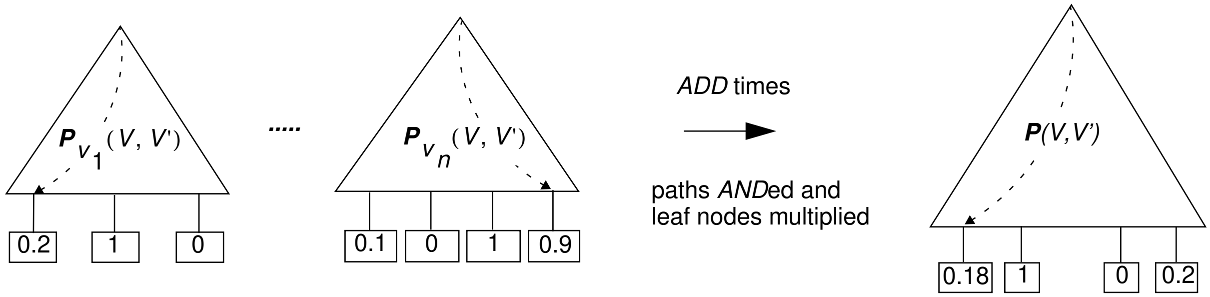

Figure 3: Illustration of the generation of the ADD for the probabilistic transition relation $P_v(V,V')$.

A, and $\pi_A(I - P_{AA})1_A^T$ is essentially the probability of the chain leaving A. In this sense, the above equation states that the sojourn time of M in A is the reciprocal of the probability of the chain leaving A given that it is in A.

5.2. Average cycle time and throughput

We define an *event* as a transition of a state variable. For simplicity, this section will discuss only events of boolean state variables. Extensions for enumerate state variables are straightforward.

Let $v(n)$ be the value of variable v at time n. We say variable v makes a transition at time n if $v(n) \neq v(n-1)$. Let $Q^v = \{s \mid s(v) = 1, s \in Q\}$ and $Q^{\bar{v}} = \{s \mid s(v) = 0, s \in Q\}$, where $s(v)$ denotes the value of variable v in state s. Then v^+ (v^-) is an event that corresponds to a state transition $t \in Q^{\bar{v}} \times Q^v$ ($t \in Q^v \times Q^{\bar{v}}$). Clearly, Q^v and $Q^{\bar{v}}$ partition Q.

The average cycle time of event v^+, denoted by C_{v^+}, is the asymptotic time difference between two consecutive occurrences of v^+. Let (v^+, l) be the time of the l^{th} occurrence of v^+ and $\Delta_{v^+, l} = (v^+, l+1) - (v^+, l)$ be the time difference between the l^{th} and $l+1^{\text{th}}$ occurrences of v^+. Note that both (v^+, l) and $\Delta_{v^+, l}$ are random variables. C_{v^+} is defined to be:

$$C_{v^+} = \lim_{k \to \infty} \frac{1}{k} \sum_{l=1}^{k} \Delta_{v^+, l} \tag{7}$$

Since the time difference of l^{th} occurrence of v^+ (v^-) and its succeeding occurrence of v^- (v^+) is the l^{th} sojourn time of Q^v ($Q^{\bar{v}}$), C_{v^+} can be computed using sojourn times as follows:

$$C_{v^+} = \lim_{k \to \infty} \frac{1}{k} \sum_{l=1}^{k} N_{Q^{v^+}, l} + \lim_{k \to \infty} \frac{1}{k} \sum_{l=1}^{k} N_{Q^{v^-}, l} = J_{Q^{v^+}} + J_{Q^{v^-}} \tag{8}$$

The average cycle time of v^- is defined and computed similarly. In fact, $C_{v^-} = C_{v^+}$.

For the purposes of this paper, we consider throughput of the system to be the reciprocal of the average cycle time of some corresponding *indicating* event (such as the rising edge of a completion signal from a functional unit). Consequently, if the indicating event is v^+, then the throughput is computed as $1/C_{v^+}$. This naturally extends to more general indicating events.

5.3. Time separations of two events

To determine other performance metrics, such as latency, we need to look at the average time difference between multiple events.

We define the time separations between events α and β as a sequence of time intervals, $\boldsymbol{I} = \{I_k = [a_k, b_k] : k = 1, 2, ..., N_{\alpha \to \beta}(n)\}$ up to time n, where the starting and ending times of each interval are recursively defined as follows,

$a_0 = min\{i : \alpha \in E_i\}$,

$b_k = min\{i : \beta \in E_i, i \geq a_k\}$, for all $k \geq 0$

$a_k = min\{i : \alpha \in E_i, i > b_{k-1}\}$, for all $k \geq 1$ and

$N_{\alpha \to \beta}(n) = i : b_i \leq n, a_{i+1} > n$,

where E_i is the set of events that occur at time i.

Note that the length of time interval I_k, denoted by $|I_k|$ is a random variable. In addition, I_k has zero length if $a_k = b_k$. We define the average time separation between event α and β, denoted by $\tau_{\alpha \to \beta}$, as the asymptotic behavior of $|I_k|$:

$$\tau_{\alpha \to \beta} = \lim_{n \to \infty} \frac{1}{N_{\alpha \to \beta}(n)} \sum_{k=1}^{N_{\alpha \to \beta}(n)} |I_k| \tag{9}$$

We compute $\tau_{\alpha \to \beta}$ using sojourn times. The idea is to partition the time intervals into two sets: intervals that have non-zero length, $\boldsymbol{I}_1 = \{I_k \mid |I_k| > 0, I_k \in \boldsymbol{I}\}$, and those that have zero-length, $\boldsymbol{I}_2 = \{I_k \mid |I_k| = 0, I_k \in \boldsymbol{I}\}$. This can be accomplished by introducing three extra state variables into the model, called x, y and z. The behavior of x is defined such that sojourns of subspace x=1 are in one-to-one correspondence with time intervals in $\boldsymbol{I}_1$. Similarly, the behavior of z is defined such that the sojourns of subspace z=1 are in one-to-one correspondence with time intervals in $\boldsymbol{I}_2$. Finally, the behavior of y is defined such that the total number of time intervals is the number of sojourns of subspace y=1. Figure 4 gives the definitions of these three vari-

```
Tx := case
   /* if α but not β happen */
   u != u' & u' = 1 & v = v'    : 1;
   /* if β happens */
   v != v' & v' = 1             : 0;
   1                            : x
Tz := case
   /* if α and β both happen */
   u != u' & u' = 1 & v != v' & v' = 1 : 1;
   1                                   : 0
Ty := case
   /* if x or z rises */
   x = 0 & x' = 1 | z = 0 & z' = 1   : 1;
   1                                 : 0
```

Figure 4: Definition of the auxiliary variables.

ables. It is important to note that the new model with these three auxiliary variables added is still Markovian because the variables can not introduce cycles in the dependency graph.

To illustrate how the extra state variables track the time separation intervals, assume that both α and β are positive transitions, $\alpha = u^+$ and $\beta = v^+$. Figure 5(a) illustrates a situation where the intervals have non-zero length while Figure 5(b) shows a zero-length interval where α and β happen simultaneously. Sojourn times of x=1 equal the corresponding *non-zero* time intervals between u^+ and v^+. Sojourns of subspace y=1 or z=1 both have unit length.

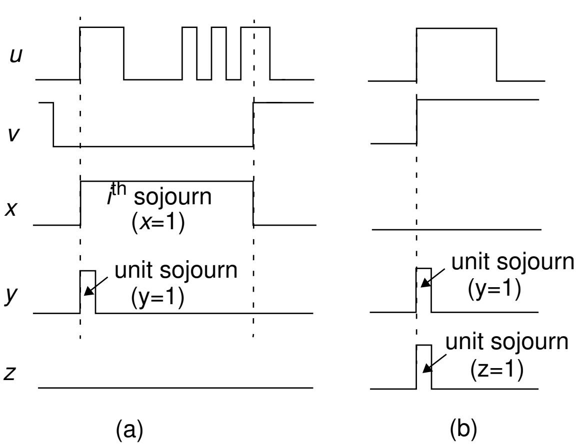

Figure 5: Time separations between u^+ and v^+ and the sojourns of the auxiliary variables when (a) u^+ and v^+ do not happen at the same time, (b) u^+ and v^+ happen at the same time.

Further analysis can show that the average time separation between α and β is related to the sojourn time of $x = 1$ and the stationary probabilities $\pi_{y=1}$ and $\pi_{z=1}$ by the following equation:

$$\tau_{\alpha \to \beta} = P\{|I| > 0 | I \in \boldsymbol{I}\} J_{x=1} \tag{10}$$

where,

$$P\{|I| > 0 | I \in I\} = \frac{\pi_{y=1} 1^T_{y=1} - \pi_{z=1} 1^T_{z=1}}{\pi_{y=1} 1^T_{z=1}} \tag{11}$$

and $J_{x=1}$ is the average sojourn time of subspace (x = 1). Also, $\pi_{y=1}$ and $\pi_{z=1}$ are the vectors representing the stationary probabilities of the states in subspaces (y=1) and (z=1), respectively, and $1^T_{y=1}$ and $1^T_{z=1}$ are all-one vectors of proper length.

6. Comparison to related work

One can see two categories of previous approaches to the performance analysis of asynchronous systems. The first category focuses on analyzing models in which delays are either fixed or bounded. The latter deals with models in which different values of delays are given different probabilities.

The first category includes the ground-breaking work by Burns [7]. He models systems as constraint-graphs where nodes represent signal transitions (events) and arcs represent causality constraints which are annotated with fixed delays. He analyzes these graphs using linear programming to find useful performance metrics. Lee [8] extends Burns' work by generalizing the constraint-graphs to include events with OR-casualty. Hulgaard and Burns [10] extend the analysis to constraint-graphs with interval delays on the arcs. Later in [11], the authors apply algebraic techniques to find tight bounds of event time separations of a class of timed petri nets that can model conditional (choice) behavior. None of these techniques, however, can model the probabilistic delays used in our model. Our approach can analyze any system of communicating FSMs which can express all variants of timed petri nets that we are aware of.

Also belonging to this research category is Tofts's work [14] where he applies timed process algebra to obtain the extreme case latencies and throughput of *micropipelines*. Our average-case analysis can be applied to micropipelines and is more automated than Tofts's manual analysis. Interestingly, to address the state explosion problem, Tofts proposes a compositional approach which can lead to hierarchical performance analysis while we propose the application of symbolic techniques. A combination of both of these approaches may be beneficial.

Research falling into the second category typically adopts Petri nets models. Kudva et al. [12] models the systems as generalized timed Petri net and uses GTPNA ana-

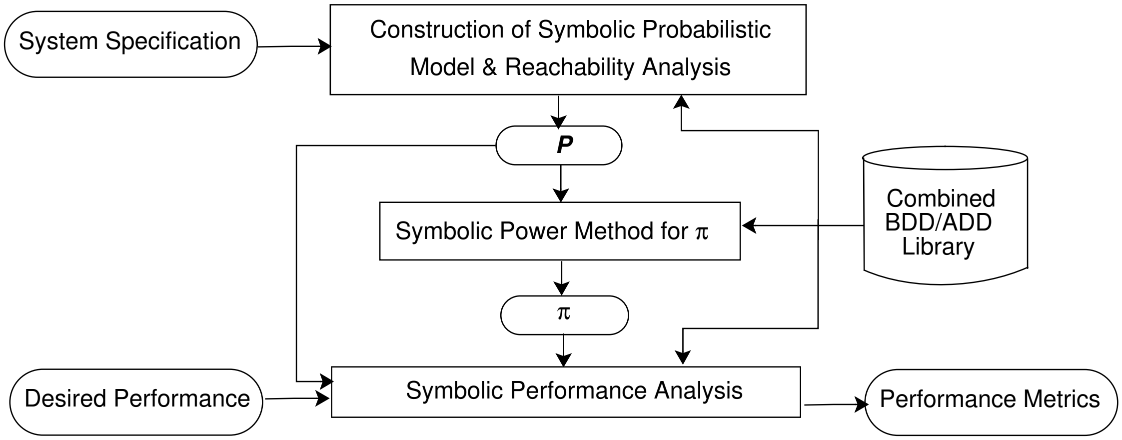

Figure 6: A block diagram of our symbolic performance analysis tool.

lyzer by Holliday and Vernon [24] to convert the model into a discrete-time Markov chain. Timed Petri nets are powerful in modeling non-determinism in asynchronous systems such as data-dependent delays and arbitration. However, the approach suffers from the state explosion problem due to the explicit reachability analysis and computation of the stead state probabilities built in the GTPNA tool and thus is limited to small systems. We should note, however, that the explicit reachability analysis algorithm contained in GTPNA tool aggregates those states that transit from one to another with probability 1, which can mitigate the state explosion problem in some cases. Applying this optimization in our symbolic approach is currently under investigation. In addition, Kudva et al. model delay ranges by their average value which can lead to inaccurate results. Finally, their performance metrics are derived from a notion of *long-run times* of individual states by Zuberek [13]. However, we believe that useful performance metrics are more directly related to our notion of average time separations of events.

7. Implementation

We implemented our preliminary symbolic performance analysis tool in C. Figure 6 shows its block diagram. System specification follows SMV syntax except that the entries of non-deterministic next-state assignments are annotated with probabilities. Parts of *SMV* data structures and codes were modified to generate the probabilistic transition relation ADDs and to compute reachable states. Functions in the *CUDD* library [2] are used to efficiently perform the needed matrix multiplications and vector summations.

8. Examples and discussion

To show the effectiveness of our techniques, we analyze two examples: an asynchronous pipeline and an asynchronous differential equation solver ASIC.

8.1. The FIFO

The first example we consider is the FIFO shown in example 1 and 2. In particular, we examine its *average response time* and *throughput*. The average response time is defined as the average time difference between a request (a transition of r_0) issued by the right environment and the successive acknowledgment (a transition of a_0) issued by the first stage. The throughput is defined to be the reciprocal of the average time difference between successive pipeline request outs. In this example, it can be regarded as the reciprocal of the average cycle time of the request issued by the last stage in the FIFO.

Performance of FIFOs have been studied extensively [14,15,16]. Using a process graph model similar to that used in [10], Ebergen and Berks give closed-form formula for the worst-case and amortized response time of general FIFO structures [16]. The amortized response time is an *upper bound* of the average response time under *worst-case* delay distributions. The important delay parameters in their model are δ and Δ , lower and upper bounds of the delay through any stage and D, an upper bound of the delay through the right environment. They show that the amortized response time is $D + (\Delta - \delta)$ assuming that $\Delta \le D$ and the delay through the left environment is no larger than δ. On the other hand, modeling of the FIFO in our approach involves the specification of (discrete) delay distributions for all the delay lines. The average response time given by our method is *exact* under a given distribution.

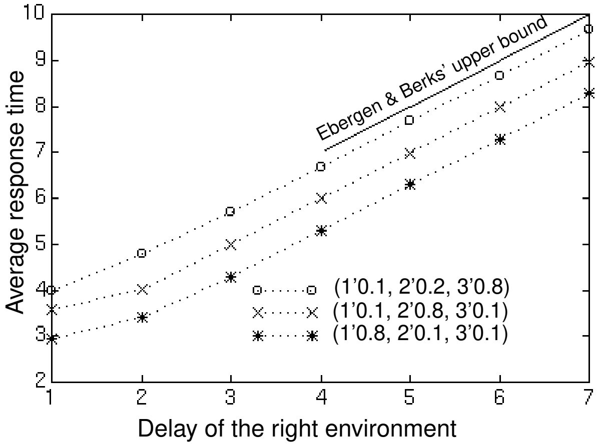

Figure 7: Average response time v.s. delay of the right environment and the distribution of the delay in delay lines of internal stages.

Assume there are 6 identical stages in the FIFO. To load the FIFO as fast as possible, the left environment issues a request one time step after it receives an acknowledge from the first stage. The delay through the right environment is fixed. To show how the average response time and throughput are affected by the right environment, we vary the delay of the right environment from 1 to 7 time units. Figure 8 shows the average response time as a function of the right environment delay. Each of the curves in the figure corresponds to a particular distribution of the internal stage delays.

For all three curves, the delay interval of the internal stages is [1,4], i.e, $\delta = 1$ and $\Delta = 4$, and the corresponding delay line has the labeled probability distribution. The solid line shows the amortized response time given by Ebergen and Berks when the delay through the right environment is greater than 3 units. The trend is that the average response time is in proportion to the delay of the right environment, which agrees with Ebergen and Berks's analysis.

However, the average response time is also sensitive to the distribution of delays. If each stage is likely to be slow (have delay close to its upper bound) the average response time is close to the Ebergen and Berks' analysis. If the internal stages are more likely to be fast (having a delay close to its lower bound), then average response time is significantly lower than Ebergen and Berks' bound. The curves in Figure 7 identify over a 30% variance depending on the delay distribution.

Figure 8 shows the throughput of the FIFO in terms of the average cycle time of the requests issued by the last stage. Interestingly, the average cycle time of request from any stage is approximately the average response time plus the delay of the left environment.

We terminate our iterative power method when the desired performance metrics have converged to within a given accuracy. Interestingly, the performance metrics converge much slower than the stationary probability vector and thus we need significantly more iterations to obtain the same precision in the metrics than is needed for the stationary probability vector. Using a convergence criterion of 1%, the run-times were less than 1 hour for each delay distribution tested on a SPARC 20 with 64 Megabytes of memory. The models have up to 28,000 reachable states.

8.2. The asynchronous differential equation solver

The asynchronous differential equation solver ASIC implements the behavioral code described below [17].

```
DIFFEQ Procedure
read inputs
while (x < a) loop
    x1 := x + dx;
    u1 := u - 3 * x * u * dx - 3 * y * dx;
    y1 := y + u * dx;
    x   := x1 ;  u := u1 ; y := y1;
end loop
output results
```

In theory, we would first analyze the performance of a high-level behavioral description of the design and use the results to guide its architectural design. However, due to time pressures, we decided to analyze a more detailed RTL model of an existing implementation which was used for verification [19]. This RTL model describes the architecture illustrated in Figure 9 and models the behavior of all controllers, C1, C2, C3, and C4. The model uses non-determinism to abstract the data-dependent delay of the functional units as well as the assignment of the *sign* signal in ALU2 which holds the result of the loop test `x < a`. This is examined in controller C2 to decide whether to initiate a

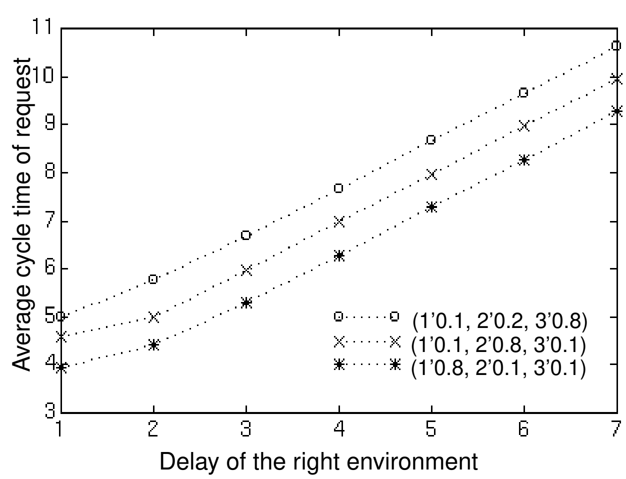

Figure 8: Analysis of FIFO throughput.

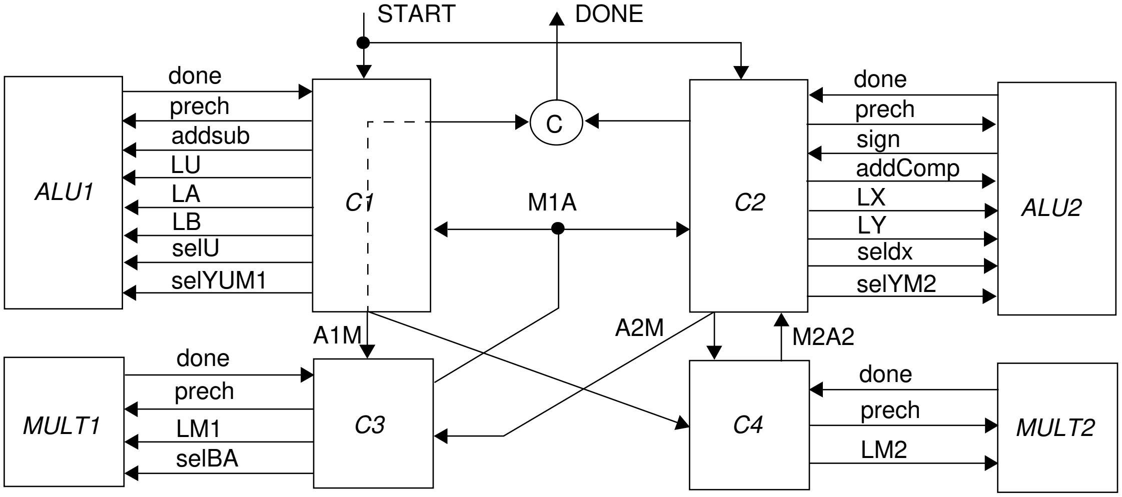

Figure 9: A block diagram of differential equation solver.

new loop iteration or to terminate the procedure. The model has a time step of 1ns, i.e., each state transition represents the evolution of the circuit after 1ns.

We focus on two performance parameters. The first is the average procedure delay from reading inputs to outputting the results. In the RTL design, this delay corresponds to the average time separation from $START^+$ to $DONE^+$. The second parameter we calculate is the average delay of one loop iteration. In the RTL design, there are two control points in which the loop is entered, one for the first iteration and the other for all subsequent iterations. To apply our notion of average time separation of events, we introduced a new indicating variable which rises when either entrance point is reached. In addition, a second indicating variable was created which rises when a loop is completed. Then, the average time separation between the rising of both indicating signals corresponds to the average loop delay. We analyzed both parameters on two versions of the design, one with estimated delays and a later one with backannotated delays. It turned out that the backannotated metrics were significantly larger, primarily because the backannotated model incorporated long wire delays that were not considered in the initial model. For both designs, we specified the datapath delay distributions with coarse back-of-the-hand estimates. The run-time results of our tool are reported in Table 1. Using a convergence criterion of 1%, the run-times were less than 1/2 hour on a SPARC 20 with 64 Megabytes of memory, demonstrating the computational feasibility of our approach. Surprisingly, the average iteration delays obtained for the backannotated design were within 8% of SPICE analysis. A SPICE or gate-level simulation for average procedural delay was not available. We might expect, however, that the accuracy of our results depends greatly on the accuracy of the probability distribution we give to the ALU sign signal. Moreover, the model does not capture the correlations between successive input data to the ALU. If desired, this correlation can be approximated in the model by the addition of extra variables at the expense of larger run-times. A more practical application of our techniques, however, is to compare different architectures in which the relative accuracy of the performance metrics is more important. Unfortunately, models of different RTL architectures of the DIFFEQ ASIC are not currently available.

9. Conclusion and Future Work

We presented an approach to the performance analysis of timed systems which uses symbolic techniques to address the state explosion problem. We introduced a notion of average time separation of events which can be symbolically derived from the set of probabilistic finite state machines that make up our system model. Useful performance metrics can be directly related to average time separations of events. We implemented a preliminary tool to demonstrate the feasibility of the approach on two examples. Our tool is currently limited to system models whose underlying Markov chain is irreducible. Fortunately, this seems to cover many of the systems that we examined.

Our future work will focus on improving the run-time efficiency of our approach. In particular, we are currently investigating the following issues:

- Aggregate strings of states during symbolic reachability analysis before beginning steady state analysis.
- Apply liberal approximations of the state space by avoiding the exploration of state space sections that are rarely visited.
- Explore sophisticated iterative methods to find the sta-

tionary probabilities.

DIFFEQ Model	Reachable States	*size*(P^{16})	CPU Time (min)	Iterations to Converge	Average iteration delay	Average Procedure delay
estimated	17,212	101,319	22.3	16	28.4ns	54.1ns
backannotated	22,072	72,241	16.5	15	42.7ns	79.6ns

Table 1: Run-time results for the performance analysis of an asynchronous DIFFEQ design.

Acknowledgments

We would like to thank the reviewers for their valuable suggestions for improving this paper. We would also like to thank V. Vakilotojar at the University of Southern California for many insightful discussions and comments related to this work.

References

[1] R. E. Bryant, "Graph-based algorithm for boolean function manipulation," *IEEE Trans. Comp.*, vol. C-35, Aug. 1986.

[2] R. I. Bahar, E. A. Frohm, G. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo and F. Somenzi, "Algebraic decision diagrams and their applications," in *Proc. IEEE Int. Conf. Computer- Aided Design*, Nov. 1993.

[3] G. D. Hachtel et al, "Symbolic algorithms to calculate steady-state probabilities of a finite state machine," in *Proc. European Conf. Design Automation*, *European Test Conf.*, *European Event in ASIC Design*, Feb-Mar. 1994, pp. 214-219.

[4] K. L. McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.

[5] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan and D. L. Dill, "Symbolic model checking for sequential circuit verification," *IEEE Trans. Computer-Aided Design*, vol.CAD-13, April 1994.

[6] K.L.McMillan, "A technique of state space search based on unfolding," *Formal Methods in System Design*, Vol. 6, No. 1, 1995, pp. 45-65.

[7] S. M. Burns, *Performance analysis and optimization of asynchronous circuits*, Ph.D. thesis, California Institute of Technology, 1991.

[8] T. K. Lee, *A general approach to performance analysis and optimization of asynchronous circuits*, Ph.D. thesis, California Institute of Technology, 1995.

[9] H. Hulgaard, T. Amon, S. M. Burns, and G. Borriello, "Timing analysis of timed graphs with bounded delays using algebraic techniques," In *Proc. IEEE Conf. Decision and Control*, 1994.

[10] H. Hulgaard, S. M. Burns, T. Amon, and G. Borriello, "An algorithm for exact bounds on time separation of events in concurrent systems," *IEEE Trans. Computers*, Vol. 44, No. 11, Nov. 1995.

[11] H. Hulgaard and S. M. Burns, "Efficient timing analysis of a class of petri nets," In *Computer Aided Verification*, 1995.

[12] P. Kudva, G. Gopalakrishnan, E. Brunvand, and V. Akella, "Performance analysis and optimization of asynchronous circuits," In *Proc. Int. Conf. Computer Design*, Oct 1994, pp. 221-225.

[13] W. Zuberek, "Timed petri nets and preliminary performance evaluation," In *7th Int. Symp. on Computer Architecture,* 1980, pp. 88-96.

[14] C. Tofts, "A compositional analysis of the performance of asynchronous micropipelines," in *1995 ACM Int. Workshop Timing Issues in the Specification and Synthesis of Digital Systems*, Nov. 1995.

[15] T. E. Williams, "Analyzing and improving the latency and throughput performance of self-timed pipelines and rings," In *Proc. Int. Symposium on Circuits and Systems*, May 1992.

[16] J. Ebergen and R. Berks, "Response time properties of some asynchronous circuits," to appear *Proc. of Int. Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 1997.

[17] K. Y. Yun, P. A. Beerel, V. Vakilotojar, A. E. Dooply, and J. Arceo, "A low-control-overhead asynchronous differential equation solver," to appear in *Proc. Int. Symposium on Advanced Research in Asynchronous Circuits and Systems,* April 1997.

[18] K. Y. Yun and R. P. Donohue, "Pausible clocking: A first step toward heterogeneous systems," in *Proc IEEE Int. Conf. Computer Design*, Oct. 1996.

[19] V. Vakilotojar and P. A. Beerel, "A methodology for the RTL verification of asynchronous and heterogeneous systems," accepted for publication in ASP-DAC'97.

[20] J. R. Burch, *Trace algebra for automatic verification of Real-time Concurrent systems*, Ph.D. thesis, Carnegie Melon University, 1992.

[21] T. G. Rokicki, *Representing and Modeling Circuits*, Ph.D. thesis, Stanford University, 1993.

[22] R. Kol, R. Ginosar and G. Samuel, "Statechart methodology for the design, validation and synthesis of large scale asynchronous systems," in *2nd Int Symp. Advanced Research in Asynchronous Circuits and Systems*, 1996.

[23] M. Holiday and M. Vernon, "A generalized timed Petri net model for performance analysis," *IEEE Trans. Software Eng.*, Vol. SE-13, Dec. 1987, pp. 1297-1310.

[24] M. Holliday and M. Vernon, "The GTPN analyzer: numerical methods and user interface," In *1986 Fall Joint Comp. Conf.*, Nov. 1986.

[25] G. W. Stewart, "Methods of simultaneous iteration for computing invariant subspaces of non-Ruminating matrices," *Numerical Mathematics*, vol. 25, 1976, pp. 123-136.

[26] A V. D. Ploeg, "Preconditioning techniques for large sparse, non-symmetric matrices with arbitrary sparsity patterns," In *Proc. IMACS Symp. Iterative Methods in Linear Algebra*, pp.173-179, 1991.

[27] G. Rubino and B. Sericola, "Sojourn times in finite markov process," *J. Appl. Prob.*, vol. 27, 1989, pp. 744-756.

[28] T. Chou and K. Roy, "Accurate power estimation of CMOS sequential circuits," IEEE Trans. VLSI Systems, Vol. 4, No. 3, Sept. 1996, pp. 369-380