Probabilistic Analysis of Large Finite State Machines

Gary D. Hachtel Enrico Macii [*] Abelardo Pardo Fabio Somenzi

University of Colorado
Dept. of Electrical and Computer Engineering
Boulder, CO 80309

Abstract

Regarding finite state machines as Markov chains facilitates the application of probabilistic methods to very large logic synthesis and formal verification problems. Recently, we have shown how symbolic algorithms based on Algebraic Decision Diagrams may be used to calculate the steady-state probabilities of finite state machines with more than 10^8 states. These algorithms treated machines with state graphs composed of a single terminal strongly connected component. In this paper we consider the most general case of systems which can be modeled as state machines with arbitrary transition structures. The proposed approach exploits structural information to decompose and simplify the state graph of the machine.

1 Introduction

Finite state machines (FSMs), or their extensions, are often employed to model real digital systems for formal verification. As the complexity of those systems increases, probabilistic approaches to design and implementation verification become of interest; for example, verification of timing properties is usually based on such kind of techniques. Beside formal hardware verification, other successful applications of probabilistic methods to finite state models can be found in the field of logic synthesis: FSM re-encoding for sequential low-power synthesis, gate-level timing analysis and verification, ATPG, and so on.
The probabilistic behavior of a FSM can be analyzed by regarding its transition structure as a Markov chain [1, 2]; in fact, it is sufficient to attach to the out-going edges of each state a label which represents the probability for the FSM to make that particular transition to obtain a finite state model that matches the definition of discrete-parameter Markov chain. Studying the behavior of a Markov chain is then related to performing the reachability analysis of a FSM.
Algorithms to analyze structurally complex Markov chains based on very sophisticated theory [3] and accurate numerical methods [4] have been used for systems with transition structures of limited size. On the other hand, FSM traversal procedures based on symbolic execution are currently available to handle very large finite state systems [5].
In [6], we have shown how, using Algebraic Decision Diagrams (ADDs) [7], the two worlds can be merged; in fact, we have proposed symbolic procedures to compute steady-state probabilities for very large FSMs. That work mainly focused on algorithms for the solution of systems of linear equations extracted from large, but structurally simple, state transition graphs; in particular, we considered Markov chains with a single terminal strongly connected component. This class of chains contains most of the examples normally encountered. However, as it will be shown later in the paper, techniques capable of dealing with arbitrary state graphs can be used to make large, structured problems tractable by decomposition.

[*]Enrico Macii is also with Politecnico di Torino, Dipartimento di Automatica e Informatica, Torino, ITALY 10129.

Therefore, in this paper, we propose an algorithm which faces the problem of computing state occupation probabilities of a FSM in its generality. Our approach relies on BDD-based structural analysis of the state graph for two important aspects: The identification of the terminal components, and the study of their periodicity. The knowledge of the structure of the sequential system being analyzed allows us to decompose and simplify it in such a way that the solution methods we have presented in [6] can be applied successfully.

2 FSMs and Markov Chains

A *Finite State Machine* , M, is represented as a 6-tuple

$$M = (\Sigma, O, S, S^0, \Delta, \Lambda),$$

where Σ is the input alphabet, O is the output alphabet, S is the finite set of states of the machine, S^0 is a set of reset (initial) states, $\Delta(s,x)$ is the next state function ($\Delta : S \times \Sigma \rightarrow S$), and $\Lambda(s,x)$ is the output function ($\Lambda : S \times \Sigma \rightarrow O$). The sets Σ, O, S, and S^0 are assumed to be non-empty. Δ and Λ are multiple-output boolean functions, that is, $\Delta = (\delta_1, \delta_2, \ldots, \delta_m)$ and $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_h)$; they implicitly define the *State Transition Graph* (STG) of the given FSM.
The *Transition Relation*, $T_M : S \times \Sigma \times S \rightarrow \{0,1\}$, of M describes all the pairs of states $(x,y) \in S \times S$ connected by an arc labeled $w \in \Sigma$ in the STG of M; T_M is defined as:

$$T_M(x_1, \ldots, x_m, w_1, \ldots, w_k, y_1, \ldots, y_m) = \prod_{i=1}^{m}(y_i \equiv \delta_i(x_1, \ldots, x_m, w_1, \ldots, w_k)),$$

where $x_1, ..., x_m$ are the present state variables, $w_1, ..., w_k$ are the primary input variables, and $x_1, ..., x_m$ are the next state variables of the FSM.
A *Discrete-Parameter Markov Chain* $\{X(t) | t \in T\}$ is an *Independent Stochastic Process* such that the number of possible states is finite and the parameter space T is discrete. The successive observations of the state of the system at different discrete time steps $t = 0, 1, 2, \ldots$, define the random variables $X_0, X_1, X_2, \ldots$ respectively, where, by convention, X_0 is the initial state of the system.
The Markov property says that the random variable representing the future behavior of the system does not depend on states reached in the past but only on the present state. This property can be formally stated as follows:

$$P(X_n = i_n | X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1}) = P(X_n = i_n | X_{n-1} = i_{n-1}).$$

Definition 2.1 *The* Discrete Density Function *of a random variable X_n is the probability $p_j(n) = P(X_n = j)$ of the system to be in state j at time n.*

Definition 2.2 *The* Conditional Discrete Density Function *of a random variable X_n is the probability $p_{jk}(m,n) = P(X_n = k | X_m = j)$, $0 \leq m \leq n$ of the system to be in state k at time n given that it is in state j at time m.*

In this paper we consider *Homogeneous Markov Chains*, that is, Markov chains for which $p_{ij}(m,n)$ depends only on the difference $n - m$; this implies that the system's behavior is independent from n; in this case, the Markov chain is said to have *Stationary Transition Probabilities.* The notation $p_{ij}(n) = P(X_{m+n} = j | X_m = i)$ is used to indicate the *n-Step Transition Probabilities*, while the *Initial Probability Vector* $\mathbf{p}(0) = [p_0(0), \cdots, p_n(0)]$ denotes the discrete density function of the random variable X_0.

Definition 2.3 *The* Transition Probability Matrix P_1 *of a Markov chain is an* $n \times n$ *matrix such that* $P_{1_{ij}} = p_{ij}(1)$.

Matrix P_1 can be seen as a STG description of the Markov chain. In fact, a node labeled i in the Markov chain corresponds to a state i in the STG, and $P_{1_{ij}}$ represents the probability of being in state j at time n, being in state i at time $n - 1$. Therefore, since a FSM implicitly specifies a STG, given a probability to every transition, a STG can be transformed into a Markov chain. By definition, every state in the STG has 2^k outgoing edges, where k is the number of primary inputs (PIs). Suppose that every combination of PIs of the FSM is equiprobable; then, every transition from state i to state j has probability 2^{-k}. Thus, given the transition relation $T_M(y, w, x)$, the transition probability matrix P_1 can be obtained as follows:

$$P_1(x,y) = 2^{-k} \cdot \sum_w T_M(x,w,y). \qquad (1)$$

If the input values are not equiprobable, the matrix is still obtained from the transition relation, but its derivation is slightly different, as will be shown in Section 4.
Also the initial probability vector can be determined from the information we have about the FSM; in fact, given the set of initial states of a FSM, $X_0 = \mathbf{p}(0)$ is obtained by assigning an initial probability value to every state in the set.
Given P_1 and $\mathbf{p}(0)$, the n-step probabilities are given by:

$$p_j(n) = P(X_n = j) = \sum_i p_i(0) p_{ij}(n). \qquad (2)$$

As a direct consequence, the n-step probability matrix can be obtained by taking powers of P_1, that is, $P_i = P_1^i$. In general, the probability distribution of the variable X_i is completely determined by the 1-step transition probability matrix P_1 and the initial probability vector $\mathbf{p}_0$.
We conclude this section by showing how we use ADDs to symbolically represent Markov chains. Given the STG of the Markov chain of Figure 1-a, whose probability matrix is presented in Figure 1-b, the corresponding ADD is the one depicted in Figure 1-c. x and y variables are used to encode present and next states respectively; therefore, each entry in the matrix is associated with a path in the ADD leading to a leaf whose value represents the transition probability between the two states.

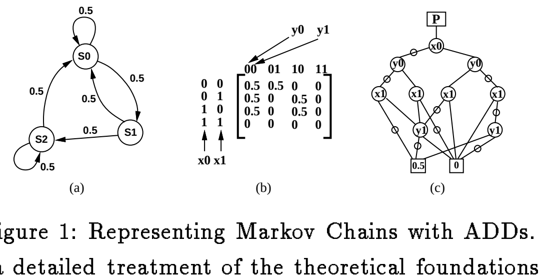

Figure 1: Representing Markov Chains with ADDs.

For a detailed treatment of the theoretical foundations of ADDs, as well as the definition of ADD operations such as *ITE*, *Apply*, and *Arithmetic Abstraction* (indicated in the following by $\backslash^+$), the reader may refer to [7].

3 Background on State Space Analysis

In this section we recall definitions and theorems related to the analysis of the state space of a Markov chain which are mostly taken from [1] and [2].

Definition 3.1 *State j is* transient *if and only if there is a positive probability that the process will not return to it.*

Then, if a state j is transient, $p_{ij}(n) \to 0$ when $n \to \infty$.

Definition 3.2 *State j is* recurrent *if and only if, starting from state j, the process eventually returns to it with probability 1.*

A set of transient states is called *transient*, and a set of recurrent states is called *ergodic*. Every STG has at least one ergodic set, but it may have no transient set. If an ergodic set contains a single state, that state is called *absorbing.* The probability, v_j, of a system to be in a particular state, j, is called the *Limit Probability* of that state.

Definition 3.3 *The* Limit Probability *of state j is:*

$$v_j = \lim_{n\to\infty} p_j(n) = \lim_{n\to\infty} \sum_i p_i(0) p_{ij}(n). \qquad (3)$$

Depending on the topology of the STG, the limit of Equation 3 may or may not exist, and if it exists its value may or may not depend on the initial probability vector $\mathbf{p}(0)$. Thus, different techniques are required to analyze the long-run behavior of a general FSM.
Let $G = (V, E)$ be a di-graph and let σ be the relation defined over the set of vertices V by $v\sigma w$ if and only if there is a path from v to w of arbitrary length; σ is an equivalence relation; therefore, it induces a partition in the set of nodes. Every equivalence class is called a *strongly connected component* (SCC) of G. The quotient graph obtained by representing every SCC as a single node and preserving the edge relations between SCCs from the original graph G, is called the *SCC graph of G*, written $G_{SCC} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is obtained from V by representing every SCC by a unique node, and $\mathcal{E}$ is the relation between SCCs derived from E. G_{SCC} is an acyclic graph, otherwise the nodes involved in a cycle would be in the same equivalence class. We will denote as *terminal strongly connected components* (TSCCs) the sinks of the graph G_{SCC}. Clearly, once the system reaches a TSCC, it will never leave it.

Lemma 3.1 *Every TSCC of the STG is an ergodic set, and every non-terminal SCC is a transient set.*

Definition 3.4 *For a recurrent state i, $p_{ii}(n) > 0$ for some $n \geq 1$. The* period *of state i, denoted by d_i, is the greatest common divisor of the set of positive integers n such that $p_{ii}(n) > 0$.*

If a recurrent state i has period $d_i = 1$, then it is called *aperiodic.*

Definition 3.5 *The* period *of a TSCC is the greatest common divisor of the length of all its closed paths.*

Definition 3.6 *A system is* non-decomposable *if every state can be reached from every other state in a finite number of steps, that is, there exists an integer $n \geq 1$ such that $p_{ij}(n) > 0$.*

The matrix P_1 of a decomposable system can be put in lower block triangular form.

Theorem 3.1 *For any non-decomposable, aperiodic system, the limiting probabilities $v_j = \lim_{n\to\infty} p_j(n)$ exist and are independent of the initial distribution $\mathbf{p}(0)$.*

In the case of periodic systems, when $n \to \infty$ the value of $p_{ij}(n)$ does not converge. Instead, a periodic series of vectors is obtained with period d, that is, $p_i(n) = p_i(n+d)$.

Following the definitions above, sequential systems can be classified as:

- **Non-Decomposable:** Systems having a unique SCC (which is, obviously, a TSCC); they can be:
  - Aperiodic: the limit probabilities are guaranteed to exist.
  - Periodic: when $n \to \infty$, the probabilities oscillate with the period of the TSCC.
- **Decomposable:** Systems having more than one TSCC; a preliminary analysis is done on the graph where every TSCC is collapsed into a single node. The limiting probabilities of the modified system reflect the probability of the system to be in each TSCC. These probabilities are used to normalize the solution obtained when computing the limiting probabilities of every single TSCC, which is now non-decomposable.

4 Computing the Limit Probabilities

In Section 3 we have seen that for non-decomposable, aperiodic systems, the v_j's are guaranteed to exist and they can be obtained by means of Equation 3. In [6] we have proposed efficient ADD-based algorithms to perform the computation of those probabilities in this particular case. Here, we extend our approach to deal with the most general case of systems having state graphs of arbitrary structure; therefore, we have to analyze non-decomposable, periodic systems, as well as decomposable systems (which may contain both aperiodic and periodic components).
Though most of the systems one encounters in practice are non-decomposable [8], there is an important reason, beyond generality, to address decomposable systems. As will be discussed in Section 4.6, the analysis of large decomposable systems can be profitably reduced to the analysis of several non-decomposable systems. To do this, we need to use reachability analysis, i.e., FSM traversal, to determine both the structure of the system under examination and the period of each single structural component.
We start this section by formally stating our problem. Then we consider systems for which the primary inputs are not equiprobable; this is a key issue when the techniques we are developing have to be applied to model real hardware devices. Then we briefly show how we handle the conceptually simple, but computationally difficult, case of non-decomposable, aperiodic systems; for a more detailed treatment the reader can refer to [6]. Then we move to the case of non-decomposable, periodic systems; in this case, we first determine the period, d, of the Markov chain by implicitly traversing its STG, and then we use the information on the periodicity of the system to check whether the iterative calculation of the limit probabilities has converged or not by comparing two probability vectors, $p_i(n)$ and $p_i(n+d)$, whose temporal distance is d. Finally, we consider the case of decomposable systems, and we show how we can decompose them into simpler non-decomposable systems which can be analyzed using the techniques mentioned above.

4.1 Problem Formulation

As mentioned in Section 2, by assigning different weights to the edges, the STG of a FSM can be translated into a Markov chain. Every node in the STG has 2^m out-going edges, where m is the number of primary inputs of the machine. The 1-step probability matrix can be obtained from the transition relation, in the case where all the primary inputs are considered equiprobable, in the following way:

$$P_1(x,y) = 2^{-m} \cdot \backslash_w^{+} T(x,w,y) \qquad (4)$$

where the operator $\backslash_d^{\mathbf{op}}$ represents the *scan* operation defined as $\backslash_d^{\mathbf{op}} T(d) = (T_{d_0} \ \mathbf{op}\ T_{d_1}\ \mathbf{op}\ \ldots\ \mathbf{op}\ T_{d_n})$.

4.2 Case of Non-Equiprobable Inputs

Considering systems for which the primary inputs are not equiprobable is a key issue when the techniques we are developing have to be applied to model real hardware devices. For signals like *reset* or *load*, for instance, usually $P(input_i = 0) \neq P(input_i = 1)$.
In the case that not all the primary inputs of a FSM are equiprobable, the 1-step transition probability is obtained by the algorithm in Figure 2. The function accepts three parameters, the transition relation T, a cube in the primary inputs C, and an array Π, where $\Pi[i]$ is the probability of input i to be one. The procedure is similar to abstraction, with the exception that when a variable in C is missing from F, the result is added to itself. When a variable has to be abstracted, a convex combination of the two subfunctions T_1 and T_0 is taken instead. The algorithm uses a table (not shown in Figure 2) to store previously computed results as all the ADD procedures.

```
ComputeTM(T, C, Π) {
    if(T is constant or C = 1) return T;
    if(top(T) > top(C)) return ComputeTM(T, then(C), Π);
    if(top(T) = top(C)) {
        T_1 = ComputeTM(then(T), then(C), Π);
        T_0 = ComputeTM(else(T), then(C), Π);
        return T_1 · Π(top(T)) + T_0 · (1 − Π(top(T)));
    } else {
        T_1 = ComputeTM(then(T), C, Π);
        T_0 = ComputeTM(else(T), C, Π);
        return ITE(top(T), T_1, T_0);
    }
}
```

Figure 2: Algorithm to Compute Conditional Transition Probabilities from Primary Input Probabilities.

The possibility of setting some probabilities of the inputs to values different from 0.5 may produce substantial changes in the STG structure. In particular, forcing an input to a fixed value in $\{0,1\}$, that is, either $p_i = 0$ or $p_i = 1$, implies deleting some edges from the original STG.

4.3 Non-Decomposable Aperiodic Systems

There are several numerical methods to calculate the limit probabilities, but not all of them are suitable for ADD-based computation. The limit probabilities can be calculated by solving the system of the so-called *Chapman-Kolmogorov* equations. One way to solve this system of linear equations would be to use Gaussian elimination; although the matrix of the coefficients may be regular, this regularity may disappear when Gaussian elimination is performed. Therefore, as we have shown in [6], direct methods cannot be applied to very large systems, and iterative methods need to be used.

4.4 Structural Analysis of FSMs

In order to be able to treat arbitrary systems, some structural information needs to be extracted from the STG. The first calculation we do is the set of reachable states. Since the FSM has a set of initial states, only those reachable from any initial state will be considered. The traversal procedure is entirely based on BDDs.
The fact that only the edges between reachable states are meaningful is used to reduce the size of the representation of the transition relation. Then the TSCCs are determined by applying the procedure presented by Matsunaga *et al.* in [9] which calculates the transitive closure of a transition relation.
In general, each TSCC may have a different period; therefore the computation of the period of each TSCC, necessary to check the convergence when solving each sub-system of equations, is done by traversing every single TSCC separately. The reset state of the TSCC being traversed is picked as one arbitrary state inside the TSCC.

4.5 Non-Decomposable Periodic Systems

In general, the limit probabilities are not independent of the initial probabilities (see Section 2). This is the case for periodic FSMs. Figure 3 shows a FSM with period $d = 6$. Depending on the initial probabilities $\mathbf{p}(0)$, the series of vectors obtained by solving the system of linear equations oscillates with a different period. If the system shown in Figure 3 is solved with the initial probability vector: $p_1(0) = 1, p_2(0) = \cdots = p_9(0) = 0$, then the series oscillates with period 6. On the other hand, if an equiprobable initial probability vector is used, the solution obtained has period equal to 2. In general, if the period of a circuit is d, it is possible to find an initial assignment of probabilities such that the computation oscillates with any period that is a divisor of d.

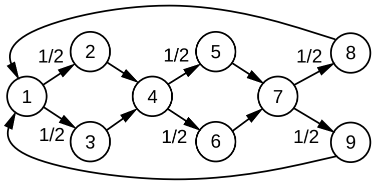

Figure 3: Periodic FSM with Period $d = 6$.

In this case, for a non-converging sequence $s_0, s_1, \ldots$, a sequence of averages can be formed, and taken as the new sequence. The original sequence is said to be *summable* by means of the averaging process. We will consider only the following averaging method:

$$t_n = \frac{1}{n} \sum_{i=0}^{n-1} s_i. \tag{5}$$

This expression is an average of terms of the sequence with non-negative coefficients whose sum is 1. If the sequence $t_0, t_1, \ldots$ converges to a limit t, then we say that the original sequence is *Cesauro-summable* to t [1].

Theorem 4.1 *If P is an ergodic transition matrix, then the sequence P^n is Cesauro-summable to a limit matrix.*

The theorem above says that, in the case of periodic systems, the sequence P^n has a limit; this implies that, even though $P \cdot v^i$ oscillates with period d, the limit of the averaged series is constant. Hence, once the series is detected to oscillate, the limit probability for every state is given by:

$$p_i = \frac{1}{d} \cdot \sum_{j=0}^{d-1} v_i^{k-j}. \tag{6}$$

The discussion above motivates the selection of a proper initial guess for the solution of the system of linear equations. Two different strategies have been considered. If the vector is chosen to have only one state with probability 1 and the remaining states with probability 0, the convergence will be achieved after a large number of iterations if the depth of the machine is large. On the other hand, considering an equiprobable initial guess may produce also large number of iterations if the solution is far from being equiprobable.
Let us recall that a non-decomposable system, either aperiodic or periodic, has a unique SCC; hence, this SCC is terminal. For systems of this type, the limit probability vector does not have any zero entry, except those due to numerical errors.

Figure 4 shows the pseudo-code of the algorithm to analyze non-decomposable, periodic systems. It should be noticed, however, that the same algorithm works also for aperiodic systems, being an aperiodic system a periodic system with period of length 1; the reason why we use ad-hoc solution methods for aperiodic systems (see [6]) has to do with efficiency in the computation.

```
SolveSystem($P, InitG, d$) {
    $Converged$ = false;
    $i = 0$;
    $Solution[i] = InitG$;
    while (not $Converged$) {
        $i = i + 1$;
        $x^i = P \cdot Solution[(i-1) mod\ d]$;
        if ($iter \geq d$)
            $Converged = Check(x^i, Solution[(i\ mod\ d)])$;
        $Solution[i\ mod\ d] = x^i$;
    }
    $p = 0$;
    for every $Solution[i]$
        $p = p + Solution[i]$;
    return $p/d$;
}
```

Figure 4: Algorithm to Solve Periodic Systems.

Given the period d, the algorithm keeps the solution of the last d systems in a *sliding window*. Whenever a new solution is found, it is checked against the solution obtained d steps before. If the norm $\|x^{k+d} - x^k\|$ is less than a given tolerance, convergence is reported. For every state, the solution is the average probability through the period:

$$Solution = [S^l, \cdots, S^{l+d-1}] \tag{7}$$

$$v_i = \frac{1}{d} \cdot \sum_{j=l}^{l+d-1} S_i^j \ , i = 0 \cdots n \tag{8}$$

4.6 Quasi-Decomposability

The notion of decomposability gives a way to analyze complex systems in terms of smaller sub-systems. However, in practice, it is not very common to find systems having STGs with several TSCCs. Rather, it happens very often that real systems have only one, large TSCC; this sometimes makes the computation of the limit probabilities numerically unstable and, therefore, convergence becomes difficult to achieve. One technique to reduce the complexity of a system, is to fix some primary inputs to specific boolean values. In terms of probabilities, this turns into fixing the probability value of some inputs to either 1 or 0. This simplification is not far from the real behavior of signals with very low probability of being in one of two states.
Setting a given input, w_i, to a fixed boolean value induces a pruning on the edges of the STG associated to the system under investigation; in fact, the predicates on some edges may be no longer satisfiable, implying that those edges of the graph will never be traversed, and therefore, they can be pruned. The system obtained in this way may be decomposable. In this case, we say that the system is *Quasi-Decomposable*, reflecting the fact that some inputs are responsible for making this system non-decomposable, but when set to a specific value, the new system is decomposable. Since the analysis of the new system is reduced to its TSCCs, a considerable reduction on the state space might be achieved as well.
Clearly, limit probabilities of the system in which some of the primary input signals have been set to either 1 or 0 differ from the ones of the system for which all the primary inputs have non-fixed values; this is because the sets of reachable states of the two systems are now different.

Let $W = (w_0, \cdots, w_k)$ be the set of primary inputs of the system, and let $PI = (pi_0, \ldots, pi_k)$ the input probability vector, that is, each pi_i is the probability of input i to be one. Let S the set of primary inputs which assume the fixed value one, T the set of primary inputs which assume the fixed value zero, and Q the set of primary inputs with non-fixed value (clearly, $S \cap T = \emptyset$). Let Γ_{orig} be the original system (i.e., the system for which $S = \emptyset$ and $T = \emptyset$), and let Γ_{new} be the system in which some of the primary input signals have a fixed value). The limiting probabilities of Γ_{orig} can be calculated starting from Γ_{new} by considering the case where the pi_i's of the primary inputs $i \in S$ have a value $pi_i - \epsilon$ and the pi_i's of the primary inputs $i \in T$ have a value $pi_i + \epsilon$, being $\epsilon > 0$. Let us denote this new system as Γ'_{new}.
By choosing a sufficiently small ϵ, the solution of Γ'_{new} is close to the solution of Γ_{new} [3]. Once the solution of Γ'_{new} is determined, a new system can be built by choosing a larger ϵ. The process may be repeated until the probabilities of all the primary inputs match the ones given by PI, that is, when the system solved is exactly Γ_{orig}.

4.7 Decomposable Systems

Figure 5-a represents a simple system which is decomposable, that is, it has more than one TSCC.

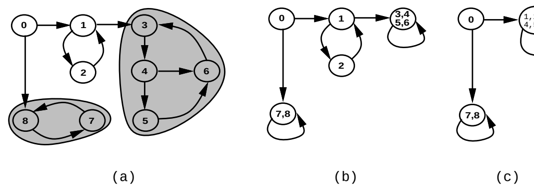

Figure 5: STG Having Two TSCCs (a), the Collapsed Graph (b), and the Efficiently Collapsed Graph (c).

The concept of *lumpability* will be used to simplify this kind of systems [1].

Theorem 4.2 *Let A be a partition of the state space of a STG C into N blocks (macro-states). Define the order N matrix $Q = [q_{IJ}]$ as*

$$q_{IJ} = \frac{\sum_{i \in A_I} \pi_i p_{iJ}}{\sum_{i \in A_I} \pi_i},$$

where $p_{iJ} = \sum_{j \in A_J} p_{ij}$ and π is the vector of the stationary probabilities of C. Then Q is stochastic and non-decomposable. Moreover, if Π is the stationary probability row vector for Q, then $\Pi_I = \sum_{i \in A_I} \pi_i$ for $I = 1, \ldots, N$.

In general, the behaviors of the original and the lumped system may differ. However, the following theorem gives a necessary and sufficient condition to lump several states while preserving the statistical behavior of the system.

Theorem 4.3 *A homogeneous Markov chain is lumpable with respect to a partition $(A_1, \ldots, A_R)$ if and only if all the q_{iA_j} have the same value, W_{ij}, for every $i \in A_i$, and for any given $A_j \neq A_i$.*

For a lumpable STG, the construction of the lumped graph is simple. Each block of the partition (a macro-state) is a state of the original STG and the q_{iA_j} are the conditional probabilities given by Theorem 4.2.
A TSCC can be considered a special case for the above theorem because there is no outgoing edge. Hence, by lumping the states in a TSCC into a representative, an *absorbing* node is created and the necessary condition of Theorem 4.3 holds. As a conclusion, every TSCC can be collapsed into a single node without outgoing edges, and the new system can be analyzed following the same technique. Further reduction could be achieved by detecting SCCs whose fanout goes entirely to another SCC. By Theorem 4.3, both can be collapsed. Figure 5-c depicts this special case. Notice that the collapsed graph has as many ergodic sets as there are TSCCs in the original system, and that the limit probabilities will be different from zero only for the absorbing states. Let us assume that the system can be decomposed into l different TSCCs, $T_0, \ldots, T_{l-1}$, and let us denote the limit probabilities of the absorbing states as $v_0^T, \ldots, v_{l-1}^T$. Given that every T_i is collapsed, v_i^T denotes the probability of the system being in any state inside T_i.
Thus, once the solution of the collapsed system has been obtained, the limit probability for every state in every TSCC still needs to be calculated. However, $T_0, \cdots, T_{l-1}$ now can be analyzed as l independent non-decomposable systems. If we denote by in_i the limit probability for state i obtained by analyzing the TSCC as an independent subsystem, the limit probabilities for the global system are obtained by the equation:

$$\forall \text{ state } i \in T_j, \quad v_i = in_i / v_j^T, \quad 0 \leq j < l.$$

The collapsing procedure is shown in Figure 6.

```
collapseTSCC(TR, TC) {
    $ExternalEdges(x,y) = TC(x,y) \cdot \overline{TC(y,x)}$;
    $NoFanout(x) = \overline{\exists_y(ExternalEdges(x,y))}$;
    $CollapsedTR(x,y) = TR(x,y)$;
    $indexSCC$ = table (representative,scc);
    while($NoFanout \neq \emptyset$) {
        $cube$ = Pick a minterm from $NoFanout$;
        $scc$ = Nodes reachable from $cube$ with a cycle;
        $sccFI$ = Predecessors of $scc$ outside $scc$;
        add edges from $sccFI$ to $cube$ into $CollapsedTR$;
        delete edges inside $scc$ from $CollapsedTR$;
        delete nodes in $scc$ from $NoFanout$;
        add the pair ($cube$,$scc$) to $indexSCC$;
    }
    return $indexSCC, CollapsedTR$;
}
```

Figure 6: Algorithm to Collapse the TSCCs in a STG.

5 An Application: The GCD Circuit

A circuit which computes the *Greatest Common Divisor* (GCD) of two n-bit numbers a and b works as follows. The two numbers a and b are loaded in two registers, A and B; a third n-bit register R is initialized to 1. At each clock cycle, the LSBs of A and B are examined. If A and B are both even, then R is shifted left. If A is even and B is odd, A is shifted right; if B is even and A is odd, B is shifted right. If both A and B are odd, $|A - B|$ replaces the largest of A and B. Termination occurs when $A = B$ or when one of the two registers equals R. The result is the product of the smaller of A and B, and R. Figure 7-a represents one possible implementation of this scheme.
The cycle time of the circuit will typically be determined by the time taken by the subtractor. Suppose we are interested in estimating the increase in speed that would derive from doubling the clock frequency and allowing the subtractor two cycles to complete. Such an estimate can be obtained by computing the probability for the circuit to be in a state where the LSBs of A and B are both one. This in turn can be obtained from the state probabilities by first cofactoring the limit probability vector with respect to the LSBs and then summing over all the other state variables. Since the GCD algorithm only considers the LSB of each operand, a simplified model could be built considering only these two bits (see Figure 7-b). The limit probabilities of the simplified system are $v_{00} = 0$, $v_{01} = v_{10} = 0.25$, $v_{11} = 0.5$. The interpretation of this result is that a subtraction is performed every two cycles, thus if the clock frequency is doubled, the average speed would increase by 25%. However, in reducing the system, the implicit assumption that the two numbers have an infinite number

of digits has been made. Therefore, the analysis of the simplified system may lead to an erroneous solution.

In Section 6 we report results for two GCD circuits with 14 and 26 state variables. In the case of **gcd4**, experimental data show that the probability of being in a state where no subtraction is performed is 0.81. If the frequency of the clock is doubled, only the transitions that do not involve a subtraction will contribute to increase the average speed. Therefore, a 40% increase on the average speed will be achieved. On the other hand, when the number of bits of the operands is doubled the circuit is expected to execute more subtractions and this probability may change. In fact, for **gcd8** (which has operands with length double than **gcd4**), the experiments show that with double frequency the system would increase its speed by 36%.

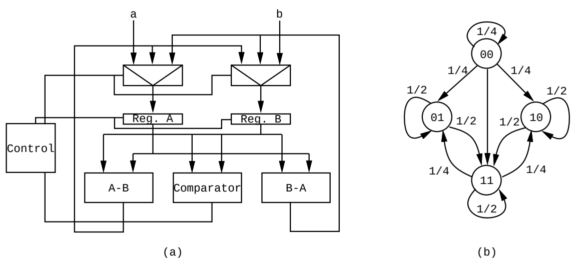

Figure 7: Circuit Implementing the GCD Algorithm (a), and the Simplified System (b).

6 Experimental Results

The experimental results obtained by applying the techniques described in this paper are reported in Table 1. For every circuit, the total number of states, the number of states whose limit probability is different from zero, the number of iterations and the execution time needed to calculate the limit probabilities are shown. Time is in seconds on a DEC-Station 5000/200 with 80 MB of memory.

Circuit	*States*	*Non-zero*	*Iter*	*Time*
s444	2.1e+6	469	17	2.93
s953	5.37e+8	504	58	127.22
mm9nr	1.34e+8	512	2	0.12
mm30nr	1.23e+27	1.07e+9	2	8.54
gcd4	16384	487	29	5.72
gcd8	6.71e+7	90311	84	1336.43
mltpl	1.68e+7	70913	90	20.96
mult16a	65536	55952	16	15991

Table 1: Experimental Results.

Circuits **s444** and **s953** from the ISCAS'89 benchmarks [10] are non-decomposable, aperiodic systems. In this case no simplification is done, and the circuit is solved as a whole. The **minmax** circuits [5] (**mm9nr** and **mm30nr**) are examples of quasi-decomposable systems. In this case the *reset* signal was set to probability zero, and both circuits turned into decomposable systems with a unique TSCC. This is a special case of decomposability, because when applying the collapsing technique, only a sink node is produced; thus, the solution of the simplified system is trivial, with probability equal to 1 in the state representing the unique TSCC. The greatest common divisor circuits (**gcd4** and **gcd8**) were discussed in more detail in Section 5. Circuit **mltpl** is an example of decomposable system. It has several TSCCs, some of them aperiodic and others with different periods. The number of iterations reported are those to solve the collapsed system and the set of TSCCs. The fact that a TSCC is periodic forces the number of iterations to solve the circuit to be at least as large as the period. The case of the circuit *mult16a* is a circuit with 64435 states in a unique TSCC, and every state has a different probability. For that reason the ADDs barely have any recombination and therefore, execution time is large.

7 Conclusions and Future Work

Probabilistic analysis of the behavior of finite state machines can be very useful in the verification and synthesis of sequential circuits.

Markov chains have been used extensively in the quantitative study of sequential systems. Their application to large systems has been made possible by sophisticated numerical techniques and skillful modeling. Until today, however, the direct analysis of systems with very many states (10^8 or more) has remained problematic at best.

In [6], we have proposed symbolic procedures to compute the limit probabilities for very large FSMs having non-decomposable, aperiodic transition structures, that is, machines with state graphs composed of a single terminal strongly connected component.

In this paper we have generalized our approach by making it able to handle systems having state graphs of arbitrary structure; therefore, we have considered the case of non-decomposable, periodic systems, as well as decomposable systems (that may be contain both aperiodic and periodic components). We have used symbolic reachability analysis techniques to perform both decomposability and periodicity investigation, and we have exploited the information calculated during this step to increase the efficiency of the iterative methods of solution of large systems of linear equations. Experimental results are very promising; in fact, by applying our techniques we have been able to calculate the limit probabilities for systems whose corresponding finite state models have more than 10^{27} states.

As future work, the simplification technique based on collapsing states in the same TSCC, although effective, can be exploited in more depth by considering sets of states not necessarily in the same TSCC. In that sense, there is Markov chain theory to support this approach, namely aggregation and decomposition and a closer look at it is being considered. Additional improvements are also needed in the numerical algorithms. In that direction, a new data structure derived from ADDs is being studied. The method presented here has a constant matrix P_1 and a variable vector of states. This situation is suitable for a more specialized code to perform matrix multiplication.

References

[1] J. Kemeny, J. Snell, *Finite Markov Chains*, D. Van Nostrand Company, 1967.

[2] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*, Prentice-Hall, 1982.

[3] P. J. Courtois, *Decomposability: Queueing and Computer System Applications*, Academic Press, 1977.

[4] W. J. Stewart, *Numerical Solutions of Markov Chains*, Marcel Dekker, 1991.

[5] O. Coudert, C. Berthet, J. C. Madre, "Verification of Sequential Machines Using Boolean Functional Vectors," *IFIP Intl. Workshop on Applied Formal Methods for Correct VLSI Design*, pp. 111-128, Leuven, Belgium, Nov. 1989.

[6] G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Symbolic Algorithms to Calculate Steady-State Probabilities of a Finite State Machine," *EDAC-94*, pp. 214-218, Paris, France, Feb. 1994.

[7] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Algebraic Decision Diagrams and their Applications," *ICCAD-93*, Santa Clara, CA, Nov. 1993.

[8] C. Pixley, "A Theory and Implementation of Sequential Hardware Equivalence,", *IEEE Transactions on CAD*, Vol. 11, No. 12, pp. 1469-1478, Dec. 1992.

[9] Y. Matsunaga, P. C. McGeer, R. K. Brayton, "On Computing the Transitive Closure of a State Transition Relation," *DAC-30*, pp. 260-265, Dallas, TX, Jun. 1993.

[10] F. Brglez, D. Bryan, K. Koźmiński, "Combinational Profiles of Sequential Benchmark Circuits," *ISCAS-89*, pp. 1929-1934, Portland, OR, May 1989.