Evolutionary Algorithm Approach for Symbolic FSM Traversals*

Mitchell A. Thornton
Mississippi State University
Mississippi State, MS
mitch@ece.msstate.edu

Rolf Drechsler
Siemens AG
Munich, Germany
drechsle@informatik.uni-freiburg.de

***Abstract*--State space traversal algorithms for *Finite State Machine* (FSM) models of synchronous sequential circuitry are used extensively in various formal verification approaches such as *Equivalence Checking* (EC) and model checking. Symbolic *Binary Decision Diagram* (BDD) based approaches have allowed many FSM models to be verified due to the compact representations they provide. However, there still remain circuits for which the traversal cannot be carried out due to the size of the *Transition Relation* (TR) BDD becoming too large. Pruning algorithms designed to reduce the size of a BDD while maintaining as much functionality as possible are examined for here. These techniques are based upon evolutionary algorithms that have bee shown to significantly reduce the size of BDDs while retaining a large amount of functionality.**

***Index Terms*—FSM Equivalence Checking, Evolutionary Algorithms, Binary Decision Diagrams, Verification**

I. Introduction

The desire to achieve true "static sign-off" in the design process has motivated the incorporation of formal verification methods in modern design flows. Typically, tools such as *Equivalence Checkers* (EC) [1] and model checkers are employed [2,3]. Many design flows use model checkers to verify certain characteristics of the *High Level Language* (HLL) specification or the *Hardware Description Language* (HDL) golden model. Next, *Finite State Machine* (FSM) EC may be used to verify that all subsequent design abstractions are functionally equivalent.

Overview descriptions of basic symbolic FSM EC are available in [1,3,4,5,6] as well as many other sources. The main idea for forward, breadth-first state traversal methods is to form a product FSM using the two abstractions that are being verified. The electronic model of the product machine consists of a composite FSM where all similar named inputs and the clock are electrically common. All like named outputs are used as input signals to a common equivalence gate (XNOR). Next, each component FSM is initialized to the same state and the product machine execution is simulated until the entire state space of the product machine has been traversed. Although many synchronous circuits have been successfully verified using this technique, the straightforward implementation of the EC algorithm may often not converge in a reasonable amount of time or it may require too much memory

One approach to this problem is to utilize an approximate EC that over- or under-estimates the set of states traversed by a FSM. Several approaches for performing approximate EC have been proposed [7,8,9,10,11,12]. Approximate EC results can be used in several different ways. If true formal verification is desired, a bounding EC can be used to quickly rule out states that need not be checked by disregarding those that the over-approximation method did not find in the state space. Additionally, those states that the under-approximation method did traverse need not be examined further since it is known that this set of states definitely is in the state space. For those states not ruled out, an alternative technique can be employed such as a directed depth-first reachability analysis or further checking of properties or even theorem proving. Another important use of approximate ECs is the generation of valid counterexamples when two abstractions of a circuit have been found to differ in functionality.

The overall objective of the work described in this paper is the development of an FSM EC tool that utilizes the concepts of *Evolutionary Algorithms* (EAs). EAs have been applied to several methods in VLSI CAD [13]. Examples include methods for finding *Binary Decision Diagram* (BDD) variable orderings [14,15] and FPGA mapping and synthesis techniques [16,17]. The goal here is to produce an EC tool that can more accurately approximate the reachability of a FSM as compared to other approximate approaches in a reasonable amount of computation time. If a finite probability of design error is acceptable, the EA employed for finding the approximated *Transition Relation* (TR) can be used to trade runtime for a decrease in the error probability until some threshold is reached.

Other approximation approaches typically fall into two categories; either partitioning the design into a set of smaller interconnected state machines and performing exact symbolic analysis on each, or, exploring a subset of the state space when some threshold or computational resource limit has been reached.

The method described here differs in that an EA approach is used to compute the over-approximation of the reachable state set. The over-approximation results since the reduced

This work was supported in part by the U.S. National Science Foundation (NSF) under Grants No. CCR-0000891 and INT-0096008 and; by the German Deutscher Akademischer Austauschdienst (DAAD) under Grant No. 315/PPP/gü-ab.

TR BDD is constructed such that it is constrained to be smaller (in terms of vertices) but represents the true TR as closely as possible with an error being that all true TR logic-0s are forced to be logic-1 values. The other over-approximation techniques described previously did not utilize EAs.

To describe the technique in more detail, the following paragraphs will give a brief introduction to the basic concepts of symbolic FSM EC and EAs. Next, the application of an EA to TR BDD pruning will be described and experimental results of incorporating this technique in FSM equivalence checking will be provided.

II. Symbolic FSM Equivalence Checking

Overview descriptions of symbolic FSM equivalence checking are available in [1,3,18,19] as well as many other sources. The basic idea for forward, breadth-first state traversal methods is to form a product FSM using the two abstractions that are being verified. The electronic model of the product machine consists of a composite FSM where all similar named inputs and the clock are electrically common. All like named outputs are used as input signals to a common equivalence gate (XNOR). Next, each component FSM is initialized to the same state and the product machine execution is simulated until the entire state space of the product machine has been traversed.

Symbolically, a *Transition Relation* (TR) is formed that consists of a function representing all possible transitions of a component FSM. This can be considered to be a single output Boolean function that has all possible present-state and next-state vector pairs as possible inputs. For those pairs where a transition is possible, the TR evaluates to logic-1. For those pairs that are not possible, a logic-0 is produced. Also, a function representing the initial state is formed that evaluates to logic-1 for the product term(s) representing an initial state set. These two functions are commonly represented as ordered *Binary Decision Diagrams* (BDDs). The explicit formation of the TR is commonly referred to as the *monolithic* TR since its' size can be quite large.

The next portion of the algorithm is the iterative image computation. The BDD initially representing the reset state is updated during each iterative step to represent all the states that have been reached in the symbolic execution of the product machine. Additionally, another BDD is formed that represents all the states reached in the last iteration of the algorithm. Tautology checking must be performed at each iteration to ensure the component FSMs are indeed producing the exact same outputs. The image computation involves forming the conjunction of the BDD representing the states reached in the last iteration (or symbolic execution of the product machine) with the TR followed by an existential quantification (smoothing operation) of the resultant BDD over the present state variables. The image BDD then is used as the currently reached state BDD in the subsequent iteration with all the next variables relabeled as the corresponding present state variables. Also, the BDD that accumulates all the currently reached states is updated as the disjunction of itself with the currently reached states just computed.

This iterative, *Image Computation* step continues until a non-tautology is found indicating non-equivalence, or, until the BDD representing all the previously reached states does not change anymore, indicating the entire state space has been traversed. Upon this convergence, the BDD representing the reached states actually represents the entire state space of the two component FSMs.

Problems can arise in the formation of the transition relation BDD, the intermediate size of the BDD representing the "reached" states and the peak intermediate BDD size that can result during the image computation. In addition, the number of iterations can also be exponential in size. In the worst case, a component FSM with m state variables and n inputs can result in 2^m reachable states and 2^n iterations. Clearly, this algorithm is complex both in terms of storage and runtime resources. This fact has motivated many researchers to enhance the basic algorithm described above. Several of these enhancements are described in detail in the reference works included in the bibliography. The complexity of this technique also motivates the investigation of approximation techniques that can be used to provide estimates, or, can be used with another criteria (such as a guided search strategy) to perform exact equivalence checking.

One major problem with the method described above arises during the explicit computation of the TR BDD. The so-called "monolithic" TR BDD is formed as the conjunction of all the next-state BDDs followed by smoothing out the input variable dependence. The fact that the TR BDD can grow to an unacceptable size has lead to methods where the TR is represented implicitly by maintaining an array of the next-state BDDs or through clustering some of the next-state BDDs, but not all of them. Also methods such as early quantification in building monolithic or clustered TR BDDs are very helpful [18]. Early quantification requires the identification of disjoint sets of input variables in the support of next state functions before computing their conjunction. Once the disjoint sets of input variables are found, those not in the common support set may be quantified out *before* the conjunction of next-state BDDs is performed.

Also, it is apparent that the exact TR relation is not required in intermediate image computations. Rather, some function that has the same behavior as the exact TR may be used during a single iteration. As long as a function preserves the same behavior in mapping a subset of reached states to the corresponding set of resulting next-states, the substitution is acceptable. This requires the formation of a new reduced TR at each image computation but can lead to algorithms that can successfully traverse the state space of FSMs where the use of the monolithic TR is prohibitive. One

way to exploit this property is to use the notion of the generalized co-factor [18,19]. The basic premise of the technique is that the inclusion of any of states already reached in iterations before the current one may be included in the currently reached state BDD. Then, using this BDD, the reduced TR may be arrived at by ensuring that it has the same behavior for this subset of states as the monolithic BDD has.

III. Evolutionary Algorithms

An *Evolutionary Algorithm* (EA) is a common term for algorithms that utilize adaptive behavior modeled after principles of nature. This class of algorithms contains genetic algorithms, evolutionary strategies and evolutionary and genetic programming. Although definitions of evolutionary algorithms differ, the more common properties of EAs are that a collection of potential solutions to the problem at hand are maintained referred to as the *population* of a current *generation*. Operations are applied to the current population to produce a new generation that will hopefully contain members that are a "better" solution to the problem in some sense. This process continues until some threshold or stopping criterion is met. Once the stopping criteria occurs, the "best" solution from the current population is used.

The new population is produced through the application of *operators* on selected members of the current generation. Typically, the members of the current generation to whom the operators are applied are chosen based on their quality of solution to the problem. In this way, it is more likely that desirable characteristics are inherited by the offspring solutions. Some concepts in the development of EAs are the criteria for choosing parents in a current generation, the halting criteria, the measure of the quality of the solution an individual provides (i.e. evaluation of the *fitness function*) and the operators that are used to generate offspring.

Typical operators are developed from the genetic notions of *mutation*, *recombination* (or *crossover*) and *inversion*. As an example of a mutation operator, a random number generator may be employed to randomly change some inherited characteristic. Crossover operations involve combining characteristics from two or more parents and placing them in the resulting offspring. Inversion involves rearranging the inherited traits of parents, but this only makes sense if there is some characteristic of interest that is related to the order of inherited traits.

A. BDD Pruning Using EAs

Recently, an EA approach was developed for Multiplexer circuit based synthesis utilizing a BDD representation that is of particular interest in the context of this proposal. In [17], a technique was described where a BDD representation of a function to be mapped to a multiplexer-based *Field Programmable Gate Array* (FPGA) was pruned until it would fit entirely within a target device. The pruning was accomplished through the use of an EA that sought to reduce the size of the BDD (in terms of vertices) while maintaining as little error as possible. While some circuit error is introduced, the tradeoff between BDD size reduction versus function error was quite impressive in some cases. As an example it was shown that, for most cases, benchmark functions in the ISCAS85 set had less than 1% error (in terms of incorrect minterm values) when 20% of the vertices were removed. Increasing the BDD size reduction to 50% resulted in errors of less than 10% in all cases tested.

The methods in [17] involved the use of two mutation operators and one crossover operator. The first mutation operator, MUT1, randomly chose an internal variable in the BDD and replaced it by a constant. The other mutation operator, MUT2, was an enhancement of MUT1 in that each internal vertex was considered to be the root node of a subfunction. The number of minterms covered by the subfunctions is extremely easy to compute thus, this value is computed for each internal vertex. The vertex that has the most extreme minterm count, whether it is closer to 0% or 100% is chosen and replaced with the constant-0 or constant-1 vertex respectively. The crossover operator combines two functions f and g. This is accomplished by recursively traversing both BDDs and searching for identical subtrees, when such subtrees are found, a node pointing to them is returned for inclusion in the child BDD.

B. Approach for EAs in FSM Equivalence Checking

The technique implemented in this work uses EAs for TR BDD pruning. The goal is to reduce the storage requirements of the TR BDD while degrading the representation error of the original function as little as possible. If such pruning is applied to the TR, constraining all errors to be of the type $0 \rightarrow 1$ results in an over-approximation scheme while errors of type $1 \rightarrow 0$ cause under-approximation to occur (the notation $0 \rightarrow 1$ refers to the case where the actual TR would result in a logic-0 while the approximated one yields a logic-1).

An initial population was formed through the application of MUT1 operator during the construction of the TR. In subsequent iterations, a modification of the MUT2 operator was used where only cases of $0 \rightarrow 1$ errors were allowed. New generations are obtained by using the same *crossover* operation as described in [17].

IV. Experimental Results

To evaluate the effectiveness of EA methods as applied to approximate FSM EC, experiments were conducted where, during the construction of a monolithic TR BDD, pruning was invoked. It should be noted that the purpose of these experiments is to provide results concerning the behavior of the TR BDDs after application of the EA. A production CAD tool would necessarily implement these methods in conjunction with reduced TR computations using co-factors and other well-known techniques.

In this scenario, a naïve approach was taken in constructing the monolithic TR where each next-state BDD was sequentially ANDed together in no particular order and the pruning algorithm was invoked. At each instance of pruning, the transition relation was reduced to a maximum of 95% of its original size (for many cases the reduction was

greater than 95%) resulting in an over-approximation scheme for reachability analysis.

The approximate EA results are compared to exact results in Table 1. The column labeled CIRCUITS contains the names of some ISCAS89 benchmark circuits, the three columns labeled SIZE, ITER. and STATES under the EXACT heading indicate the original size of the monolithic transition relation BDD (in number of vertices), the number of image computation iterations to convergence and the total number of states in the state space, respectively. Corresponding information is provided for the approximation case headed by label EA.

TABLE I
EXPERIMENTAL RESULTS WITH EA PRUNED MONOLITHIC TRANSITION RELATION BDD

CIRCUIT NAME	SIZE	EXACT ITER.	STATES	SIZE	EA ITER.	STATES
s27	3	3	6	3	3	6
s208.1	1	256	256	1	128	256
s298	59	19	218	59	19	218
s344	590	7	2625	768	7	2625
s382	123	151	8865	232	142	22649
s386	11	8	13	11	8	13
s400	94	151	8865	206	32	10845
s420.1	1	65536	65536	1	7168	65536
s444	126	151	8865	1	7168	65536
s510	7	47	47	7	47	47
s526	159	151	8868	138	90	224456
s526n	159	151	8868	142	151	21232
s641	79	7	1544	118	7	6501
s713	82	7	1544	191	7	15226
s820	9	11	25	9	11	25
s832	9	11	25	9	11	25
s953	634	11	504	649	8	299336
s1488	10	22	48	10	22	48
s1494	10	22	48	10	16	48

The significance of these results is that in many cases the approximated results are identical to the exact results. In several instances this occurred with fewer image computations such as the case for the benchmark s420.1. These results indicate that EA approximations can be used successfully and provide motivation for finding alternative genetic operators that can yield better approximate results.

V. CONCLUSIONS

The experimental results yield exact agreements in some cases with others that differ significantly. Since no attention was given to the order in which the TR was constructed, more information could be used enhance the approximations. Future efforts will include the optimization of the pruning BDD EA operators and the investigation of using EAs to produce a reduced transition relation BDD by picking the best sub-BDDs to AND together. Since a TR can be replaced (in a single iteration) by a generalized co-factor, if the generalized co-factor is smaller it becomes the reduced transition relation. Furthermore, the reduced transition relation can be produced as the conjunction of the co-factored sub-relations. Thus, the problem becomes one of determining a good co-factor. We plan to incorporate EA methods for this selection in addition to TR pruning.

ACKNOWLEDGMENT

The authors would like to thank Wolfgang Günther for providing a modified version of the software used in [17].

REFERENCES

[1] S.-Y. Huang and K.-T. Chen, *Formal Equivalence Checking and Design Debugging*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.

[2] K. L. McMillan, *Symbolic Model Checking: An Approach to the State Explosion Problem*, Kluwer Academic Publishers, Boston/Dordrecht/ London, 1993.

[3] E. M. Clarke, O. Grumberg and D. A. Peled, *Model Checking*, MIT Press, Cambridge, Massachusetts, 1999.

[4] H. Touati, H. Savoj and B. Lin, "Implicit enumeration of finite state machines using BDD's", in Proceedings of the International Conference on Computer Aided Design, November 1990, pp. 130-133.

[5] O. Coudert and J. C. Madre, "A unified framework for the formal verification of sequential circuits," in Proceedings of the International Conference on Computer Aided Design, November 1993, pp. 126-129.

[6] G. Cabodi, P. Camurati and S. Quer, "Improved reachability analysis of large finite state machines," in Proceedings of the International Conference on Computer Aided Design, November 1996, pp. 354-360.

[7] G. D. Hachtel, E. Macii, A. Pardo and F. Somenzi, "Symbolic algorithms to calculate steady-state probabilities of a finite state machine," in Proceedings of the European Design Automation Conference, February 1994, pp. 214-218.

[8] A. Xie and P. Beerel, "Implicit enumeration of strongly connected components," in Proceedings of the International Conference on Computer Aided Design, November 1999, pp. 37-40.

[9] S. G. Govindaraju, D. L. Dill, A. J. Hu and M. A. Horowitz, "Approximate reachability with BDDs using overlapping projections," in Proceedings of the Design Automation Conference, June 1998, pp. 451-456.

[10] I.-H. Moon, J. Kukula, T. Shiple and F. Somenzi, "Least fixpoint approximations for reachability analysis," in Proceedings of the International Conference on Computer Aided Design, November 1999, pp. 41- 44.

[11] A. Aziz, J. Kukula and T. Shiple, "Hybrid verification using saturated simulation," in Proceedings of the Design Automation Conference, June 1998, pp. 615-618.

[12] A. Kuehlmann, K. L. McMillan and R. K. Brayton, "Probabilistic State Space Search," in Proceedings of the International Conference on Computer Aided Design, November 1999, pp. 574-579.

[13] R. Drechsler, *Evolutionary Algorithms in VLSI CAD*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.

[14] R. Drechsler, B. Becker and N. Göckel, "A genetic algorithm for variable ordering of OBDDs," in Notes of the International Workshop on Logic Synthesis, May 1995, pp. 5.55-5.64.

[15] M. A. Thornton, J. P. Williams, R. Drechsler, N. Drechsler and D. Wessels, "SBDD variable reordering based on probabilistic and evolutionary algorithms," in Proceedings of the Pacific Rim Conference on Communications, Computers and Signal Processing, August 1999, pp. 381-387.

[16] V. Kommu and I. Pomeranz, "GAFAP: genetic algorithm for FPGA technology mapping," in Proceedings of the European Design Automation Conference, September 1993, pp. 300-305.

[17] R. Drechsler and W. Günther, "Evolutionary synthesis of multiplexer circuits under hardware constraints," in Proceedings of the Genetic and Evolutionary Computing Conference, July 2000, pp. 513-518.

[18] H. Touati, H. Savoj and B. Lin, "Implicit enumeration of finite state machines using BDD's," in Proceedings of the International Conference on Computer Aided Design, November 1990, pp. 130-133.

[19] O. Coudert and J. C. Madre, "A unified framework for the formal verification of sequential circuits," in Proceedings of the International Conference on Computer Aided Design, November 1993, pp. 126-129.