TEST VECTOR GENERATION AND CLASSIFICATION USING FSM TRAVERSALS

Ralph Marczynski, Mitchell A. Thornton, Stephen A. Szygenda
Department of Computer Science and Engineering
Southern Methodist University

ABSTRACT

Design correctness has become a bottleneck in the modern digital system design cycle. In an effort to improve current ad hoc simulation processes, this paper presents a method for the automated generation of simulation vectors using Symbolic FSM Traversal techniques. Generated vectors are classified into three categories, Forward Inter-Frontier, Reverse Inter-Frontier, and Intra-Frontier vectors; a classification based on a vector's ability to generate Forward-, Reverse-, and Inter-Frontier transitions in an FSM's state transition graph. Additionally, a State-Element Transition Relation (S-ETR) is introduced. This technique involves the construction of a Transition Relation (TR) for each state holding element and defining a smaller, incomplete, over-approximation of the TR. Combining the information present in the S-ETRs coupled with simulation is used to perform image computations.

1. INTRODUCTION

Functional verification is the process of determining some level of confidence that a design meets its specifications. One trend is the partial validation of designs through simulation [1], consisting of sending input to the design under test and observing the output. Often the design is judged by the equivalence found when comparing the output with that of some reference model or other abstraction of the circuit when it is exposed to the same stimulus. This approach is often used in the commercial environment; unfortunately this in not due to the method's effectiveness, but rather the lack of superior options.

The shortcoming of a simulation-based verification methodology is the limited functional coverage. Even for modest designs, the number of vectors needed to for an exhaustive test is too large to simulate in a feasible amount of time. In order to utilize simulation, one needs to choose the test vectors carefully as to gain the maximum amount of coverage and ensure tests for rare "corner-cases". Automated methods for such intelligent vector generation are the saving grace of simulation, but this area still requires further study to reach its full potential.

Formal methods provide alternatives that are potentially capable of overcoming the obstacles faced in simulation. A formal method is the application of mathematical methodologies to the specification and verification of systems [2]. The key advantage is the ability to exhaustively test the design with respect to its specification. To handle complex designs, symbolic BDD-based system representations [3,4,5] can be employed to allow for a compact representation of the state space and symbolic FSM traversal. With many mature to relatively mature technologies plagued with computational obstacles and the need for human intervention; a workable, practical, and effective verification flow appears to be rooted in the integration of multiple verification methods [6,7,8].

This paper addresses the integration and reuse of existing methods. Symbolic FSM traversal is used to generate test vectors, which are classified into three categories:

1.) *Forward Inter-Frontier Vectors:* vectors which will cause a transition from a state in frontier i to a state in frontier $i+1$.
2.) *Reverse Inter-Frontier Vectors:* vectors which will cause a transition from Frontier i to a state in frontier j, where $j < i$.
3.) *Intra-Frontier Vectors:* vectors that will cause transitions among states within a given frontier, including vectors causing self-loops.

Such a vector classification can be exploited during simulation. Vectors tracing a minimum length path between a given pair of a state's frontiers can be formed and statements about reachability can be made if no paths can be found. In the presence of temporal logic assertions, vectors with the greatest probability of cycle generation can be used to simulate for the failures of *eventuality* properties, as can Inter-Frontier transition vectors be used to simulate for the reachability of a fail state, necessary for validation of safety properties. In a guided search of the state space, Inter-Frontier communication complexity can also be used as heuristic for selecting further simulation origin states.

The remainder of this paper is organized as follows. Section 2 provides the necessary background concerning Symbolic FSM Traversal. Section 3 outlines the vector generation and grouping procedures, with Section 4 identifying its applications. Section 5 introduces *State-Element Transition Relations* (S-ETR) and their usefulness within a verification system integrating simulation with formal methods. Finally Section 6 provides concluding remarks.

2. BACKGROUND – FSM TRAVERSAL

FSM traversal is a traversal of a system's state transition graph (STG) in either a breath-first or depth-first manner. Starting at an initial state, typically the reset state, the search iterates through all nodes in the STG resulting in a set of all reachable states. The knowledge gained from such a procedure proves useful in many EDA-CAD settings.

An implicit STG representation is achieved through the construction of a *Transition Relation* (TR) [5], a function representing all possible transitions within a FSM. FSM

Traversal is then carried out as a series of *Image Computations*; a conjunction of the TR and all previously reached states with existential quantification of the resultant BDD over the present state variables [9]. The result of each iteration is then the set of states on the frontier (a set of next states), which are further re-labeled as present states and added to the set of reached states. The process is repeated until an evaluated frontier does not provide transitions into undiscovered states.

The BDD for the TR of complex systems can exceed memory limitations. The inputs to a system are not necessary for the image computation, therefore commonly used TRs differ from the one described in this paper in that inputs are removed through existential quantification, resulting in the *smoothed TR*. Smoothing can take place at different stages of TR construction, depending on the method [5,9]. Vector generation, however, requires the knowledge of input values with respect to state transitions; hence, the sub-optimal *monolithic* TR without smoothing is required. Discussion of a new representation of the TR, the S-ETR, is presented in Section 5, which significantly reduces the monolithic TR size through over-approximation, while retaining the input vector information.

3. TEST VECTOR GENERATION

At each image computation, test vectors for all transitions are determined and grouped into the three categories described in the Introduction. This classification enables the selection of vectors in order to meet specified coverage criteria (e.g. cycles, inter-frontier paths, heuristics based on complexity, etc.). In addition to the three classes of vectors, *State Entry Vectors* are determined; the only vectors which can possibly cause entry into a given state.

3.1. State Entry Vectors

For each reachable state, the TR can be manipulated in a straightforward manner to extract all the possible vectors that may lead to an entry into that state. The procedure is simply an existential quantification of the TR over the present state variables. The state entry vectors for the example FSM depicted in Figure 1 are presented in Table 1.

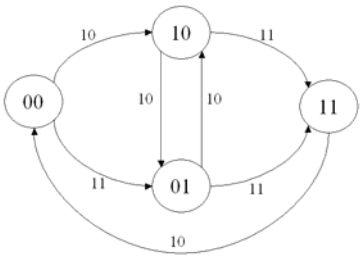
3.2. Forward Inter-Frontier Vectors

At each image computation, the vectors capable of causing entry into the new frontier are evaluated. The initial frontier R_0 is assumed to be the reset state. Table 2 presents the Forward Inter-Frontier Vectors for the example FSM. The procedure is implemented as follows:

$$R_{i+1} = TR \wedge R_i$$
$$S_{i+1} = \exists\, (PS \wedge IN)\, R_{i+1}$$
$$R_i\, Shifted = Re\text{-}Label\, (R_i, PS, IN)$$
$$R_{i+1}\, NewOnly = R_i\, Shifted \oplus S_{i+1}$$
$$Fwd\text{-}Inter\text{-}Frontier\ Vectors = R_{i+1}\, NewOnly \wedge R_{i+1}$$

3.3. Reverse Inter-Frontier Vectors

During a given image computation, the newly reached states on the frontier may contain states capable of a transition to a state



PS0	PS1	NS0	NS1	IN0	IN1
0	0	1	0	1	0
0	0	0	1	1	1
1	0	0	1	1	0
1	0	1	1	1	1
0	1	1	0	1	0
0	1	1	1	1	1
1	1	0	0	1	0

Figure 1: FSM and Corresponding Transition Relation

TABLE 1: STATE ENTRY VECTORS

State	Entry Vector(s)
00	0-
00	10
01	- -
10	0-
10	10
11	0-
11	11

TABLE 2: FORWARD INTER-FRONTIER VECTORS

	Source State	Target State	Vector
Frontier 1	00	01	11
	00	10	10
Frontier 2	10	11	11
	01	11	11

lying in a previously reached frontier. Extracting such vectors (and classifying them accordingly) allows for additional procedures to generate "cycle" producing vectors. In order for this to be computed, we are forced to perform an image computation originating from only the newly reached states. Reusing the BDD from the Forward Inter-Frontier Vector procedure, the necessary steps are as follows:

$$R_{i+1}\, NewOnly\_Shifted = Re\text{-}Label\, (R_{i+1}\, NewOnly, NS,PS)$$
$$R\_fromNewOnly = R_{i+1}\, NewOnly\_Shifted \wedge TR$$
$$Rev\_\ Inter\text{-}Fronteir\_Vectors = R_i\, Shifted \wedge R\_fromNewOnly$$

The procedure computes Frontier $i+2$, and the union of Frontier $i+2$ with Frontier(s) j, with $j < i+1$, determines the Reverse Inter-Frontier Vectors. The only Inter-Frontier vector for the example FSM is (10) corresponding to source state (11) and target state (10).

3.4. Intra-Frontier Vectors

Finally, the set of vectors that cause transitions among states in a single frontier are evaluated. This classification allows for further generation of cyclic sequences and can be used in creating heuristics for guided state space traversal. Reusing the BDD from the previously defined procedures, the evaluation is:

$$\textit{Intra-Frontier Vectors} = R\_\textit{fromNewOnly} \wedge R_{i+1}\ \textit{NewOnly}$$

The resultant set of vectors contains all vectors capable of initiating transitions within a single frontier, including vector resulting in self-loops. Figure 1 of the example FSM does not depict self-loop vectors; their presence is implied as the vectors that do not cause a state-pair transition. The Intra-Frontier Vectors for the example FSM are presented in Table 3.

TABLE 3: INTRA-FRONTIER VECTORS

	Source State	Target State	Vector
Frontier 1	01	01	1-
	10	01	10
	01	10	10
	10	10	0-
Frontier 2	11	11	0-
	11	11	11

4. APPLICATIONS

4.1. Minimum Path Vector Sequence

The grouping of vectors with respect to their role in FSM traversal allows for the efficient generation of vector sequences resulting in minimum paths between any given state pair. If any two states lie in two distinct frontiers, the minimum path between them can be found through manipulation of the Forward and Reverse Inter-Frontier Vectors found in the two state holding and possible intermediate frontiers.

A procedure has been implemented which determines minimum path vectors from reset to a specified reachable state. This involves finding the "deepest" frontier containing the target state (identified when the intersection between the states in the inspected Frontier intersected with the target state is not an empty set) and a traversal toward the reset state by selecting vectors creating the desired path during each iteration. This concept can be extended to any state pair by locating the source state, tracing either the Forward or Reverse Inter-Frontier vectors towards the target state.

If a pair of states lie in the same frontier, a similar procedure can be created considering Intra-Frontier vectors. Finally, if a Frontier containing the target state is not found, the target state can deemed unreachable

4.2. Heuristics for Guided State Space Exploration

Several techniques have been developed which limit the power of model checking to "find bugs" rather than exhaustively prove a system's correctness [10]. These methods rely on heuristics in order to target a portion of the state space that is most likely to contain flaws. Although these methods offer promise they are either dramatically inconsistent or require significant designer input. The information stored by the classes of vectors generated through the techniques presented in this paper can be used as heuristics for guided FSM searches.

The Intra-Frontier Vectors capture the amount of activity among states in a single frontier. If we suspect a frontier to contain an error state and is portrayed to be "highly-active", then that frontier may be marked as a candidate for more intense simulation, utilizing a variety of Intra-Frontier vector sequences. Continued traversal can then be focused on the "most-popular" state (a characteristic additionally obtainable from the Intra-Frontier Vectors) with a propagation of the observed properties into the next "partial" frontier.

5. STATE-ELEMENT TRANSITION RELATION

BDD explosion, in terms of the TR and/or the reachable state space, is one of the greatest obstacles when applying formal verification techniques incorporating FSM traversal. The techniques presented up to this point do little to help this situation. Complete FSM traversal using a sub-optimal TR is required to generate and group the vectors, further limiting the complexity of designs that can be investigated.

To address this limitation, a method for generating TRs local to each state-holding element is presented. Each *state-element TR (S-ETR)* is part of a partitioned TR providing information as to which input vectors cause a single state-holding element to change its value, essentially a bit relation with existential quantification of the present state variables. Such a TR can often be much simpler than a complete TR and our experiments show this to be the case. Table 5 compares the size of the S-ETR with the size of a monolithic TR without smoothing for selected circuits of the ISCAS89 benchmark suite.

Combining the S-ETRs produces an over-approximation of the reachable state space since complete state transition pairs are not enumerated, i.e. next states can be evaluated that do not actually occur in the FSM. Simulation can therefore be employed to determine which of the identified transitions are actually valid by simulating the vectors associated with the transitions. Using simulation to sample the behavior of the FSM has been shown to be efficient in other systems coupling simulation and formal verification methods [9]. The benefit of this approach is the reduction of the TR by incorporating simulation to "fill in" the relationships information missing in the S-ETR. The S-ETR, in turn, identifies the small portion of all possible input combinations that require simulation.

The technique is presented using the example FSM from Figure 1. We compute the S-ETR in a similar manner as the monolithic TR, only we do not perform the step of combining the bit relations to form the complete TR. Before merging wire relations, we trim all information that is not of local concern through the existential quantification of the present state variables. As a result, we obtain the vectors that generate local state element changes, presented in Table 5.

From the BDD representing the information in Table 5, we can extract all the *Inconsistencies* resulting from the over approximation. Clearly, a state exposed to a distinct input will

transition to one and only one state. An *O(n)* procedure, where *n* is the number of cubes in the BDD, can be employed to identify the set of *Inconsistent* transitions, resulting in the set of present-state/next-state vector sets that use simulation to mark them valid or invalid. The procedure simply iterates over all the cubes, performs existential quantification of the next state variable and intersects each cube with the S-ETR. If the intersection results in a cube with more minterms than the original cube itself, the set of transitions is marked as inconsistent. Simulation can then separate which of the inconsistent set are valid transitions.

TABLE 4: S-ETR AND MONOLITH TR BDD SIZES FOR ISCAS89

Bench	S-ETR Size	Monolithic TR
s27	51	17
s208.1	29	40
s298	45	367
s344	612	415
s382	109	343
s386	13	131
s400	109	343
s420.1	53	80
s444	56	559
s510	16	182
s526	59	381
s526n	59	381
s641	1923	3990
s713	2302	4204
s820	138	252
s832	138	239
s953	1147	1023
s1488	8	310

TABLE 5: STATE ELEMENT CHANGE VECTORS

	State Element 0	State Element 1
Change	Vector	Vector
0→1	10	11
	11	10
1→1	01	00
	00	01
	11	11
0→0	00	00
	01	01
	11	10
1→ 0	10	10

One could imagine that an FSM can exist where each element is changed by any possible combination of vectors, leading to a simulation of all possible inputs to determine the next state. The results from experiments using the ISCAS89 benchmark suite show this to not be the typical case. In most cases, the number of possible vectors to cause a single element change was less than 20.

6. CONCLUSION

This paper presented several techniques for the manipulation of monolithic TRs in order to obtain vector sequences that can be used to traverse specified paths in an FSM during simulation. For the ease of such sequence generations, the vectors are grouped into classes, governed by a vector's role during FSM traversal.

Although the generated vectors may prove to be useful in some CAD settings; such as putting stress on assertions during simulation or creating heuristics for guided state searches, the techniques used rely on the ability to represent the TR and reachable state space during symbolic FSM traversal.

To loosen these restrictions, a procedure for deriving a State-Element Transition Relations is introduced. The S-ETR is built without the knowledge of the relationships among states, and it therefore only contains partial information. By coupling symbolic techniques for image computation with simulation, the missing information can be created on-the-fly by simulating state transition candidates produced by an analysis of the S-ETR and identifying them as valid or invalid.

7. REFERENCES

[1] International Technology Roadmap for Semiconductors, 2001 Edition, http://public.itrs.net/Files/2001ITRS/Home.htm.

[2] C. Kern and M. Greenstreet, "Formal Verification in Hardware Design: A Survey," *ACM Transactions on Design Automation of Electronic. Systems*, Vol. 4, April 1999, pp. 123-193.

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

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

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

[6] D. Dill, "What's Between Simulation and Formal Verification?," slides from a presentation by Prof. Dill, Stanford University at DAC 1998.

[7] D. Dill and S. Tasiran, "Simulation meets Formal Verification," slides from a presentation at ICCAD 1999.

[8] E.M Clarke and J.M Wing, "Formal Methods: State of the Art and Future Directions," *Technical Report CMU-CS-96-178*, Carnegie Mellon University, 1996.

[9] Yang, C. H. and D. L. Dill: 1998, "Validation with Guided Search of the State Space", *Proc. of the Design Automation Conf.*

[10] Chen K.C "Memory verification needs fresh approach" http://www.eedesign.com/silicon/OEG20030428S0057, EEDesign, 2003