Efficient Combinational Verification Using BDDs and a Hash Table

Rajarshi Mukherjee† Jawahar Jain‡ Koichiro Takayama‡
Masahiro Fujita‡ Jacob A. Abraham† Donald S. Fussell†

†Computer Engineering Research Center
University of Texas
Austin, TX 78713

‡Fujitsu Laboratories of America
3350 Scott Blvd., Bldg. #34, Santa Clara
CA 95054-3104

Abstract

We propose a novel methodology that combines local BDDs with a hash table for very efficient verification of combinational circuits. The main purpose of this technique is to remove the considerable overhead associated with the case-by-case verification of internal node pairs in typical internal correspondence based verification methods. Two heuristics based on the number of structural levels of circuitry looked at and the total number of nodes in the BDD manager are used to control the BDD sizes and introduce new cutsets based on already found equivalent nodes. We verify the ISCAS85 benchmark circuits and demonstrate significant speedup over existing methods. We also verify several hard industrial circuits and show our superiority in extracting internal equivalences.

1 Introduction

The problem of combinational verification can be substantially simplified by using the internal correspondences that networks often have with their synthesized versions. In this paper we propose a technique for combinational verification that combines Binary Decision Diagrams (BDDs) with a hash table to automatically identify internal equivalent nodes in the network. The remaining candidates for equivalence (identified by random simulation) are verified by composing the XOR of the BDDs of the two nodes. We use the information of the equivalent internal nodes to simplify the problem of verifying the two networks. Verification results on the ISCAS 85 benchmark circuits (redundant against non-redundant and redundant against SIS-optimized) and some industrial circuits (original against synthesized) prove the efficacy of our algorithm.

The rest of the paper is organized as follows. In section 2 we discuss previous research on combinational verification. Our algorithm is described in section 3. Experimental results are presented in section 4. We conclude in section 5.

2 Previous Research

Traditionally, combinational verification has been carried out by building and comparing canonical ROBDDs for the corresponding primary outputs of the two given logic networks. Variable ordering techniques that allow the building of smaller BDDs, thus reducing the memory requirement, have been proposed in [7, 10, 13]. However, for a number of circuits, for instance, the multiplier, OBDDs have a size exponential in the number of input variables for all variable orderings.

Berman *et al.* [2] proposed a technique to use internal equivalences in order to establish the functional equivalence of two networks. However, this method was plagued with the problem of *false-negatives* (two nodes can be declared to be inequivalent, when they are really equivalent). Cerny and Mauras [5] introduced the use of *cross-controllability* and *cross-observability* that exist among the internal nodes on two appropriate cutsets in the two given networks. Brand [3] proposed an ATPG based technique to determine equivalences between internal nodes in two given networks. Another verification technique based on internal variables has been proposed in [6]. Several learning based techniques, [9, 11, 8, 12], have been proposed in the recent years. But, learning based techniques are unable to derive *all* internal equivalences using limited computational resources. Matsunaga [16] has proposed an efficient method based on local BDDs and novel cutset manipulation techniques. Here, potential candidates for functional equivalence are verified using BDDs built using cutsets of internal nodes in the network.

3 Our Verification Algorithm

Our verification algorithm has the following three phases:**(1)** Random pattern simulation **(2)** Building and hashing of internal BDDs **(3)** False-negative elimination.

The flow diagram of the algorithm is shown in Fig. 1. First, the two networks are joined together by connecting their corresponding primary inputs to create a *composite network*. Next, we describe each phase in detail.

(1) Random pattern simulation: Each node is assigned a bit vector of length k (set by user; we chose 32). The bit vectors store the results of a parallel simulation at each node and act as signatures for the nodes.

0-7803-3583-X/97 $10.00 ©1997 IEEE

The nodes are hashed into a hash table **Cand-hash**, using their bit-vectors as keys. Nodes with identical-valued bit-vectors hash to the same collision chain and constitute a set of candidates for functional equivalence.

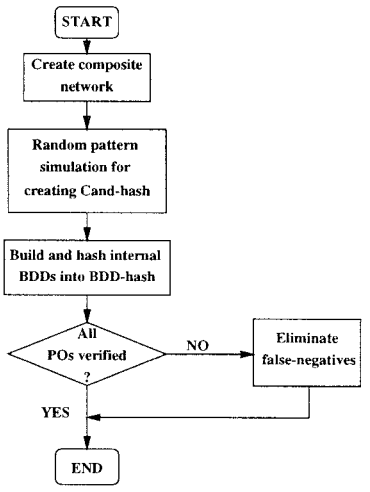

Figure 1: *Flow Diagram of the Verification Algorithm*

(2) Building and hashing of internal BDDs: In this phase the primary inputs of the composite network form the first cutset for building BDDs for the nodes in the network. We assign independent BDD variables to the primary inputs and start building BDDs for the internal nodes in the composite network. The nodes are processed in a breadth first order. The nodes are

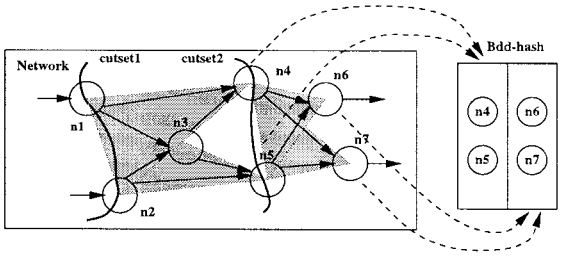

Figure 2: *Hashing of nodes with equivalent internal BDDs*

hashed into a hash table, **BDD-hash**, using the pointer to their BDDs as the key (Fig. 2). Nodes that hash to the same collision chain are functionally equivalent and are grouped together in separate **EQ-lists**. Two heuristics have been developed to control the size of the BDDs built in this phase:

BDDAmanager based size control: Before building the BDD of any node we check the size of the already existing shared BDD data structure. If this size exceeds a preset constant limit K (e.g. $K = 50,000$), we stop building BDDs based on the present cutset and choose a new cutset by a depth first search from the primary outputs. The new cutset consists of a set of gates Γ such that each gate in Γ is either a member of an **EQ-list** or is a primary input.

Structural level based size control: A limit L (we used $L = 5$) is set on the number of structural levels (All primary inputs are assigned a structural level of 0. For all non-primary input nodes g the level $l(g) = \forall h\{max(l(h)) + 1 \mid h \in fanin(g)\}$, where $fanin(g)$ is the set of all nodes that fanin to the node g.) of the network for which BDDs are built using any single cutset. A new cutset is introduced whenever a node is reached which has a structural level that exceeds the limit for the present cutset.

(3) False-negative elimination: Now, we check if all the corresponding primary outputs of the composite network have been verified. If that is the case, the algorithm exits. Otherwise, each pair of candidate nodes (g_i, h_i) in **Cand-hash**, that have not been proved to be equivalent yet, is verified by composing the XOR of their BDDs in terms of successive cutsets that consist of nodes in EQ-lists or primary inputs. Each pair is equivalent if the XOR BDD reduces to a 0.

Some gates that hash to the same collision chain in **Cand-hash** can be proved to be functionally unequal if we analyze their input cone applying the notion of *dominators*. A node g is called a **topological dominator** of a node v in a network with a single primary output F if every path from v to F passes through g. An almost linear-time algorithm to find all topological dominators in any directed acyclic graph has been presented in [15]. A node g in a network C is called a **perfect dominator** if it dominates every node in its transitive fan-in. A node g is called a **functional dominator** of a primary input variable v in a network C with a single primary output F if F can be expressed in terms of some cutset λ such that $g \in \lambda$, and $\{\forall h \in \lambda \mid h \not\equiv g\}$, v is not in the support set of h. Clearly, every topological dominator is also a functional dominator.

After each step of successive compose, dominators can be found in a modified copy of the composite network, obtained by removing from the original network all internal nodes (variables) that disappear from the support set of the BDD during composition. The network is further simplified by removing all nodes that do not fan out to any other node. Dominators found in the modified network are **functional** dominators and cannot be determined by a topological analysis of the original network.

If a node is a functional dominator, and does not represent a Boolean constant, the following theorem can be used to reduce the time and space required for verification.

Theorem 3.1 *If $g \in \lambda_i$ is a perfect functional dominator and is not a Boolean constant then to check for the*

satisfiability of F, while composing $F(\lambda_i)$ in terms of cutset λ_j, g can be treated as a primary input variable.

4 Experimental Results

The verification algorithm has been implemented in C in the SIS-1.2 [14] environment. We have used the BDD package from CMU that incorporates dynamic variable ordering. Our experiments have been conducted on a Sun Sparc-10 workstation equipped with 128 MBytes of memory. We have verified the ISCAS 85 benchmark circuits against their non-redundant MCNC versions. Next, we optimized the ISCAS 85 circuits using SIS and verified each original circuit against its SIS-optimized version. In addition, we have verified two difficult industrial circuits (original against synthesized versions). One of these circuits has over 220 inputs and 2600 complex gates. The other circuit has more than 200 inputs and more than 2000 complex gates. In the circuit **indus2** which has more than 30 structural levels, OBDDs (in terms of primary inputs) exceed the available memory at the 16th structural level. We are able to verify this circuit in less than **3 minutes**.

The verification results are shown in Table 1, where we display our superiority over some learning based techniques. We present the size of the largest BDD built, and the times spent in hashing the local BDDs. We obtain a **948% speedup** over the results of [11] and a **2130% speedup** over the results of [12].

We present in Table 2, the verification results for the ISCAS 85 benchmark circuits when the original circuit is verified against its SIS-optimized copy.

Learning based verification algorithms cannot extract *all* equivalent nodes in a given network using limited computation resources. This is specially true if a higher level of learning is required to derive this information, because most learning algorithms have an exponential time complexity with respect to the number of learning levels. In Table 3, we show that we are much more efficient in extracting internal equivalences from networks. All the equivalences were identified while hashing the local BDDs. From our experiments we conclude that for verification, the information of equivalent nodes is much more important than that of indirect implications. For instance, in c7552 our method extracts 0.5 % more equivalent nodes than the method of [9]. But, this results in a drastic speedup in the time taken to verify this circuit.

5 Conclusions and Future Research

In this paper we have presented a novel combinational verification technique that combines local BDDs with a hash table, thus automating collection of equivalent nodes. The main attractive features of this algorithm are its efficiency both in terms of memory requirement and speed of execution. We have developed two simple, yet effective, heuristics to control the sizes of the BDDs built during verification. In addition, we have developed a successive compose based method to eliminate false-negatives. By the efficient verification of the ISCAS 85 benchmark circuits and some difficult industrial circuits, we have demonstrated the efficacy of our verification technique. As future work, better cutset selection heuristics like [16] will be incorporated.

References

[1] Cook S. A., "The complexity of theorem proving procedures", 3rd ACM SIGACT, pp. 151-158, 1971.

[2] Berman C. L., Trevyllian L. H., "Functional Comparison of Logic Designs for VLSI Circuits", ICCAD, 1989, pp. 456-459.

[3] Brand D., "Verification of Large Synthesized Designs", ICCAD, 1993, pp. 534-537.

[4] Bryant R. E., "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers, vol. C-35, no. 8, Aug. 1986, pp. 667-691.

[5] Cerny E., Mauras C., "Tautology Checking Using Cross-Controllability and Cross- Observability Relations", ICCAD, 1990, pp. 34-38.

[6] Nakaoka T. *et al.*,"A Verification Algorithm for Logic Circuits with Internal Variables", ISCAS 95, pp. 1920-1923.

[7] Fujita M., Fujisawa H., Kawato N., "Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams", ICCAD, 1988, pp. 2-5.

[8] Jain J., Mukherjee R., Fujita M., "Advanced Verification Techniques Based on Learning", 32nd Design Automation Conf., June 1995, pp. 420-426.

[9] Kunz W., "HANNIBAL: An Efficient Tool for Logic Verification Based on Recursive Learning", ICCAD 1993, pp. 538-543.

[10] Malik S. *et al.*, "Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment", ICCAD, 1988, pp. 6-9.

[11] Mukherjee R., Jain J., Fujita M., "VERIFUL : VERIfication using FUnctional Learning", European Design and Test Conf., March 1995, pp. 444-448.

[12] Reddy S., Kunz W., Pradhan D. K., "Novel Verification Framework Combining Structural and OBDD Methods in a Synthesis Environment", 32nd Design Automation Conf., June 1995, pp. 414-419.

[13] Rudell R., "Dynamic Variable Ordering for Ordered Binary Decision Diagrams", ICCAD, 1993, pp. 42-47.

[14] "SIS: A System for Sequential Circuit Synthesis", Report M92/41, University of California, Berkeley, 1992.

[15] "Lengauer, T. and Tarjan, R. "A fast algorithm for finding dominators in a flowgraph", ACM Transactions on Programming Languages, July 1979, vol. 1, pp. 121-141.

[16] Matsunaga Y., "An Efficient Equivalence Checker for Combinational Circuits", DAC 1996.

[17] Mukherjee R. *et al.*,"Observations on Verification Techniques Based on Learning", IWLS 1995.

Ckts	Max BDD size	Hash Time (sec.)	Total Time (sec.)	Time (sec.) [9]	Time (sec.) [11]	Time (sec.) [12]	Time (sec.) OBDDs
c432	9613	1.92	**3.65**	3	0.49	2.2	8.03
c499	1681	1.22	**1.25**	6	1.14	2.17	247.32
c1355	1164	6.05	**6.10**	19	3.50	6.73	815.42
c1908	4532	6.58	**8.08**	26	5.76	14.54	89.64
c2670	8682	26.8	**28.78**	231	48	159.3	119.33
c3540	10283	31.15	**69.87**	2057	365	67.64	914.25
c5315	6849	30.67	**65.6**	797	417	372.8	32.78
c6288	514	9.73	**9.87**	48	24.87	32.74	mem. out
c7552	20185	88.87	**99.75**	4724	1911	5583.3	1581.99
Tot. Time	-	-	**292.95**	7911	2776.73	6241.42	n/a
indus1	58125	820.40	**820.90**	n/a	n/a	n/a	10827.7
indus2	174976	134.60	**135.35**	n/a	n/a	n/a	mem. out

Table 1: *Results of Combinational Verification*

Ckts	max BDD size	time (sec.)
c432 vs. c432.opt	7583	1.32
c499 vs. c499.opt	1713	0.75
c1355 vs. c1355.opt	328	3.08
c1908 vs. c1908.opt	7648	67.83
c2670 vs. c2670.opt	5528	84.33
c3540 vs. c3540.opt	2704	11.45
c5315 vs. c5315.opt	9390	33.85
c7552 vs. c7552.opt	9283	1284.20

Table 2: *Verification of ISCAS 85 circuits against their SIS-optimized copy*

Ckts	# Candidates	# Equiv. (us)	# Equiv. [9]	% improvement (us)
c432	126	**121**	110	**10 %**
c499	184	**176**	112	**57 %**
c1355	496	**486**	328	**48 %**
c1908	462	**437**	424	**3 %**
c2670	739	**666**	496	**34 %**
c3540	909	**773**	778	**-0.6 %**
c5315	1546	**1484**	1267	**17 %**
c6288	2384	**2332**	2332	**0 %**
c7552	2035	**1823**	1814	**0.5 %**
Total	8881	**8928**	7551	**18.2 %**

Table 3: *Detection of Equivalent Nodes*