UDC 621.3.049.771.14:621.372.54

An Efficient Filter-based Approach for Combinational Verification

●Jawahar Jain ●Rajarshi Mukherjee ●Koichiro Takayama

(Manuscript received December 15, 1999)

We have developed a filter-based framework where several fundamentally different techniques can be combined to provide fully automated and efficient heuristic solutions to verification and possibly other NP-complete problems. Such an integrated methodology has been shown to be far more robust and efficient than any single existing technique on a wide variety of circuits.

1. Introduction

The problem of automatic combinational verification or Boolean comparison (BC) can be stated as follows; *Given two Boolean circuits, check whether the corresponding outputs of the two circuits are equivalent for all possible input combinations.*

The existing methods for combinational verification can be broadly classified into two categories. 1) One contains methods based on building and comparing the BDDs of the entire circuits. 2) The other contains methods based on the extraction and use of internal correspondences using a combination of structural and functional techniques.[1),2),5),8)-14)]

Most current methods for combinational verification are based on a single "core" technique such as OBDDs, ATPG, learning etc. We refer to any verification technique as a *core* technique if given enough time, it can verify a particular circuit without the help of another verification technique. We call two core techniques *mutually orthogonal* if there are many circuits where one technique is far more superior than the other and vice-versa. Thus, exhaustive simulation, OBDDs, functional partitioning, resynthesis/learning based methods, *BMDs,[4)] or the use of ATPG in the framework of Ref. 2) are core techniques which are orthogonal. For example, ATPG can efficiently verify erroneous multipliers, whereas *BMDs are very efficient when the design is correct. Also, by the *orthogonality* of two techniques we do not imply that there are no circuits on which both techniques are equally effective. Due to the NP-hard nature of the verification problem it is expected that a single core technique will not perform well on a wide range of circuits. In addition, the nature of any given verification problem is usually not known a-priori. This uncertainty is further aggravated in internal correspondence-based verification techniques where the entire verification problem is broken down into many other verification problems. Thus, a verification program that uses only a single core technique, or a very small number of such techniques (especially if improperly combined), cannot be expected to be very robust.

2. Characteristics of an efficient BC scheme

An efficient verification methodology should have the following characteristics. 1) It should have good performance on a wide variety of circuits. 2) It must be robust in memory usage. 3) It

must be modular and extensible.

To achieve these objectives we propose a *filter*-based combinational verification methodology. The *filter* approach is a combination of communicating techniques where each technique calculates (*filters out*) the information it is most suited for, alters the circuit accordingly, and passes (sieves) its results to the subsequent techniques (*filters*). Typically, easier cases of verification are handled first with fast, low-cost filters, followed by more complex and expensive filters that have a higher time and space complexity. A set of filters can also use partial results of each other to solve a verification instance more efficiently. For example, BDDs can be used to prune a large portion of the search space for an ATPG-based filter, thus making the latter more effective. Such interaction of two or more filters could result in a verification instance being verified more efficiently than by any filter alone.

The goal of the filter configuration is to systematically integrate various orthogonal verification techniques and, thus, minimize the time and space resources used by each technique. To pass a verification instance to a subsequent technique, we use three criteria; a) runtime bound, b) memory usage bound and, c) based on information extracted from the circuit.

The key contributions of this paper are as follows. 1) We have developed a filter configuration for very efficient Boolean comparison. 2) We have explained the need for such a configuration and the reason for its efficiency and robustness. 3) Several specialized techniques such as BDD-hash and partitioning have been developed to further enhance the performance of the verifier and make very difficult verification problems more tractable. We demonstrate the efficiency of our BC framework on a large set of industrial designs as well as on the ISCAS 85 benchmark circuits. Many of the industrial circuits could not be verified by several published techniques.

3. Details of a filter-based verifier

The flow diagram of the approach is shown in **Figure 1**. Given a pair of circuits, a specifica-

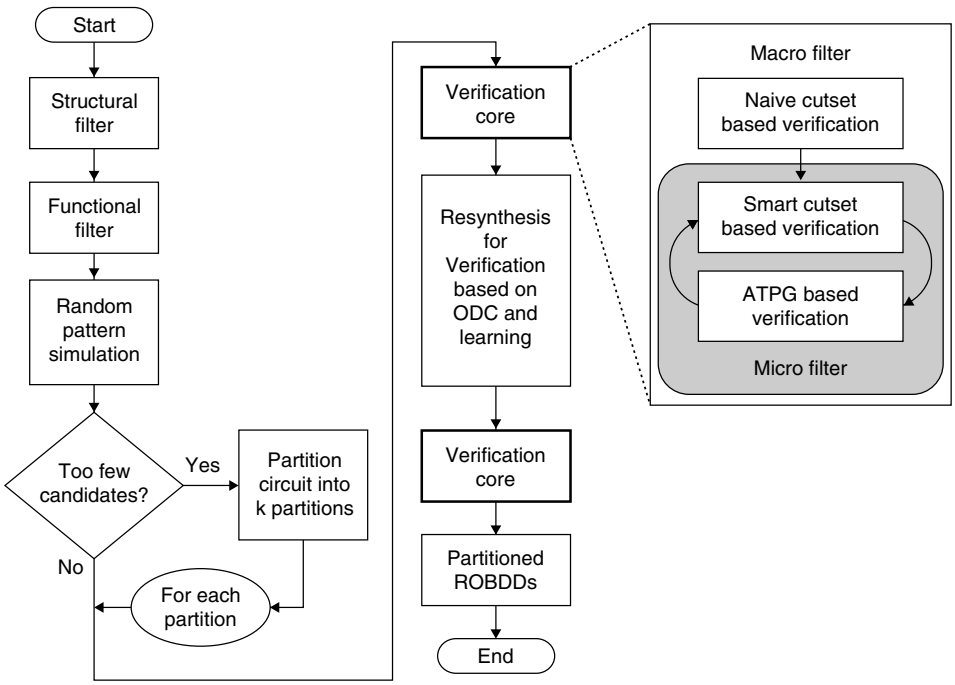

Figure 1
Verification flow diagram.

tion (N_1) and an implementation (N_2), to be verified, a miter circuit (N_C) is created by joining the primary inputs and connecting the corresponding primary outputs by EXOR gates as shown in **Figure 2**. Then, internal equivalence candidates for verification are found in N_C. These candidates are verified with a set of techniques ordered such that the faster/simpler techniques are used before the slower/sophisticated techniques to ensure maximum efficiency. The figure also shows how solving easy candidates earlier improves the efficiency of the verifier drastically. In the figure, let pairs of nodes, (E_1, E_1'), (H_1, H_1') and (H_2, H_2') be candidates where (E_1, E_1') is an easy problem, and (H_1, H_1') and (H_2, H_2') are hard problems. We have found that by finding (E_1, E_1') first with a simple technique, we can avoid spending many time to verify a pair (H_1, H_1') and the process gets speeded-up through experiments.

3.1 Early low-cost filters

The following low-cost and simple filters are the first two filters used in the proposed filter configuration. 1) **Subgraph isomorphism-based structural filter** (SIF): this filter identifies and merges structurally isomorphic parts of two circuits using subgraph-isomorphism. 2) **BDD hashing-based functional filter** (BHF): this filter[9),13)] builds small BDDs for internal nodes in terms of cutsets of other internal nodes and hashes the nodes based on their BDDs to identify and merge functionally equivalent nodes.

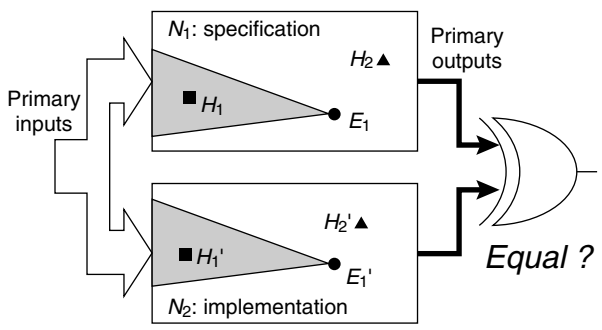

Figure 2
A miter circuit and simplifying effect of early filters.

3.2 Auxiliary filters

Regression-based random pattern simulation and partitioning are the two auxiliary filters. Random pattern simulation is done to collect candidates for internal correspondences. If the number of candidates is below a certain threshold, or if they are too far away from the primary outputs, the miter is partitioned by assigning Boolean constants to a small set of primary inputs. Partitioning can create new candidates for functional equivalence and new indirect implications in each partition. These help simplify verification.

3.3 Principal filters - verification core

Three filters, 1) **Naive cutset-based BDD** (NCV), 2) **Smart cutset-based BDD** (SCV)[11)] and, 3) **ATPG-based technique** (AV),[2)] constitute the verification core (VC) (macro-filter).

At this stage, any candidate pair that has not yet been verified, can either be functionally equivalent or inequivalent. Since a rigorous simulation has been done, we expect a majority of the remaining candidate pairs to be functionally equivalent. Since BDDs are usually more efficient in proving functional equivalence, NCV is used first. SCV, with an initial size limit on the BDDs, is used if NCV fails to prove equivalence. To prove the equivalence of a candidate pair (n_1, n_2), we first create a BDD representing a function f (n_1, n_2) = $n_1 \oplus n_2$. Then the BDD is successively composed and reordered in terms of a cutset selected by two heuristics, naive and smart method,[8),11)] until we reach the primary inputs, or the function is reduced to a 0, or the size of BDD exceeds the limit. If SCV fails, AV with an initial backtrack limit, is applied. In AV a stuck-at-0 fault at the output of the function f is tested. If the fault is proved to be redundant, two nodes n_1 and n_2 are equivalent. SCV and AV are repeatedly applied (micro-filter) with increasing BDD size limit and increasing backtrack limit respectively, for a fixed number of cycles. The gradual increase in effort spent in the micro-filter allows the application of the appropriate technique to solve each pair of equivalence

candidates. This makes verification very efficient.

Since pairs of corresponding primary outputs are also in candidates, if all the pairs are proved to be equal at the end of the process, two circuits are verified.

3.4 Verifying very hard instances

Each pair of primary outputs that are aborted by the previous stages is verified in the presence of BDD partitioning.[7] Each partition is maintained under a separate BDD manager and is independently reordered to reduce the possibility of a memory explosion. Selective resynthesis is applied to create new correspondences.

4. Experimental results

The proposed algorithm has been implemented in C within the SIS environment, using the CMU BDD package with dynamic reordering, and run on a Sun SPARC 20 with 512 Mbytes RAM. Our test circuits include the combinational parts of various designs from Fujitsu, such as data transfer buffers, data transfer controllers, hardwired models for logic/fault simulation, crossbar switch controllers, and the switching unit of a parallel machine. We have also successfully verified numerous difficult circuits provided by other industrial organizations, including EDA vendors. The sizes of the circuits verified ranged up to 100 K gates. In this section we compare our methodology with several other published techniques. As shown in **Table 1**, our methodology is usually faster than the other techniques and can verify many circuits that Ref. 11) is unable to verify due to a BDD blow-up and functional learning-based techniques[8),12)] are inefficient on due to the high computational cost of extracting implications. Since recursive learning[10)] has similar complexity as functional learning, we expect recursive-learning-based verification to be equally inefficient.

We ran our program with different filter configurations. The result is shown in **Table 2** where

Table 1
Comparing our method with other published techniques (all methods run on SUN Sparc-20).

Circuits	Filter (in SIS)	Ref.11(A) (in SIS)	Ref.11(B) (original) (non-SIS)	Ref.12 (original) (non-SIS)	OBDDs (CUDD)
c432 vs. c432nr	0.40	0.88	0.73	0.49	1.8
c499 vs. c499nr	0.37	1.05	0.84	1.14	46.2
c1355 vs. c1355nr	0.95	4.55	1.67	3.50	155.8
c1908 vs. c1908nr	2.13	5.08	4.32	5.76	8.1
c2670 vs. c2670nr	3.38	7.73	2.90	48	3.0
c3540 vs. c3540nr	12.65	22.03	15.14	365	36.5
c5315 vs. c5315nr	8.32	12.85	10.04	417	5.9
c6288 vs. c6288nr	7.20	44.72	11.85	24.87	unable
c7552 vs. c7552nr	20.78	45.73	17.56	1911	32.8
RC2 vs. RC3	15.43	11.57	10.80	unable	unable
RC2 vs. RC2.opt	1.75	5.18	2.22	unable	unable
RC2 vs. RC2.high.opt	38.45	unable	unable	unable	unable
fsm vs. fsmt	123.35	unable	unable	unable	unable
ut vs. utnr	48.50	unable	unable	unable	unable
mp6288 vs. mp6288diff	58.72	unable	unable	unable	unable
msw vs. msw.new1	52.88	unable	unable	unable	unable
msw vs. msw.new2	51.35	unable	unable	unable	unable
nin vs. nin.new1	44.18	unable	unable	unable	unable
ut vs. ut.new	1073.10	unable	unable	unable	unable

the columns represent (1) our proposed configuration, (2) micro-filter sequence is as ATPG, naive-cut, smart-cut, ATPG, and smart-cut, (3) micro-filter sequence is as ATPG, naive-cut, and smart-cut with no iteration, (4) micro-filter sequence is as naive-cut, smart-cut, and ATPG with no iteration, (5) no fault tested by ATPG under ODC, (6) subgraph isomorphism disabled, and (7) both subgraph isomorphism and BDD-hash-based filters turned off. The results show that several alternative filter configurations can be faster than the proposed configuration on certain circuits. However, the proposed configuration has the best average performance over a large number of circuits among all the viable configurations investigated. This points to the robustness of the proposed configuration.

4.1 Simplifying effect of earlier filters

Verification of easier instances by the early filters can modify the circuit so that the verification of harder instances by the subsequent filters can become disproportionately easier. For example, in verifying *msw* vs. *msw.new1* in Table 1, we find that initial filters (SIF and BHF) consume a total of 4.6 seconds in processing/modifying the circuit. Later, when ATPG is invoked on the modified circuit, in 32 seconds it identifies 20 particular pairs of gates as equivalent. On the other hand, without the early filters, ATPG took 48 seconds on the same 20 verification instances. Similarly, the BDD time in the NCV, and SCV filters increased from 9 to 22 seconds, and also larger BDDs were required.

4.2 Economy of the filter process

Table 3 presents detailed results for verifying *c432* against its heavily optimized version. The results show that our arrangement of the filters ensures that the *time per instance*[note 1)] (TPI) for

Table 2
Runtime comparison for different filter configuration.

Circuits	(1)	(2)	(3)	(4)	(5)	(6)	(7)
c432 vs. c432nr	0.40	0.47	0.50	0.43	0.38	0.48	0.90
c432 vs. c432.opt1	10.45	17.58	23.82	10.30	10.60	11.80	2.65
c432 vs. c432.opt2	13.78	12.53	15.85	13.53	13.60	15.00	13.75
c499 vs. c499nr	0.37	0.37	0.37	0.38	0.35	0.32	1.13
c1355 vs. c1355nr	0.95	0.90	0.88	0.87	0.92	0.72	4.47
c1908 vs. c1908nr	2.13	1.95	1.85	2.22	2.12	2.08	4.83
c2670 vs. c2670nr	3.38	5.45	5.55	3.37	3.47	2.30	7.67
c3540 vs. c3540nr	12.65	27.53	27.57	69.20	12.23	11.10	22.95
c5315 vs. c5315nr	8.32	35.10	35.13	8.18	8.15	6.00	14.52
c6288 vs. c6288nr	7.20	7.07	7.22	7.05	7.07	5.63	56.05
c7552 vs. c7552nr	20.78	45.48	45.52	20.58	25.80	19.28	41.37
RC2 vs. RC3	15.43	26.20	25.05	13.75	15.62	14.52	15.50
RC2 vs. RC2.opt	1.75	1.60	1.65	1.70	1.63	1.47	11.05
RC2 vs. RC2.high.opt	38.45	87.45	86.35	34.05	43.07	146.68	17.00
fsm vs. fsmt	123.35	263.70	268.68	599.68	123.23	145.65	244.30
ut vs. utnr	48.50	47.80	48.08	42.70	47.90	50.83	1418.32
mp6288 vs. mp6288diff	58.72	1543.80	1543.43	254.45	56.32	53.97	78.93
msw vs. msw.new1	52.88	182.15	183.65	555.32	52.37	69.70	81.10
msw vs. msw.new2	51.35	188.55	194.73	360.28	51.47	65.97	175.13
nin vs. nin.new1	44.18	47.18	46.10	43.65	47.30	40.72	1036.08
ut vs. ut.new	1073.10	5380.70	5262.40	1100.03	1167.32	1037.23	1824.73
TOTAL	1588.12	7923.56	7824.38	3141.72	1690.92	1701.45	5072.43

Table 3
Comparison of time spent in different filters.

Phase	# runs	eq/inv.	ineq.	time	TPI
str-filter	1	11	–	0.07	0.006
BDD-hash	1	13	–	0.17	0.013
(Random sim.)	1	–	0	0.13	–
BDD-based-1 note 1)	19	11	0	7.12	0.37
ATPG-1	2	0	1	1.55	0.77
BDD-based-2 note 2)	1	1	0	4.37	4.37
ATPG-2	0	–	–	–	–

note 1) Naive-cut + Smart-cut (first iteration)
note 2) Smart-cut (second iteration)

each filter in the verifier is usually less than the TPI of its subsequent filters. This implies that the filter framework is usually successful in verifying each verification instance using a filter with lowest computational cost.

4.3 Verification in the very hard domain: Use of partitioning

In all the circuits presented in Table 1 and Table 2 partitioning was not necessary. We discuss next the verification of two difficult circuits (not shown in the tables) on which automatic partitioning had to be invoked. All published techniques available to us and several commercial verifiers failed to verify these circuits.

Case 1:

In these circuits we found that the frontier of candidate nodes (found by simulation) that is nearest to the primary outputs is actually on an average 15 structural levels away from the primary outputs. Thus, both BDDs and ATPG proved ineffective. Partitioning on the input space was applied (7 partitions were created), which reduced the verification time from 23 000 seconds to only 1336 seconds.

Case 2:

Two artificial circuits were created to verify multipliers in Ref. 6). A function $f(x, y)$ is a multiplier if the following relations are satisfied: $f(x, 0) = 0$ and $f(x, y + 1) = f(x, y) + x$. The first condition is easy to check. To verify the second condition two circuits N_1 and N_2 are built for $f(x, y+1)$ and $f(x, y)+ x$ respectively and are verified for equivalence. N_1 and N_2 are intractable to OBDD-based verification. We also found that these two circuits have no internal equivalences. Therefore, the circuits were partitioned to create internal equivalences, after which they could be verified in 481 seconds.

5. Conclusions

We have proposed a fully automated filter-based approach for Boolean comparison where numerous verification techniques are arranged according to their fundamental characteristics. We have presented intuitive explanations and experimental evidence to show that our approach is extremely efficient. Verification results have been presented on the ISCAS 85 benchmark circuits and a large number of industrial circuits. Many of these industrial circuits could not be verified using several available published techniques and popular commercial verification tools. Detailed comparison with several published techniques shows the superiority of our approach.[note 2)] As future work, we plan to investigate the interaction of various filters and use of partial information produced by one filter by other filters in order to improve the performance of the verifier.

References

1) C. L. Berman and L. H. Trevillyan: Functional Comparison of Logic Designs for VLSI Circuits. Proc. of ICCAD, 1989, pp.456-459.
2) D. Brand: Verification of Large Synthesized Designs. Proc. of ICCAD, 1993, pp.534-537.
3) R. E. Bryant: Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Comp*, **C-35**, 8, pp.667-691 (Aug. 1986).

note 1) This is the average runtime required by any given filter per verification instance given to it.

note 2) Non-disclosure agreements do not allow us to publish verification results comparing our approach with commercial verification tools.

4) R. E. Bryant and Y.-A. Chen: Verification of Arithmetic Circuits with Binary Moment Diagrams. Proc. of DAC, 1995, pp.535-541.
5) E. Cerny and C. Mauras: Tautology Checking Using Cross-Controllability and Cross-Observability Relations. Proc. of ICCAD, 1990, pp.34-37.
6) M. Fujita: Verification of Arithmetic Circuits by comparing two similar Circuits. Proc. of CAV, 1996, pp.159-168.
7) A. Narayan, J. Jain, M. Fujita, and A. Sangiovanni-Vincentelli: Partitioned ROBDDs – A Compact, Canonical and Efficiently Manipulable Representation for Boolean Functions. Proc. of ICCAD, 1996, pp.547-554.
8) J. Jain, R. Mukherjee, and M. Fujita: Advanced Verification Techniques Based on Learning. Proc. of DAC, 1995, pp.420-426.
9) A. Kuehlmann and F. Krohm: Equivalence Checking Using Cuts and Heaps. Proc. of DAC, 1997, pp.263-268.
10) W. Kunz: HANNIBAL: An Efficient Tool for Logic Verification Based on Recursive Learning. Proc. of ICCAD, 1993, pp.538-543.
11) Y. Matsunaga: An Efficient Equivalence Checker for Combinational Circuits. Proc. of DAC, 1996, pp.629-634.
12) R. Mukherjee, J. Jain, and M. Fujita: VERIFUL: VERIfication using FUnctional Learning. Proc. of ED&TC, 1995, pp.444-448.
13) R. Mukherjee, J. Jain, K. Takayama, M. Fujita, J. A. Abraham, and D. S. Fussell: Efficient Combinational Verification Using BDDs and a Hash Table. Proc. of ISCAS, 1997, pp.1025-1028.
14) S. M. Reddy, W. Kunz, and D. K. Pradhan: Novel Verification Framework Combining Structural and OBDD Methods in a Synthesis Environment. Proc. of DAC, 1995, pp.414-419.

Jawahar Jain received the Ph.D. degree in Electrical and Computer Engineering from the University of Texas, Austine in 1993. He joined Fujitsu Laboratories of America, Inc. in 1994. He is currently a Senior Researcher and manages research in the area of verification and test.

Rajarshi Mukherjee received the B. Tech (Hons.) degree in Electronics and Electrical Communication Engineering from the Indian Institute of Technology, Kharagpur, India in 1991, the M.S. degree in Computer Science from the Texas A&M University in 1994, and the Ph.D. degree in Computer Engineering from the University of Texas, Austin in 1996. Since 1997 he has been involved in research and development in Verification, Diagnosis and Error Correction of digital circuits and systems at the Fujitsu Laboratories of America. He is a holder of a U.S. patent and is a member of Sigma Xi.

Koichiro Takayama received the B.E. and M.E. degrees in Electronic Engineering from Osaka University, Osaka, Japan in 1985 and 1987, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1987. His research interests include VLSI CAD systems and design methodologies. He is currently working with Fujitsu Laboratories of America.