# Combinational Equivalence Checking using Boolean Satisfiability and Binary Decision Diagrams

Sherief Reda
Computer & Systems Eng. Dept.
Ain Shams University
Cairo, Egypt

Ashraf Salem
Mentor Graphics Egypt
Cairo, Egypt

## Abstract

*Most recent combinational equivalence checking techniques are based on exploiting circuit similarity. In this paper, we focus on circuits with no internal equivalent nodes or after internal equivalent nodes have been identified and merged. We present a new technique integrating Boolean Satisfiability and Binary Decision Diagrams. The proposed approach is capable of solving verification instances that neither of both techniques was capable to solve. The efficiency of the proposed approach is shown through its application on hard to prove industrial circuits and the ISCAS'85 benchmark circuits.*

## 1 Introduction

With the widespread use of synthesis and optimization tools, the Combinational Equivalence Checking problem (CEC) is considered to be important from both practical and theoretical point of views. Though the problem is NP-complete, most of verification instances can be solved in reasonable space and time resources.

Most recent CEC techniques focus on identifying internal equivalent nodes in order to simplify the miter[1] circuit [1], [2], [3], [4], [5]. Even though these approaches have shown to be quite successful, they often fail due to the following reasons: First, the two circuits under verification might have no structural similarity, i.e. verifying a Wallace multiplier against a functionally equivalent Booth multiplier. Second, due to memory or time restrictions the algorithm fails to identify all the internal equivalent signals. In case of failure, it is suggested that the problem should be forwarded to other BDD or ATPG engines to solve [5]. Another recent approach is Boolean Satisfiability (SAT). SAT has been successfully used as an efficient complete method for solving ATPG and equivalence checking problems [6], [7]. However, the problem is that it sometimes requires tremendous amount of time and backtracks to prove the circuit.

In this paper, we present a new combinational equivalence checking method. The technique addresses circuits with no internal equivalences or after the internal equivalences had been identified and merged. In our approach, the miter circuit is partitioned and the SAT formula is built for a partition of the circuit and BDDs are built for each cutpoint. Using BDDs, we make sure that any satisfying assignment for the SAT formula is also satisfying for the whole circuit. In addition, through simulation and functional learning [8] we are able to extract implications among the cutpoints and add these implications to the SAT formula; thus, reducing the time and backtracks needed to solve the problem. The novelty of the new technique is based on the new integration method for SAT and BDDs, and the use of functional learning to reduce the decision tree of the SAT branch & bound procedure. We show the efficiency of our approach by verifying the ISCAS'85 benchmark circuits and a number of hard to prove industrial circuits.

The organization of the paper is as follows: in section 2 a quick review of the previous related work in the literature is presented. Section 3 presents our methodology for the integration of Binary Decision Diagrams and Boolean Satisfiability. Experimental results of the algorithm are presented in section 4, and finally section 5 presents the concluding remarks.

## 2 Previous Work

CEC techniques can generally be classified into three categories: functional, ATPG and incremental approaches. Functional approaches are based on constructing the Binary Decision Diagrams (BDDs) [9] of the two circuits under verification. The circuits are equivalent if and only if their corresponding BDDs are isomorphic. Since the BDD size is sensitive to its variable ordering; several methods were proposed to represent minimum size BDDs

[1] Given the two circuits under verification, the miter circuit is constructed by joining the two circuits primary inputs, and feeding their primary output into a 2-input XOR gate.

for the circuit domain [10], [11]; however, there are classes of the circuits that do not have good variable ordering [9].

Automatic Test Patten Generation (ATPG) [12] techniques represent another alternative for solving the CEC problem. ATPG approaches prove the equivalence of the two circuits under verification by proving that the stuck-at-0 is a redundant fault for the their miter output. Recently, Boolean Satisfiability (SAT) has enjoyed a widespread as an efficient ATPG technique [6], [7]. SAT based methods construct the Conjunctive Normal Form formula of the miter circuit and find a satisfying assignment for it using the branch & bound procedure.

Incremental approaches are based on reducing the miter circuit through the identification of Internal Equivalent Pairs (IEP) of the circuit nodes. Berman and Trevillyan [1] were the first to propose such approach; however, their method suffered from the problem of false negatives. Reddy et al. [4] suggested using recursive learning to identify IEP and used ATPG techniques to eliminate false negatives. Brand [2] suggested using ATPG techniques for identifying IEPs under the observability don't care set and substituting one node by the other in the miter circuit and thus simplifying it. To eliminate false negatives, Matsunga [3] suggested composing the cutpoints by their incoming functions using BDDs. Kuehlmann and Krohm [5] used a BDD hash table to identify equivalent nodes and suggested building multiple BDDs originating from different cutsets for the same node to decrease the likelihood of false negatives. They also suggested the use of different verification engines as filters where the circuit passes from one filter to the other until it is solved. Functional learning was proposed by [8] in order to extract internal equivalences and implications among the internal circuit nodes.

Recently, a number of SAT+BDD approaches were proposed to solve the problem. Gupta and Ashar [13] integrated SAT and BDDs by partitioning the miter circuit. A single BDD is used to capture the partition of the circuit near the miter output and SAT clauses to capture the rest of the circuit. They proposed the early bounding method to avoid enumerating the solutions at the cutset nodes between the two partitions. Also, SAT and BDDs were integrated in the work of [14]. Their approach is based on identifying IEP using a BDD composition engine to eliminate the false negative problem. However, in order to overcome of a real negative; they suggested representing the circuit by a root BDD in terms of variables representing the primary inputs and cutpoints variables, and BDDs representing the characteristic function of each cutpoint. They proposed using a randomized local search SAT algorithm to find a satisfying assignment that is consistent for all the BDDs.

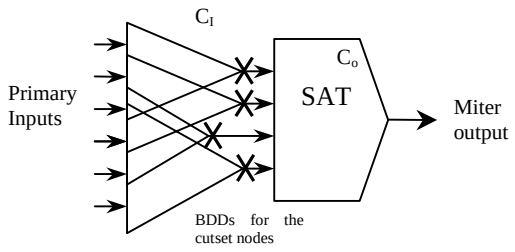

Figure 1: Miter Circuit partitioning

## 3 Integrating SAT and BDDs

Our strategy for solving the verification problem starts by partitioning the miter circuit into two partitions where the two partitions share the cutset nodes. We refer to the partition near the miter output as $C_o$ and other partition by $C_I$ as illustrated in Figure 1. The SAT formula is built for the $C_o$ partition and the BDD of every cutset node is built. It should be noted that the partition $C_I$ acts as a filter that allows the occurring of only certain patterns at the cutset [15]. Let us refer to the set of these patterns by the symbol $\pi$. Also, let us refer to the set of satisfying assignments to the SAT formula of partition $C_o$ by $\eta$. Clearly, to find a pattern at the primary inputs that satisfies the miter output, the resultant set of the intersection of $\pi$ and $\eta$ should not be *NULL*; i.e., $\eta \cap \pi \neq \varnothing$.

A naive approach to find such a satisfying pattern for the miter output is to explicitly enumerate every satisfying pattern for the SAT formula and to check its consistency along the cutset. To check consistency, the corresponding on-set or off-set of the cutset nodes BDDs should be intersected. If the result of the intersection is a non-NULL BDD, this means that there exists a pattern or more at the primary inputs that would satisfy the miter output. Such patterns could be enumerated by tracing the paths of the resultant BDD to the leaf node 1. Clearly, enumerating the satisfying patterns of the SAT formula and checking their consistency is prohibitively expensive. Instead of testing the consistency of all satisfying patterns of the SAT formula, the following technique can greatly reduce the time needed to find a satisfying assignment.

Let us assume that the BDD $\beta$ is initialized to the identity BDD 1, and that every cutset node $n_i$ has the corresponding BDD $\beta_{ni}$. Our method is based on the observation that whenever one of the cutset nodes is assigned in the SAT branch & bound procedure, its BDD should be intersected with $\beta$, then $\beta$ is assigned the new resultant BDD. Thus, if at any time $\beta$ reduces to *NULL*, we immediately conclude a contradiction and consequently backtrack. Let us consider an example.

**Example 1** Suppose that the cutset node $n_3$ is assigned the value *FALSE*. Immediately, we re-calculate $\beta$ as follows: $\beta = \beta \cap \neg\beta_{n3}$. Clearly, $\beta$ represents the resultant intersection BDD of the assigned cutset nodes so far.

Figure 2 shows the modified branch & bound procedure. Since a contradiction could be detected as early as possible, the proposed approach saves the time wasted in the fruitless enumerating of satisfying assignments of the SAT formula.

The proposed approach could be further enhanced by noticing an important observation: *When a NULL BDD occurs, it is not known which assignments caused the conflict*. Let us consider the following example.

**Example 2** Suppose that we have a cutset of five nodes $\{n_2, n_5, n_3, n_7, n_9\}$ and that the following implications exist between the cutset nodes, $n_9$ implies $n_2$ ($n_9 \rightarrow n_2$) and $\neg n_9 \rightarrow \neg n_5$. Assume that during branch & bound the following cutset nodes have been assigned in the following order $n_2 = FALSE$, $n_5 = TRUE$, $n_3 = TRUE$, $n_7 = TRUE$. So far, $\beta$ is not *NULL*. However, with the assignment of a new cutset node $n_9 = TRUE$, $\beta$ reduces to *NULL*. With the earlier proposed approach, we do not know what caused the conflict with the last assignment. Clearly, a mindless branch & bound will keep backtracking on $n_9$ then $n_7$ and $n_3$, until it recognizes that $n_5$ is the source of this problem.

From the previous example, we conclude that a better approach is to analyze the implications among the cutset nodes before the branch & bound procedure starts and add these implications to the SAT formula. More specifically, after the BDDs of the cutset nodes are built, we can use them to deduce such implications (functional learning [8]). For example, if $\neg\beta_{n9} + \beta_{n2} = 1$, the implication $n_9 \rightarrow n_2$ is deduced and it is added to the SAT formula. Thus, an implication extraction phase precedes the branch & bound procedure to deduce all implications among cutset nodes. Since there are four kinds of implications to consider between every pair of cutset nodes, this phase might take some time. In order to reduce the time needed, random bit parallel simulation is used to deduce some of the allowable logic values at the cutset. Analyzing the simulation results has two consequences: First, we can quickly abandon pairs where no implication can be deduced from simulation results. Second, for the rest of the pairs, we can determine what kind of implication might exist. Some optimizations could be made in this phase. For example, if we learned that $n_3 \rightarrow n_4$ and that $n_2 \rightarrow n_3$. There is no need to perform the calculation $\neg\beta_{n2} + \beta_{n4}$. We can immediately conclude by transitivity that $n_2 \rightarrow n_4$.

```
bound (sat_formula φ, literal v, BDD β)
begin
  if v ∈ S then
  begin
      if (v == TRUE) then β = β ∩ BDD(v);
      else β = β ∩ ¬BDD(v);
      if (β == zero_bdd) then return FALSE;
  end
  result = assign v in φ;
  if (result == FALSE) then return FALSE;
  result = process all implications of v;
  return result;
end

branch(sat_formula φ, BDD β)
begin
  choose literal v to split on;
  if v = NULL then return true;
 β_backup = β;
  if bound(φ, v, β) then
     if branch(φ, β) then return true;
  undo v assignment;
  β = β_backup;
  if assign(φ, ¬v, β) then
     if branch(φ, β) then return true;
  return false;
end
```

Figure 2: The modified branch & bound procedure

Figure 3 represents an overall view of the proposed approach. At the start, we try to build the BDD of the miter output by advancing through the cutset levels in a breadth first manner from the primary inputs to the miter output. If at anytime we reach the miter output at level $i_{max}$ without exceeding a certain BDD size limit, the circuit is proved. However, suppose that at level $i+1$ the BDD exceeded the size *LIMIT*, so cut $i$ is retained and the SAT formula is built for the rest of the circuit. Random bit parallel simulation is used to detect possible implications among the cutset nodes. Using BDDs, the extract implications phase verifies the implications among cutset nodes and adds the implications to the SAT formula. After extraction of the implications, the integrated branch & bound procedure is invoked to prove the circuit. The choice of the size *LIMIT* is important because it affects the performance of the implication extraction phase and the branch & bound. Our choice criteria is purely functional and by performing experiments on various circuits, we have found that imposing a *LIMIT* of about 15k nodes on the total number of nodes in the unique table of the BDD manager produces good results.

```
// Top level view of the proposed approach
do
  build BDDs in breadth first for level i nodes;
  i = i + 1;
while (i <= i_max && total_bdd_node_size < LIMIT)
if  i = i_max then circuit proved using BDDs.
else
   - Build SAT formula for the rest of circuit.
   - Use random bit parallel simulation to build list of
     candidate implications along the cutset nodes.
   - Verify the extracted implications from the previous
     step using the BDDs of the cutset nodes.
   - Add calculated implications to the SAT formula.
   - Start integrated branch and bound procedure.
```

Figure 3: A top level view of the proposed approach

## 4 Experimental Results

Our verification framework is implemented in C, using TEGUS [7] as a SAT solver, and the Colorado University CUDD BDD package [17] using a Sun Ultra 10 with 256 MB memory. In order to examine the efficiency of the proposed approach, we used it to verify the ISCAS-85 benchmark circuits against their both non-redundant versions and against their optimized versions by SIS script.rugged [18]. We also tested our methodology by proving a number of large industrial circuits. In our experiments, we have used the Iterated Global Implications (IGI) engine proposed in [7] as a preprocessing step before the start of our proposed approach. The IGI engine proved helpful in proving circuit outputs that are easily solvable by SAT.

Table 1 illustrates the verification time needed to verify the ISCAS-85 circuits against their non-redundant versions and their optimized versions by SIS script.rugged. Column 1 shows the miter circuit name. Column 2 shows the number of mitered outputs in the miter circuit. Column 3 shows the number of outputs proved by the IGI engine. Column 4 shows the number of outputs proved by the BDD engine, that is the BDD construction method managed to reach the miter output without exceeding the maximum BDD size limit. Column 5 illustrates the number of outputs solved by using the integrated proposed method. Column 6 shows the number of unsolved outputs. Column 7 shows the total verification time needed. Column 8 to 12 of the table have the same descriptions as those from 3 to 7 but this time for verification of the circuit against its optimized version by script.rugged.

From the table of results, one verification instance concerns us and needs further illumination. Without identification of internal equivalences, the verification of the c6288 (a 16x16 bit multiplier) against its optimized version is a hard verification problem because neither SAT nor BDD techniques are capable of proving further than the $10^{th}$ output. To the best of our knowledge, we are the first to prove the whole c6288 against its optimized version without relying on the identification of the internal correspondences.

In a second series of experiments, we consider large industrial circuits. These circuits were presented to Mentor Graphics from different customers. A modified version of these circuits with several thousand inputs and outputs, and up to nearly 100,000 gates represent an ideal benchmark for our approach for three reasons: First, we were given the miter version of the circuits under verification. Second, a substantial proportion of the circuits internal equivalent pairs had already been identified and merged. Third, in each circuit there are many outputs where BDDs and SAT fail to prove. Thus, these circuits represent an ideal case study for our approach. Table 2 illustrates the time needed to prove these circuits. Column 1 shows the miter circuit name. Columns 2, 3 and 4 give the number of primary inputs, primary outputs, and gates respectively. Column 5 gives the number of outputs solved by the IGI SAT engine. Column 6 gives the number of outputs solved by the BDD engine before the size limit is reached. Column 7 gives the number of outputs solved by the proposed integrated SAT BDD engine. Column 8 gives the number of unsolved outputs. Column 9 gives the total time needed to prove the circuit. From the table of results, we conclude that the proposed approach managed to prove the hardest outputs in the industrial circuits since neither SAT nor BDDs were capable of solving them. It is also important to note that even if the BDD or SAT engine were given tremendous amount of memory and time, they are still incapable of solving most of the outputs proved by our approach.

## 5 Conclusion

In this paper, we presented a new approach for the combinational equivalence checking problem. While most of the existing techniques in the literature rely on proving the equivalence of circuits using internal equivalences, the presented technique address circuits with no internal equivalences or circuits where internal equivalences have been identified and the merged. The contribution of the paper is in the new integration method for SAT and BDDs, and the use of functional learning to reduce the

| Miter Circuit[1] | # Outputs[2] | Verification against Non Redundant version | | | | | Verification against Optimized version | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | # by IGI SAT[3] | # by BDDs[4] | # by SAT+BDD [5] | Unsolved outputs[6] | Total Time (seconds) [7] | # by IGI SAT[8] | # by BDDs[9] | # by SAT +BDD[10] | Unsolved outputs[11] | Total Time (seconds) [12] |
| C0432 | 7 | 7 | 0 | 0 | 0 | 0.65 | 0 | 7 | 0 | 0 | 1.47 |
| C0499 | 32 | **32** | **0** | 0 | **0** | 1.77 | **0** | **32** | 0 | **0** | 3.6 |
| C0880 | 26 | **26** | **0** | 0 | **0** | 0.19 | **15** | **11** | 0 | **0** | 2.72 |
| C1355 | 32 | 32 | 0 | 0 | 0 | 3.16 | 0 | 32 | 0 | 0 | 6.95 |
| C1908 | 25 | 25 | 0 | 0 | 0 | 4.27 | 0 | 25 | 0 | 0 | 10.2 |
| C3540 | 22 | 11 | 11 | 0 | 0 | 87.2 | 3 | 19 | 0 | 0 | 72.9 |
| C5315 | 123 | 117 | 5 | 0 | 0 | 22.4 | 61 | 62 | 0 | 0 | 8.67 |
| C6288 | 32 | **32** | **0** | 0 | **0** | 61.2 | 4 | 6 | 22 | 0 | 2582 |
| C7552 | 108 | **107** | **1** | 0 | **0** | 16.7 | 82 | 26 | 0 | 0 | 31.8 |

Table 1: Results on ISCAS'85 benchmark circuits

| Circuit name[1] | # PI[2] | # PO[3] | # Gates[4] | # by IGI SAT[5] | # by BDDs[6] | # by SAT + BDDs[7] | # Unsolved[8] | Total Time[9] (seconds) |
|---|---|---|---|---|---|---|---|---|
| miter01 | 2381 | 29 | 45214 | 0 | 2 | 27 | 0 | 2398 |
| miter02 | 866 | 925 | 26215 | 508 | 192 | 225 | 0 | 35941 |
| miter03 | 444 | 403 | 12057 | 129 | 238 | 36 | 0 | 5818 |
| miter04 | 4231 | 2553 | 93199 | 1254 | 67 | 1232 | 0 | 78838 |

Table 2: Industrial circuits results

search tree of the SAT branch & bound procedure. The efficiency of the checker has been shown through its application on the ISCAS'85 benchmark circuits and hard to prove industrial circuits.

## References


[1] C. Leonard Berman, Louise H. Trevillyan. "Functional Comparison of Logic Circuits for VLSI Circuits," In Proceedings of International Conference on Computer-aided Design, Nov. 1989.

[2] D. Brand, "Verification of Large Synthesized Designs," In Proceedings of International Conference on Computer-aided Design, 1993.

[3] Y. Matsunaga, "An Efficient Equivalence Checker for Combinational Circuits," 33rd IEEE/ACM Design Automation Conference, 1996.

[4] S. M. Reddy, W. Kunz, D. K. Pradhan, "Novel Verification Framework combining Structural and OBDD Methods in a Synthesis Environment," 32nd ACM/IEEE Design Automation Conference, June 1995.

[5] A. Kuehlmann and F. Krohm, "Equivalence Checking Using Cuts an Heaps," In Proc. of 34th ACM/IEEE Design Automation Conference, 1997.

[6] T. Larrabee, "Test Pattern generation Using Boolean Satisfiability," IEEE Trans. on Computer Aided Design, Vol. 11, No. 1, January 1992.

[7] P. Stephan, Robert K. Brayton and Alberto L. Sangiovanni-Vincentelli, "Combinational Test generation Using Boolean Satisfiability," IEEE Trans. Computer Aided-Design of Integrated Circuits and Systems, Vol. 15, No. 9, September 1996.

[8] J. Jain, R. Mukherjee, M. Fujita, "Advanced Verification techniques based on learning," 32nd ACM/IEEE Design Automation Conference, June 1995.

[9] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Trans. on Computers, Vol. C-35, No. 8, August 1986.

[10] S. Malik, A. Wang, R. Brayton, A. Sangiovanni-Vincentelli, "Logic verification using Binary Decision Diagrams in a Logic Synthesis Environment," Int. Conf. On Computer Aided Design, 1988.

[11] R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams," In Proc. of Int. Conf. On Computer Aided Design, 1993.

[12] M. Abramovici, M. A. Breuer, A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.

[13] A. Gupta and P. Ashar, "Integrating Boolean Satisfiability Checker and BDDs for Combinational equivalence Checking," In Proc. Intl. Conf. On VLSI Design, Chennai, India 1998.

[14] J. Burch, V. Singhal. "Tight Integration of Combinational Verification Methods," International Conference on Computer Aided Design, 1998.

[15] E. Cenry and C. Mauras, "Tautology Checking using cross-controllability and cross-observability relations," In Proc. of IEEE International Conference on Computer Aided Design, 1990.

[16] M. Schulz, E. Trischler and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test pattern Generation System," IEEE Trans. on Computer Aided Design, Vol. 7, No. 1, January 1988.

[17] F. Somenzi, CUDD: Colorado University Decision Diagram Package. ftp://vlsi.colorado.edu/pub.

[18] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli. "Sequential circuit design synthesis and optimization," In Proc. Int. Conf. on Computer Design, 1992.