Using SAT for Combinational Equivalence Checking[*]

Evgueni I. Goldberg
Cadence Berkeley Laboratories
Cadence Design Systems
egold@cadence.com

Mukul R. Prasad, Robert K. Brayton
Department of Electrical Engineering & Computer Sciences
University of California, Berkeley
{mukul,brayton}@eecs.berkeley.edu

Abstract

This paper addresses the problem of combinational equivalence checking (CEC) which forms one of the key components of the current verification methodology for digital systems. A number of recently proposed BDD based approaches have met with considerable success in this area. However, the growing gap between the capability of current solvers and the complexity of verification instances necessitates the exploration of alternative, better solutions. This paper revisits the application of Satisfiability (SAT) algorithms to the combinational equivalence checking (CEC) problem. We argue that SAT is a more robust and flexible engine of Boolean reasoning for the CEC application than BDDs, which have traditionally been the method of choice. Preliminary results on a simple framework for SAT based CEC show a speedup of up to two orders of magnitude compared to state-of-the-art SAT based methods for CEC and also demonstrate that even with this simple algorithm and untuned prototype implementation it is only moderately slower and sometimes faster than a state-of-the-art BDD based mixed engine commercial CEC tool. While SAT based CEC methods need further research and tuning before they can surpass almost a decade of research in BDD based CEC, the recent progress is very promising and merits continued research.

1 Introduction

Combinational equivalence checking (CEC) is one of the most widely used formal techniques in the verification of digital circuits. While, theoretically the problem is coNP-Hard, practical instances of the problem are more tractable. Current design methodology ensures that the two combinational circuits being checked for equivalence have a fair degree of structural and functional similarity [1]. In recent years several approaches to CEC have been proposed which exploit the above property. While these techniques have significantly advanced the state of the art in CEC, the inherent complexity of the problem and the growing size and complexity of digital systems continues to motivate further research.

Most of the successful programs for CEC use a combination of various engines, with Binary Decision Diagrams (BDDs) [3] as the main workhorse. Although a few of the proposed approaches use Boolean Satisfiability (SAT) [14] or SAT-like engines (viz. ATPG methods [2], recursive learning [11]) as the principal engine, these methods have not been popular. Consequently, the use of SAT in current CEC is largely ancillary to BDDs; e.g. it is used to eliminate false negatives or to choose candidate pairs for deducing intermediate relationships [4].

This paper makes a case for the use of SAT methods in CEC. There are several reasons for pursuing this line of research. First, there have been significant advances in SAT algorithms [15, 19]. Second, while it has been claimed that BDDs are relatively more efficient for CEC, neither has a quantitative comparison been published nor the reasons for purported inefficiency of SAT algorithms analyzed in detail. Third, as discussed in more detail in Section 3, SAT algorithms have several inherent features which BDDs lack, that can potentially make them a more flexible and robust core technology for the application under study. This raises the following questions:

- Is the perceived inefficiency of SAT algorithms in CEC a necessary consequence of the use of SAT algorithms per se or is it an artifact of the particular SAT algorithm used and the way it was used in the CEC framework?
- Is it possible to bridge the efficiency gap between SAT based and BDD based CEC tools by using more sophisticated SAT algorithms that are currently available and/or by fine-tuning the implementation of the tool[1]?

[*]This work was supported in part by the California MICRO program and industrial sponsors Cadence, Synopsys and Fujitsu.

[1]It is noteworthy that BDD based tools draw upon over a decade of research in variable ordering and efficient implementation of BDD packages,

This paper addresses these issues. The main contributions of this work can be summarized as follows:

- We present a detailed analysis of the features of SAT algorithms and BDDs in the context of CEC to argue that SAT based algorithms can be a more flexible and robust core technology for this application.

- We present a simple CEC framework drawing from a number of previously proposed CEC methodologies [2, 4, 21] as well as our own insights into applying SAT for CEC. This framework works entirely off SAT algorithms as the core engine.

- We make a direct quantitative comparison between a preliminary implementation of our tool and a state of the art BDD based mixed engine for CEC [4], and assess the performance gap between BDD based and SAT based checkers.

- We offer insights into several avenues for improving the performance of the above SAT based tool to potentially make it even better than the state of the art in BDD based checkers.

Briefly, the experiments reported in Section 5 show that our checker outperforms state-of-the-art SAT based tools by over two orders of magnitude. Moreover, even the current prototype implementation is only moderately slower (a factor of 2-3) and sometimes faster than state of the art BDD-based mixed-engine checkers. This paper is intended as a proof of concept to show how SAT based techniques can effectively remedy the inherent problems associated with BDD based methods. We advocate that once suitably tuned and applied, SAT based techniques can more *actively* complement and even replace BDDs in CEC while significantly advancing the state of the art in this area.

The rest of the paper is organized as follows. Section 2 discusses previous efforts in the areas of BDD based and SAT based CEC. In Section 3 we provide arguments and illustrations to show how SAT based methods can potentially be a more flexible and robust tool for Boolean reasoning in CEC. Section 4 describes our proposed SAT based CEC framework. We present experimental results comparing our method with several existing SAT based CEC tools as well as a state-of-the-art BDD based mixed engine CEC tool in Section 5. Section 6 concludes the paper with a discussion of several avenues for improving the performance of the proposed CEC framework.

2 Previous Approaches

Most of the recently proposed approaches for CEC [1, 2, 4, 10, 11, 14, 16, 17] operate under the following general framework. The similarity between the two circuits is exploited to deduce specific succinct relationships (equivalences, implications, replacability relationship) between internal nodes (called *cutpoints* [10]) of the two circuits being checked for equivalence. Using these relationships the overall equivalence check is performed as a set of smaller equivalence checks. Briefly, a *cutpoint* is an internal node of one circuit that is proven to be functionally related to one or more internal nodes of the other circuit through a specific succinct relationship (usually *equivalence* or *equivalence modulo inversion*). The algorithm proceeds by sweeping the two circuits (or the *miter* [2]) from inputs to outputs, deducing new cutpoints from previously deduced cutpoints, until the primary outputs are proved equivalent or a miscomparing pattern is found. Negatives (either false or true) encountered during this process, as a result of functional constraints between internal circuit nodes, are resolved by attempting to justify them towards the primary inputs.

Overall, this methodology comprises a *Deduction Engine* to derive internal node correspondences and a *Justification Engine* which eliminates false negatives or identifies true negatives. In many of the proposed works on CEC [7, 10, 16], BDDs are used in both the deduction engine as well as the justification engine. Recently, Burch and Singhal [4] have proposed a methodology where BDDs are the primary deduction engine as well as part of the justification engine. Randomized SAT algorithms, modified to work off the BDDs are used to supplement BDD based justification. Even more recently, Paruthi and Kuehlmann [17] have proposed a tighter integration of BDDs and SAT based methods for CEC. They propose using an interleaved combination of BDDs and a SAT solver as the deduction engine. However, BDDs continue to be a major part of their deduction engine. Moreover their method of using the SAT solver in overall flow is fairly orthogonal to our proposed approach.

There have been a few attempts to use SAT based algorithms to perform the entire equivalence check. Brand [2] proposed a cutpoint based methodology based on *replacability relationships* which were derived using an ATPG tool. HANNIBAL [11] used *recursive learning* to derive implications which were then used by an ATPG tool to perform the equivalence check. More recently, Marques-Silva et. al. [13] proposed using a recursive learning based preprocessor to derive equivalence relationships which are subsequently used by a general purpose SAT solver to perform the verification task. While these methods offer an innovative alternative to BDD based methods, they have not become the method of choice for CEC; it is generally believed that SAT based methods are not as efficient as BDD based methods. However, we believe that this is not a necessary consequence of using SAT methods vs. BDDs but rather a result of the specific SAT algorithms used and the way they

as well as highly tuned implementations of CEC packages, while precious little has been done in this respect for SAT in CEC applications.

have been applied in the overall methodology. This work is a preliminary attempt to validate this claim.

A number of other approaches (see [4, 8] for a more detailed survey) addressing the CEC problem have appeared in the literature. However, they are omitted from the above survey since they are not directly relevant to the focus of this paper.

3 SAT Vs. BDDs in CEC

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ denote the two combinational circuits being checked for equivalence. For ease of exposition we assume that both circuits have a single primary output, denoted by o_1 and o_2 respectively, and an identical set of primary inputs, denoted by $I = \{i_1, i_2, \dots i_n\}$. Let ψ denote an *equivalence cut*. Note that an equivalence cut carries a topological interpretation on $\mathcal{C}_1$ and $\mathcal{C}_2$ (it partitions the inputs of $\mathcal{C}_1$ and $\mathcal{C}_2$ from their outputs) as well as a set interpretation (it is a set of variables forming the physical cut in the circuits $\mathcal{C}_1$ and $\mathcal{C}_2$). In the following we use both these interpretations interchangeably.

As described in Section 2, BDDs have been successfully used as the core *equivalence deduction engine* in a number of programs for CEC [4, 10, 16]. The reason for this success is the ability of BDDs to compactly represent the *complete Boolean space* of reasonably large functions (variable support of up to 15-20 variables or even greater).

However, SAT algorithms have an inherent advantage over BDDs in Boolean reasoning, under a given set of constraints. BDDs have no means of performing *Boolean constraint propagation (BCP)*, a feature that is integral to all branch and bound (or DLL [5]) based SAT solvers. Since branching based SAT solvers explore each assignment to the variables of the formula one by one, BCP or "examining the logical consequences of each assignment", is a natural component of such an algorithm. Thus, such an algorithm can actually work with (branch on) only a small portion of the given Boolean formula while still being able to examine the logical consequences of this branching on the remaining variables at a negligible additional expense (the expense of BCP). On the other hand, BDDs work by constructing a representation of the *entire Boolean space* of a specified set of output variables, in terms of a specified set of input variables. The only way to introduce additional variables is to explicitly construct BDDs of those functions as well and connect them to the existing BDDs by some logical operation, *viz.* conjunction or existential quantification. For the current application *i.e.* CEC, this single feature gives a SAT based algorithm several operational advantages. Properly harnessed, these can translate into significant gains in the overall efficiency and robustness of the tool. Some of these are discussed below.

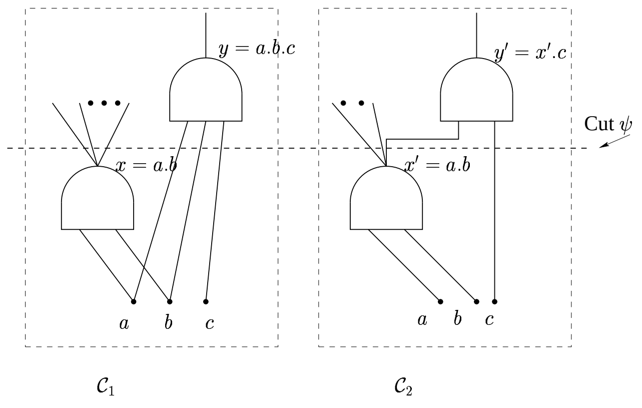

Figure 1. False Negatives can be resolved by local BCP

3.1 Locality and robustness of cutpoint resolution

In order to deduce an internal equivalence $x \equiv x'$ a typical BDD based deduction engine has to build BDDs of x and x' in terms of a common set of cutpoints $(\mathbf{Y,Y'})$ such that $x(\mathbf{Y}) = x'(\mathbf{Y'})$. In order to determine a suitable set $(\mathbf{Y,Y'})$ such methods [4, 16, 21] resort to a host of heuristics to *resolve* cutpoints backwards till a suitable cut is found. Such an approach is inherently unrobust since there is no good criterion to determine the "right cut" to learn an equivalence from. Thus, often such an approach comes up with a set $(\mathbf{Y,Y'})$ much larger and farther away than is needed to learn the equivalence. The key point is that the inability to learn $x \equiv x'$ from a given cut $(\mathbf{Y,Y'})$ is due to the presence of certain *false negatives* on the this cut. Often it is possible to resolve these through local BCP or a fairly "local branch and bound search" by a SAT algorithm.

Example 3.1 *Consider the circuit of Figure 1 where the signals y and y' are not equivalent in terms of the cut $\psi = \{x, x'\}, a, b, c$ but are actually globally equivalent in terms of the signals a, b, c* [2]. *The miscomparing patterns (e.g. $x = x' = 1, a = 0, b = 1, c = 1$) can be easily resolved by a SAT procedure through local BCP, while operating from the cut ψ, but a BDD based approach operating in terms of the same cut would not be able to deduce the equivalence $y \equiv y'$.*

3.2 Use of previously deduced equivalences

To the best of our knowledge all cutpoint based methods immediately merge nodes that are deduced as equivalent.

[2]such a situation is frequently produced by simple operations such as factoring and re-substitution in logic optimization

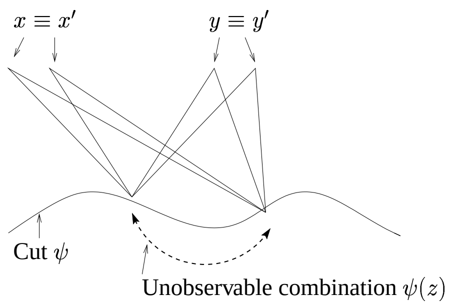

Figure 2. Previously deduced equivalences as shallow witnesses of false negatives

With BDD based methods there is probably no benefit in doing otherwise. However, with SAT based methods it is possible to simply add the deduced equivalence as a clause or constraint to the overall formula without merging the two nodes and benefit from it.

Example 3.2 *Consider the example of Figure 2 where the current cutpoint frontier ψ has an unobservable assignment $\psi(z)$ which prevents us from learning equivalences $x \equiv x'$ as well as $y \equiv y'$ from the cut ψ. Once we have expended some branching effort in backjustifying this false negative for learning $x \equiv x'$ we can add in $x \equiv x'$ as a* local witness *of $\psi(z)$ (and not merge x, x') so that when trying to learn $y \equiv y'$ this false negative can be justified with no branching effort (since this assignment will immediately violate $x \equiv x'$).*

The same reasoning applies to not merging equivalences behind the current cutpoint-frontier so that they can be used as *shallow witnesses* of unobservable assignments when trying to backjustify false negatives towards the primary inputs.

3.3 Learning more general relationships

In almost all BDD based cutpoint methods the notion of cutpoints corresponds to equivalence relationships (or equivalence modulo inversion) in terms of the circuit primary inputs. Such relationships can be naturally obtained by BDD pointer comparisons. However, using SAT methods it is possible to work with a much more general notion of cutpoints. One such generalization [2] proposed the notion of *replacability* of gates where x can be replaced with y iff on replacing x with the gate $z = x \oplus y$ there does not exist a test for the *stuck-at-0* fault at the output of z. This and a number of other variations of this notion can be realized by slightly modifying the SAT problem posed to the solver. However, for simplicity we have chosen not to exercise this degree of freedom in this work.

4 Proposed Methodology

Our overall framework is similar to most cutpoint based methods as described in Section 2. The key difference is that we use SAT procedures alone to accomplish both the deduction and the justification phases. As in [10] the two circuits to be checked for equivalence are decomposed into a network of two input AND gates, allowing inversions on the edges. This decomposed network is used as the base data-structure. Currently, the deduction procedure is restricted to deducing *equivalences* ($x \equiv y$) and *equivalences modulo complementation* ($x \equiv \overline{y}$). Thus, for the purpose of this exposition we refer to the *cut* as the *equivalence cut*. All deduced relationships are tagged onto the respective gates. Thus each node (gate), x has associated with it a (potentially NULL) set of nodes, called its *equivalence class*. This is the set of nodes which have been deduced as being equivalent to x.

Our methodology is implemented through a combination of two SAT engines which work in tandem in an interleaved fashion. The first engine is an inexpensive, DLL based engine designed to catch most of the “easy to prove” equivalences in the “vicinity” of the equivalence cut. The second engine uses a more advanced general purpose SAT solver (in our case the GRASP [15] solver) to deduce the relatively more difficult equivalences. The two engines are described below.

4.1 Segment sweeping based deduction

This engine is designed to catch all equivalence pairs (x, x') such that the cardinality of the combined support of x and x' in terms of the current equivalence cut, ψ is less than some specified parameter k. The intuition behind this engine is similar to motivation of the node hashing scheme [10]. It is roughly analogous to building BDDs, in terms of the current cut, of all those nodes whose support size (in terms of ψ) is less than k and deducing all equivalences that can be deduced from these BDDs. However, for reasons discussed in Section 3 our method is much more powerful than either the BDD schemes or hashing, even if the hashing is generalized on the lines of [6].

Let ψ denote the cutpoint frontier. A *segment* ϕ of this cut, is a subset of cutpoints (as well as their equivalent counterparts) of ψ. Let *Base*(ϕ) denote all those gates (variables), lying in front of the cut ψ in circuits $\mathcal{C}_1$ and $\mathcal{C}_2$, whose support in terms of ψ is a subset of the segment ϕ. Consider a pair of variables $x, x' \in$ *Base*(ϕ). The equivalence deduction procedure is based on the following principle. Under each assignment to the variables of ϕ, each variable of *Base*(ϕ) takes a Boolean value (0 or 1). If x and x' assume the same Boolean value under each of these $2^{|\phi|}$ assignments, then $x \equiv x'$ “globally”, i.e. in terms of the primary inputs. Similarly, if $x \not\equiv x'$ (globally) then the following result follows.

Proposition 4.1 *Given segment ϕ of cut ψ and two variables $x, x' \in$ Base(ϕ), if $x \not\equiv x'$ (globally) then there exists a Boolean value assignment α to the variables of ϕ such that $x(\phi(\alpha)) \neq x'(\phi(\alpha))$.*

In fact this proposition can be further strengthened by the addition of the following additional condition.

Proposition 4.2 *Given segment ϕ of cut ψ and two variables $x, x' \in$ Base(ϕ), if $x \not\equiv x'$ (globally) then there exists a Boolean value assignment α to the variables of ϕ and an assignment ρ to the primary inputs I such that $\phi(I(\rho)) = \alpha$ and $x(\phi(\alpha)) \neq x'(\phi(\alpha))$.*

Based on Proposition 4.1 equivalence relationships are deduced by constructing and manipulating *equivalence classes* as follows. Given a segment ϕ of cut ψ the variables *Base*(ϕ) are first put into a single equivalence class, Γ. Then each of the $2^{|\phi|}$ assignments to ϕ is explored one by one with the associated values of the variables *Base*(ϕ) under each assignment, using Boolean value propagation through the circuits. Suppose under the first assignment to ϕ, variables Γ_1 evaluate to 1 whereas variables Γ_0 evaluate to 0, where $\Gamma_0, \Gamma_1 \subseteq \Gamma$, $\Gamma_0 \cup \Gamma_1 = \Gamma$ and $\Gamma_0 \cap \Gamma_1 = \emptyset$. Then the equivalence class Γ is split into two sub-classes, Γ_0 and Γ_1. This process is repeated for each current equivalence class, after each assignment (and value propagation) to ϕ. After exploring all $2^{|\phi|}$ assignments to the segment, if two variables x and x' lie in a common equivalence class, it follows from Proposition 4.1 that $x \equiv x'$ holds globally.

Using the result of Proposition 4.2 the above strategy can be improved considerably. First, when branching on the segment variables (i.e. exploring the $2^{|\phi|}$ possible assignments to segment variables) complete Boolean value propagation is done after each variable assignment. The propagation is carried both in front of and behind the cut, using the functional gate level circuit description as well as all previously deduced equivalence relationships. The current branch is terminated as soon a *conflict* is encountered. Secondly, the equivalence classes are split if and only if the branching doesn't terminate in a conflict. The above deduction procedure based on branching on a single segment and splitting equivalence classes is called a *segment deduction run*.

Given the current equivalence cut, ϕ, the local deduction of equivalences is accomplished through a sequence of segment deduction runs, each with a new segment drawn from ϕ. At the end of these runs we can informally guarantee that all equivalences, deducible from the current cut and within a certain neighborhood of it, have been deduced. In our experiments a segment size of 5 provided a good compromise between deduction power and efficiency.

4.2 Global hypothesis based deduction

While the above procedure works fairly well in regions of the circuit where equivalences are abundant and densely scattered, it cannot be generalized to handle all equivalences for the following reasons:

- The distribution of equivalences is highly non-uniform for difficult verification instances. Hence it is impossible to determine a good value of the segment size a priori, in the absence of which the algorithm does a lot of wasted work.
- In some cases, especially arithmetic circuits, missing even a few equivalences can make an appreciable difference to the difficulty of the remaining sub-problem.

We concur with the view of [4] on the issue that for more difficult equivalences one needs a robust approach to generate candidate pairs of cut-points to verify (we refer to these as *global hypothesis*) and a robust mechanism for verifying these pairs that does not work off a preset "hard" threshold on the amount of effort to invest in verifying a particular pair.

Our framework for global hypothesis generation and proving, draws on techniques [2, 4, 21] and is similar to the one used in [4]. The key difference is that a **single SAT algorithm** is used both for proving equivalent pairs as well as identifying true negatives[3]. The algorithm pseudo code is shown in Algorithm 1.

Algorithm 1 Global Hypothesis Deduction

```
GenerateInitGlobalHypothesisClasses
while Outputs Unresolved & Not Deduced Sufficient New Equivs. do
  $(x_1, x_2)$ = ChooseHypothesis
  Status = ResolveHypothesis($x_1, x_2$)
  if Status = "TRUE_NEGATIVE" then
    RefineGlobalHypothesisClasses
    if Outputs in different classes then
      return UNEQUAL, test
    end if
  else
    MergeEquivClasses($x_1, x_2$)
    if Outputs in the same EquivClass then
      return EQUAL
    end if
  end if
end while
```

The **GenerateInitGlobalHypothesisClasses** routine picks up all nodes in the transitive fanout of the current

[3][4] used BDDs for the proving equivalences and heuristically combined it with a randomized SAT algorithm implemented on the BDDs to identify some of the true negatives

cut-point frontier and clusters them into *Global Hypothesis Classes* by running 32-bit parallel simulation on the circuit. Nodes with identical signatures under this simulation lie in the same global hypothesis class. The simulation can be performed with purely random vectors or any other "interesting" set of vectors. This function is used only when global equivalence deduction is invoked for the first time.

ChooseHypothesis selects a pair of nodes, x_1, x_2, belonging to the two circuits from a global hypothesis class, such that the pair is topologically closest to the current cut frontier. This hypothesis is resolved by invoking a SAT solver on the formula denoting $x_1 \oplus x_2$. All previously deduced equivalences are part of this formula. If the formula is unsatisfiable then $x_1 \equiv x_2$ and hence the corresponding equivalence classes[4] are merged. If the formula is satisfiable, the solution returned provides a vector to simulate and refine *i.e.* to split the current global hypothesis classes (routine **RefineGlobalHypothesisClasses**).

This process iterates till the primary outputs are resolved or a certain number (this is a parameter to the algorithm) of new cutpoints are deduced. After this the cutpoint frontier is advanced and segment sweeping is initiated again, returning to global hypothesis deduction when segment sweeping gives up.

5 Experimental Results

We present experimental results based on a preliminary implementation of our proposed methodology for SAT based CEC. It has been implemented in **C** and uses the **GRASP** SAT solver [15] for the global hypothesis based deduction phase (Section 4.2). Our results are reported on a 167 MHz Sun Ultra Sparc-1 with 256 Mbytes of memory. Our current interface to GRASP is through files [5] but the reported runtimes do not include the file I/O times since this can easily be removed through a better integration of the tools. As mentioned earlier, the main objective is to present a realistic assessment of a SAT based CEC tool, rather than to present an optimized and complete equivalence checker.

We present two sets of results. The first set compares our method against the following four tools which in our opinion represent the state of the art in SAT based combinational equivalence checking.

1. **RL_GRASP [14] :** An implementation of GRASP augmented with *Recursive Learning* [12].
2. **RL_CGRASP [13] :** An implementation of GRASP augmented with Recursive Learning as well a framework for exploiting circuit topology.
3. **HANNIBAL [11] :** A CEC tool using Recursive Learning and a test generator.
4. **Implication Graph based method [20]:** Presents a tuned and optimized implementation of a backtracking SAT algorithm that employs some elements of non-local implications and recursive learning. The algorithms are implemented on a specialized data-structure called the *implication graph*.

The comparative results presented in Figure 3 show the results of verifying the ISCAS'85 benchmarks against their irredundant versions. These benchmarks are relatively easy instances of combinational verification. They are also the only common set of benchmarks on which the above tools have reported results.The results of **RL_CGRASP** were obtained by running the publicly available version of the tool on our machine, using optimized settings which the authors [13] kindly provided. Since **RL_GRASP** was not available we used the runtime numbers from [14] appropriately scaling for architectural differences. The results for HANNIBAL are the original numbers from the paper [11] since we were unable to find suitable scaling data for the Sparc ELC station used by the authors in those experiments. The results of [20] are the numbers from the original paper reported on a DEC Alpha Station $250^{4/266}$, which is a machine comparable in performance to our own. Although, a direct exact value to value comparison is neither fair nor intended, the results of Figure 3 clearly demonstrate that our method consistently outperforms all the other techniques. Especially noteworthy is the fact that on the three hardest instances of the set, namely `C5315, C3540` and `C7552` our method outperforms all the other methods by over two orders of magnitude.

The second set of results provide a comparison on a much more difficult set of instances with a state-of-the-art BDD based mixed engine combinational equivalence checker [4]. Figure 4 reports results on verifying some of the MCNC91 circuits against a version optimized by a general purpose logic optimization script, `script.rugged` from SIS [18]. The results of [4] are reported on the same machine as ours. It is noteworthy that even with our current untuned and prototype implementation our runtimes are mostly comparable to that of [4], sometimes a factor of 2-3 slower. However *C3540* is an example where our algorithm is in fact faster than [4]. Interestingly enough, this is an example with a fairly non-uniform distribution of cutpoints, some of which are fairly hard to deduce. We believe the runtime discrepancy can be easily made up and in fact bettered by the improvements listed in Section 6.

6 Conclusions and Future Directions

We have revisited the application of Satisfiability (SAT) algorithms to CEC. We argued the case for SAT as a more

[4]i.e. the equivalences tagged to each node as explained earlier in this section

[5]GRASP is written in C++ whereas our tool is in C

Circuit	RL_GRASP (secs)	RL_CGRASP (secs)	HANNIBAL (secs)	Implication Graph Method [20] (secs)	Our Method (secs)
C432	2.8	3.6	3	1.3	0.7
C499	6.8	8.8	6	1.4	1.17
C1355	18.0	27.4	19	7.0	2.37
C1908	94.8	153.0	26	19.5	3.87
C2670	56.4	74.6	231	24.1	4.46
C3540	4006	2560	2057	791.0	38.94
C5315	445.4	476.6	797	33.4	6.96
C6288	109.6	43.6	48	8.9	5.04
C7552	2124	2868	4724	570.1	23.11

Figure 3. Verifying original vs. irredundant circuits

Circuit	Mixed Engine [4] (secs)	Our Method (secs)
C432	–	2.14
C499	–	0.92
C1355	–	1.1
C1908	–	5.90
C2670	3.5	4.93
C3540	**25.7**	**20.98**
C5315	5.3	27.45
C6288	12.1	14.52
C7552	12.7	35.18

Figure 4. Verifying original vs. optimized circuits

robust and flexible engine of Boolean reasoning for the CEC application than BDDs. We presented a simple framework for SAT based CEC and reported results on a preliminary implementation. The results show a speedup of up to two orders of magnitude compared to state-of-the-art SAT based methods for CEC. They also demonstrate that even with this simple algorithm and untuned prototype implementation it is only moderately slower and sometimes faster than a state-of-the-art BDD-based mixed-engine commercial CEC tool.

There are several avenues for improvement of the current algorithm and implementation:

- **Variable ordering in the SAT solver:** It is well known that variable ordering can affect the performance of SAT solvers, tremendously[6]. Currently, we have experimented with only a few *static variable ordering* schemes with GRASP. More experimentation in this direction could provide substantial speedups.

- **Better implementation of our CEC framework:** The current data-structure and routines are designed for flexibility of rapid algorithm prototyping rather than optimality of the specific proposed framework. Once rewritten and tuned for efficiency, these could easily speed up the time spent outside the calls to the SAT solver at least by a factor of 2-5. This time contributes 30-70% of the overall reported time.

- **More effective use of Initial Vector Simulation:** Currently the 32-bit parallel vector simulation, used for pruning the hypothesis set, works with randomly generated vectors. However, simulating a more intelligent set of vectors could substantially cut down the number of calls to the SAT solver and boost performance proportionally. One idea to do this is to make use of test vectors that are routinely generated for simulating designs, during the design process. These could be "interesting" vectors proposed by the designer or the ATPG test set for one or both of the circuits being checked.

- **Sharing effort between individual hypothesis checks:** One of the reasons for the efficiency of BDD based methods is their ability to re-use previous work by storing part of the Boolean search space in the form of the BDD itself. While our current method makes use of previously done work by storing and using previously deduced equivalences as shallow witnesses of conflicts (Section 3), a more direct sharing of effort between individual SAT calls, somewhat along the lines of [9] could substantially improve performance.

- **Using an improved SAT solver:** As research in SAT solvers produces better algorithms and implementations performance of the proposed framework will improve.

While it is clear that SAT based CEC methods need further research and application based tuning before they can surpass almost a decade of research in BDD based combina-

[6]just as BDD variable ordering affects the size of BDDs

tional verification, SAT based methods for verification are certainly a promising option and merit continued research.

Acknowledgements

We would like to thank Jerry Burch, Armin Biere, Andreas Kuehlmann and all the reviewers of this paper for useful discussions, feedback and comments.

References

[1] C. L. Berman and L. H. Trevillyan. Functional Comparison of Logic Designs for VLSI Circuits. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 456–459, November 1989.

[2] D. Brand. Verification of Large Synthesized Designs. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 534–537, Nov 1993.

[3] R. E. Bryant. Graph Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C(35):677–691, August 1986.

[4] J. R. Burch and V. Singhal. Tight Integration of Combinational Verification Methods. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 570–576, November 1998.

[5] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[6] M. K. Ganai and A. Kuehlmann. On-the-fly compression of logical circuits. In *IEEE/ACM International Workshop on Logic Synthesis*, May 2000.

[7] J. Jain, R. Mukherjee, and M. Fujita. Advanced Verification Techniques Based on Learning. In *Design Automation Conference*, pages 420–426, June 1995.

[8] J. Jain, A. Narayan, M. Fujita, and A. Sangiovanni-Vincentelli. Formal verification of combinational circuits. In *International Conference on VLSI Design*, pages 218–225, January 1997.

[9] J. Kim, J. Whittemore, J. Marques-Silva, and K. Sakallah. On applying incremental satisfiability in delay fault testing. In *IEEE/ACM Design and Test in Europe*, March 2000.

[10] A. Kuehlmann and F. Krohm. Equivalence Checking Using Cuts and Heaps. In *Design Automation Conference*, pages 263–268, June 1997.

[11] W. Kunz. HANNIBAL: An Efficient Tool for Logic Verification Based on Recursive Learning. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 538–543, November 1993.

[12] W. Kunz and D. Stoffel. *Reasoning in Boolean Networks.* Kluwer Academic Publishers, 1997.

[13] J. Marques-Silva and L. G. e Silva. Algorithms for Satisfiability in Combinational Circuits Based on Backtrack Search and Recursive Learning. In *Workshop notes of The International Workshop on Logic Synthesis*, pages 227–241, June 1999.

[14] J. Marques-Silva and T. Glass. Combinational Equivalence Checking Using Satisfiability and Recursive Learning. In *IEEE/ACM Design, Automation and Test in Europe*, pages 145–149, March 1999.

[15] J. Marques-Silva and K. A. Sakallah. GRASP-A New Search Algorithm for Satisfiability. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 220–227, November 1996.

[16] Y. Matsunaga. An Efficient Equivalence Checker for Combinational Circuits. In *Design Automation Conference*, June 1996.

[17] V. Paruthi and A. Kuehlmann. Equivalence Checking Using a Structural SAT-solver, BDDs, and Simulation. In *International Conference on Computer Design*, September 2000.

[18] E. M. Sentovich, K. J. Singh, C. M. anf H. Savoj, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Sequential Circuit Design Using Synthesis and Optimization. In *International Conference on Computer Design*, pages 328–333, October 1992.

[19] G. Stalmårck. A system for determining propositional logic theorems by applying values and rules to triplets that are generated from a formula, 1989. Swedish Patent 467 076 (1992), US Patent 5 276 897 (1994), European Patent 0 403 454 (1995).

[20] P. Tafertshofer, A. Ganz, and M. Henftling. A SAT-Based Implication Engine for Efficient ATPG, Equivalence Checking and Optimization of Netlists. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 648–657, November 1997.

[21] C. A. J. van Eijk. *Formal Methods for the Verification of Digital Circuits*. PhD thesis, Eindhoven University of Technology, Dept. of Electrical Engineering, Eindhoven, Netherlands, 1997.