SAT-Based Unbounded Symbolic Model Checking

Hyeong-Ju Kang and In-Cheol Park
Dept. of EE, KAIST
373-1 Guseong-dong Yueong-gu
Daejeon, KOREA
+82-42-869-4405
{dk,icpark}@ics.kaist.ac.kr

ABSTRACT
This paper describes a SAT-based unbounded symbolic model checking algorithm. BDDs have been widely used for symbolic model checking, but the approach suffers from memory overflow. The SAT procedure was exploited to overcome the problem, but it verified only the states reachable through a bounded number of transitions. The proposed algorithm deals with unbounded symbolic model checking. The conjunctive normal form is used to represent sets of states and the transition relation, and a SAT procedure is modified to compute the existential quantification required in obtaining a pre-image. Some optimization techniques are exploited, and the depth first search method is used for efficient safety-property checking. Experimental results show the proposed algorithm can check more circuits than BDD-based symbolic model checking tools.

Categories and Subject Descriptors
B.7.2 [**Integrated Circuits**]: Design Aids – *Verification*

General Terms Algorithms, Verification.

Keywords
Formal verification, symbolic model checking, unbounded symbolic model checking, Boolean satisfiability checking.

1. INTRODUCTION
As the design complexity is increasing rapidly, design verification is attracting more attention in recent years. Formal verification has emerged during the last decade as a promising verification method that can detect corner-case errors that are hardly detected by simulation [1]. Formal verification has a variety of methods, and model checking is most widely used among them. A major difficulty in applying model checking to practical designs is the state explosion problem that the problem size grows very rapidly as the size of a target design grows.

In 1990, K. L. McMillan and et al. proposed a BDD-based model checking algorithm [2], and many methods have been proposed to improve the algorithm. Although the BDD-based model checking is efficient and being widely used, it is still restricted to a limited range of circuits because of the memory explosion. In order to manage large circuits, it is required to properly order the variables, but it is another difficult problem to find good orderings. Bounded Model Checking (BMC) [3] uses Boolean Satisfiability Checking (SAT) procedures instead of BDDs. The SAT problem is to determine whether a given Boolean formula has a satisfying assignment [4], [5]. Although BMC algorithms can check large circuits, they guarantee the property's truth only for bounded transitions. In opposition to BMC, model checking that is not limited by a bound is sometimes called Unbounded Model Checking (UMC) [6].

In this paper, we propose a SAT-based UMC algorithm. Sets of states and the transition relation are expressed in Conjunctive Normal Form (CNF). The proposed algorithm performs quantification required in image or pre-image computation by using a modified SAT procedure that generates all satisfying assignments. The generated assignments are compressed by a two-level logic minimizer. A conventional SAT solver determines whether a fix-point is reached. To make the proposed algorithm more efficient, iterative squaring and score adjustment are proposed. In addition, some depth first techniques are exploited for faster safety-property checking.

The rest of this paper is organized as follows: Section 2 discusses the differences between the related works and the proposed algorithm. Section 3 explains how a SAT procedure is modified to perform quantification. Section 4 presents the proposed algorithm, and Section 5 shows experimental results. Concluding remarks are made in Section 6.

2. RELATED WORKS
The main reason why BDDs cannot deal with large circuits is that BDDs are based on canonical representation requiring much memory. Therefore, some works have been made to reduce the memory requirement by using non-canonical representation.

A. Gupta and et al. proposed an algorithm that represents sets of states in BDDs and the transition relation in CNF [7]. The algorithm performs quantification with a modified SAT procedure similar to ours. However, when a value is assigned to a variable, the algorithm checks whether the assignment makes the state set BDD a zero BDD. The proposed algorithm does not require the separate checking because sets of states are also represented in CNF.

Algorithms in which sets of states and the transition relation are represented in formula have been proposed in [8], [9]. They usually use their own representations for formulas, such as Binary

This work was supported by the Korea Science and Engineering Foundation through the MICROS center.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.

Expression Diagrams (BEDs) and Reduced Boolean Circuits (RBCs). They do quantification $\exists x.f(X)$ by generating an equation $f(X)|_{x=0}$ and an equation $f(X)|_{x=1}$ and taking the disjunction of the two equations. Though the algorithms include some reduction techniques, the lengths of formulas increase exponentially for quantification. Therefore, they can be applied to a limited range of circuits.

K. L. McMillan proposed a SAT-based UMC algorithm [6] which represents sets of states and the transition relation in CNF and performs quantification $\exists x.f(X)$ by invoking a SAT procedure with a CNF formula equivalent to $\sim f(X)$ and gathering clauses that are complements of satisfying assignments. The algorithm requires redrawing an implication graph and consumes much time, while the proposed algorithm does not redraw the implication graph and has some efficient techniques.

3. QUANTIFICATION BY A MODIFIED SAT PROCEDURE

The existential quantification is denoted as follows:

$$g(X) = \exists Y.(f(X,Y)) \quad (1)$$

, where X and Y are sets of variables. The equation means that an assignment of X satisfies $g(X)$ iff there is an assignment of Y such that the assignments of X and Y satisfy $f(X,Y)$. In this section, we will present how to obtain $g(X)$, given $f(X,Y)$, X, and Y.

3.1 The Conventional SAT Procedure

The SAT procedure determines whether there is an assignment that satisfies a given formula represented in CNF. A CNF formula is the conjunction of clauses, each of which is the disjunction of literals, where a literal is a variable or its complement. An arbitrary Boolean formula f can be transformed into a CNF formula f_{CNF} with a representative variable v_f, where $(f_{CNF} \wedge v_f)$ is satisfiable iff f is satisfiable and $(f_{CNF} \wedge \sim v_f)$ is satisfiable iff $\sim f$ is satisfiable. In this CNF form, an operation between CNF formulas can be implemented easily. Given a binary operation $\circ$, the CNF formula $(f \circ g)_{CNF}$ is the conjunction of $f_{CNF} \wedge g_{CNF}$ and a CNF formula equivalent to $(v_{f \circ g} \leftrightarrow v_f \circ v_g)$.

Many SAT solvers have been proposed, and most of them are based on the Davis and Putnam procedure [4]. After a variable is selected and its value is decided, the procedure reveals all implications resulting from the decision. If a conflict occurs, the procedure analyzes the conflict, inserts conflict clauses, and then backtracks to the decision point.

3.2 The Modified SAT Procedure

The modified SAT procedure takes a CNF formula $f_{CNF}(X,Y)$ and variable sets X and Y, and returns all the assignments of X that satisfy $g(X)=\exists Y.(f(X,Y))$. The major difference from the conventional SAT procedure is that the modified SAT procedure does not stop when a satisfying assignment is found. All the satisfying assignments are needed to construct $g(X)$. Therefore, when a satisfying assignment is found, the assignment of X is stored and the procedure backtracks to continue as in the case that a conflict occurs.

Figure 1 is a pseudo code of the procedure, where G is the set of assignments of X that have an assignment of Y satisfying $f(X,Y)$, in other words, the assignments of X satisfying $g(X)$. When the

```
1  mSAT(f_CNF,X,Y) {
2      G = ∅;
3      while(1) {
4          if all clauses are satisfied {
5              insert the assignment of X into G;
6              insert an excluding clause to f_CNF;
7              if(backtrack()=fail) return G;
8          } else {
9              choose an undecided variable v from X U Y;
10             assign a value to v;
11             if(deduce()=conflict) {
12                 analyze the conflict;
13                 insert conflict clauses to f_CNF;
14                 if(backtrack()=fail) return G;
15             }
16         }
17     }
18 }
```

Figure 1. The modified SAT procedure.

procedure is completed, G has all the assignments satisfying $g(X)$. An assignment of X satisfying $g(X)$ can have several assignments of Y satisfying $f(X,Y)$. To search such an assignment of X only once and reduce the search time, an excluding clause at the sixth line in Figure 1 is inserted, which is equivalent to the complement of the found assignment of X.

3.3 Converting a Set of Assignments into a CNF Formula

The modified SAT procedure in the previous subsection provides a set of all the assignments satisfying $g(X)$. However, it should be converted into a CNF formula to use it in the proposed model checking algorithm. The set of assignments, G, can be thought of as a two-level logic circuit in sum-of-products form. It can be, therefore, transformed into a CNF formula. As a result, we obtain a CNF formula g_{CNF} with a representative variable v_g. Before converting the assignment set into a CNF formula, it is processed by a two-level logic minimizer to reduce the size of the assignment set and the resulting CNF formula. Any two-level logic minimizer can be used, but we have used espresso [10] to obtain experimental results.

4. A SAT-BASED UNBOUNDED MODEL CHECKING ALGORITHM

The conventional UMC algorithm is based on BDDs. In this section, we will propose a SAT-based UMC algorithm. The algorithm is different from the BDD-based UMC algorithm in that BDD operations are substituted with CNF operations explained in Section 3.1. Due to the limit of length, we only present the procedures related with pre-image and fix-point calculation in this paper.

4.1 Pre-image and Fix-point Calculation

Generally, a pre-image is computed as follows:

$$S(V) = \exists V'.(T(V,V') \wedge S'(V')) \quad (2)$$

, where $T(V,V')$ is the transition relation, V and V' are the sets of the current and next state variables, $S(V)$ and $S'(V')$ are the sets of the current and next states, and $S(V)$ is the pre-image of $S'(V')$. In

```
preImage(S',T) {
    X = V;
    Y = D(T) – V;
    substitute variables of V in S' with the
        corresponding variables of V';
    Y = Y U D(S');
    G = mSAT(S'□T□v_S',□v_T,X,Y)
    G' = espresso(G)
    return convertCNF(G');
}
```

Figure 2. Pre-image calculation.

our algorithm, $S'(V')$ and $T(V,V')$ are represented in CNF, and quantification is done by the modified SAT procedure described in the previous section. In fact, $T(V,V')$ and $S'(V')$ do not depend only on the current and next state variables. They have the input and output variables and some intermediate variables inserted when their CNF formulas are constructed. Those variables must be quantified, too.

Figure 2 is a pseudo code that computes a pre-image, where $D(f)$ is a function that returns a set of variables the formula f depends on. The modified SAT procedure in Section 3 returns a set of the assignments of the current state variables, G, which is equivalent to the pre-image of S'. The set is processed by espresso to reduce the number of assignments. At last, the reduced set is converted into a CNF formula to be returned. A fix-point is obtained by successively applying the pre-image calculation. The conventional SAT procedure is used to check whether a fix-point is reached.

4.2 Improving Techniques

Iterative squaring is used to reduce the number of iterations when calculating fix-points [2][9]. Before invoking the fix-point procedure, the transition relation is squared. Squaring the transition relation leads to increasing the number of variables and clauses in the transition relation CNF formula. Therefore, squaring the transition relation increases the time of each iteration, but reduces the number of iterations. The proposed algorithm squares the transition once as it leads to the best performance in experiments.

In the SAT procedure, the decision algorithm affects the overall performance very much. Increasing the probability of selecting a variable in a specific variable group can reduce the searching time. In the pre-image computation in Figure 4, we have three groups of variables: the current state variables V, the next state variables V', and the intermediate variables. The usual SAT problem prefers increasing the probability of selecting a state variable in V and V'. However, in the modified SAT procedure, increasing the probability of the intermediate variables gives better results. It is better to find a satisfiable assignment in which more state variables have unknown values because such an assignment leads to a shorter excluding clause that can reduce the search space more. If the state variables are decided earlier, almost all of them have values and the search space is reduced less. The proposed algorithm assigns higher priorities to the intermediate variables to decide them earlier.

4.3 Enhancing Safety-Property Checking

Safety properties are to assert that some bad states are never reached. They are usually written as AG $\sim p$ and encountered many times in the real model checking problems. One of the well-known techniques that improve the safety-property checking is to check at each iteration whether the reached set, R, has some states in the initial states. By this technique, we can know whether the safety property is false earlier than a fix-point is found. The proposed algorithm also includes some other efficient techniques for safety properties.

When computing a pre-image, it is not necessary to compute the entire pre-image at a time. In the proposed algorithm, when the pre-image is large, only a part of the pre-image is obtained, and the fix-point calculation is continued. In checking at each iteration whether R has some states in the initial states, some false safety properties can be revealed earlier. This technique does not always give better performance, but in many cases, it does.

The SAT-based BMC algorithm can check a property efficiently up to k transitions. For safety properties, the result that a safety property is false means that the property is really false. Therefore, by exploiting the BMC algorithm before computing a fix-point, we can know that the property is false without obtaining the fix-point. The proposed algorithm represents the transition relation in CNF and can invoke the SAT-based BMC procedure easily.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The performance of the proposed algorithm is compared with those of BDD-based model checking tools. The proposed algorithm is written in C language and compiled by g++ with –O3 option. A SAT solver modified from Chaff [5] is used in the experiment. The experiment is done on a SUN Blade 2000 workstation that has a 900MHz processor and 4GB memory. The BDD-based model checking tools compared are VIS [11] and Bwolen's SMV [12] because they are the most efficient ones ever known.

ISCAS 89 benchmark circuits are used for target designs. As the benchmark does not provide properties nor the information on functionality, properties to be checked are arbitrarily generated. All the generated are safety properties with a form AG $\sim p$, where p is a state.

Table 1 shows the processing times taken for the model checking algorithms to check each circuit. The first column denotes the benchmark circuits. The second, third, and forth columns specify the number of inputs, outputs, and D flip-flops in each circuit. The fifth and sixth columns represent the execution times taken by the BDD-based model checking tools, VIS and Bwolen's SMV, and the seventh column represents those by the proposed algorithm. The hyphen '-' means that the algorithm is halted because it takes more than 3 hours or more than 3.5GB memory. Experimental results for the circuits with identity numbers greater than 9234 are not shown in the table because all the three algorithms failed. The algorithms are compared in terms of two aspects: the number of circuits checked and the execution time. The table shows that the proposed algorithm can check 9 more circuits than the BDD-based ones. There is no case that the proposed algorithm cannot check the circuits that the BDD-based ones can. Most of the cases that the BDD-based tools fail to check are resulted from memory overflow, as the algorithms use BDDs to represent the transition relation and sets of states. However, the

Table 1. Experimental results for ISCAS 89 circuits

Circuits	#In	#Out	#DFF	VIS	SMV	Proposed	Circuits	#In	#Out	#DFF	VIS	SMV	Proposed
s27	4	1	3	0.03	2.92	0.05	s938	34	1	32	-	-	-
s208.1	10	1	8	0.2	3.47	3.6	s953	16	23	29	0.94	14.18	0.78
s298	3	6	14	0.26	3.36	0.34	s967	16	23	29	0.98	9.66	0.7
s344	9	11	15	0.31	5.24	0.32	s991	65	17	19	1.08	349.03	0.03
s349	9	11	15	0.33	4.42	0.37	s1196	14	14	18	1.11	30.9	0.94
s382	3	6	21	0.52	4.56	0.39	s1238	14	14	18	1.08	158.55	0.94
s386	7	7	6	0.25	4.03	0.31	s1269	18	10	37	-	154.39	1.05
s400	3	6	21	0.56	3.7	0.36	s1423	17	5	74	-	-	2.75
s420.1	18	1	16	9.71	10.63	151.44	s1488	8	19	6	1.21	15.85	6.62
s444	3	6	21	0.47	4.23	0.36	s1494	8	19	6	1.23	23.36	8.09
s499	1	22	22	0.33	5.88	0.39	s1512	27	21	57	-	18.04	1.63
s510	19	7	6	0.42	6.04	1.75	prolog	36	73	136	-	-	-
s526	3	6	21	0.54	4.31	0.56	s3271	26	14	116	-	-	2.68
s526n	3	6	21	0.54	4.39	0.49	s3330	40	73	132	-	-	2.93
s635	2	1	32	-	-	-	s3384	43	26	183	-	-	3.14
s641	35	24	19	0.85	5.4	0.56	s4863	49	16	104	-	-	3.74
s713	35	23	19	0.86	5.93	0.6	s5378	35	49	179	-	-	4.47
s820	18	19	5	0.6	7.87	0.96	s6669	83	55	239	-	-	5.52
s832	18	19	5	0.59	8.78	1.09	s9234.1	36	39	211	-	-	9.13
s838.1	34	1	32	-	-	-	s9234	19	22	228	-	-	9.29

proposed algorithm represents them in CNF that requires less memory than BDDs.

The processing times of the proposed algorithm are similar to those of the BDD-based ones except one case, s420.1. This circuit has a very long diameter, and the property is proved after about 2,000 iterations. The BDD-based tools can prove properties fast if the transition relation and sets of states are effectively expressed in BDDs. For such a circuit, the proposed algorithm takes a longer time than the BDD-based ones. This is a cost to pay for proving more circuits.

6. CONCLUSION

In this paper, we presented a SAT-based unbounded model checking algorithm. The sets of states and the transition relation of a target design are represented in CNF, which is non-canonical and memory efficient. The quantification, one of the most frequent operations encountered in model checking, is processed by a modified SAT procedure that generates all satisfiable assignments. The generated assignments are reduced by using a two-level logic minimizer, and converted into a CNF formula. The proposed algorithm obtains a fix-point with the quantification method. To enhance performance, incremental iterative squaring and score adjustment are included in the proposed algorithm. The proposed algorithm takes advantage of the depth first search to make safety-property checking more efficient. Experimental results show that the proposed algorithm can verify more benchmark circuits than the pervious BDD-based unbounded model checking algorithms.

7. REFERENCES

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

[2] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10^{20} states and beyond," in *Proc. Symp. Logic in Comput. Sci.*, 1990, pp. 428-439.

[3] A. Biere, A. Cimatti, E. M. Clarke, Y. Zhu, "Symbolic model checking without BDDs," in *Proc. Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1579 of Lecture Notes in Comput. Sci., 1999, pp. 193-207.

[4] M. Davis and H. Putnam, "A computing procedure for quantification theory," *Journal of the ACM*, vol. 7, pp. 201-205, 1960.

[5] M. W. Moskevicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering and efficient SAT solver," in *Proc. ACM/IEEE Design Automation Conf.*, 2001, pp. 530-535.

[6] K. L. McMillan, "Applying SAT methods in unbounded symbolic model checking," in *Proc. Int. Conf. Computer Aided Verification*, vol. 2404 of Lecture Notes in Comput. Sci., 2002, pp. 250-264.

[7] A. Gupta, Z. Yang, P. Ashar, and A. Gupta, "SAT-based image computation with application in reachability analysis," in *Proc. Int. Conf. Formal Methods in Computer Aided Design of Electron. Circuits*, vol. 1954 of Lecture Notes in Comput. Sci., 2000, pp. 354-371.

[8] P. F. Williams, A. Biere, E. M. Clarke, and A. Gupta, "Combining decision diagrams and SAT procedures for efficient symbolic model checking," in *Proc. Int. Conf. Computer Aided Verification*, vol. 1855 of Lecture Notes in Comput. Sci., 2000, pp. 124-138.

[9] P. A. Abdulla, P. Bjesse, and N. Eén, "Symbolic reachability analysis based on SAT-solvers," in *Proc. Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1785 of Lecture Notes in Comput. Sci., 2000, pp. 411-425.

[10] *Espresso*.[Online]. Available: http:// www-cad.eecs.Berkeley.edu/Software/software.html

[11] The VIS Group. *VIS 1.4*. [Online]. Available: http://vlsi.colorado.edu/~vis/.

[12] B. Yang. *SMV 2.4b*. [Online]. Available: http://www-2.cs.cmu.edu/~bwolen.