An Improved Hybrid SAT Solver for Bounded Model Checking in Circuit Design*

Yuesheng Zhu, Deke Yu

Communication & Information Security Lab, Shenzhen Graduate School, Peking University, Shenzhen, China
zhuys@pkusz.edu.cn, yudeke@sz.pku.edu.cn

Abstract – Model checking is one of main formal verification methods that are used in the process of circuit design and verification. However, there is a problem of state memory explosion in traditional model checking methods. Bounded model checking (BMC), in which the Davis-Putnam-Logemann-Loveland (DPLL) algorithm based Satisfiability (SAT) solver is used to verify the circuits, can avoid this problem, whereas, its efficiency depends on the performance of the solver. Hybrid SAT solver combines the advantages of the completeness of DPLL and the fast solving property of stochastic local search algorithm, e.g. WalkSAT, and is proved to be an efficient improving way. However, it is noted that the noise parameter in WalkSAT could affect the solver's overall performance. In this paper, an adaptive noise mechanism is proposed to integrate with the hybrid algorithm framework to further improve the solver's performance. Our experimental results have shown that the designed hybrid solver with adaptive noise performs well in BMC instances and some other circuits.

Index Terms – BMC, SAT, DPLL, WalkSAT, adaptive noise

I. Introduction

Verification method is an important way to guarantee the correctness of the circuits design. The traditional analog verification method is costly and incomplete. Formal verification is complete and can be used in the early stage of system design so that the design cost can be reduced greatly.

Model checking is a general formal verification method in hardware design. With the increase scale of systems, the state memory explosion problem limits the application of traditional model checking methods, e.g. binary decision diagram. Thus, to overcome this problem, the model of bounded model checking (BMC) [1] is developed and proved that it can be reduced to the satisfiability problem (SAT) in polynomial time. Then, the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [2] based SAT solver can solve the BMC more efficiently compared with traditional methods and overcome the problem of state memory explosion. However, DPLL-based SAT solvers, e.g. zChaff [3] and MiniSAT [4], are time-consuming in large instances, and even exponential time in worst cases. Stochastic local search algorithm (SLS) is a term of a set of local search algorithms, e.g. DLM [5] and WalkSAT [6]. SLS-based SAT solvers can quickly find solutions for a SAT problem, but the solvers can't prove the problem is unsatisfiability (UNSAT) if they stop without a SAT solution. Ref. [7] mentions that a hybrid SAT solver based on DPLL and SLS is an efficient improving way. Ref. [8] dynamically calls WalkSAT as a heuristic in some nodes of branching decisions during DPLL based on some conditional probability factor. Ref. [9] presents a general HyBrid-Incremental-SAT solver (HBISAT) framework. In this framework, SLS is used to identify a subset of clauses to be passed to DPLL through an incremental interface. In addition, the solution obtained by DPLL on the subset of clauses is fed back to SLS to jump over any locally optimal points. Ref. [10] adds circuit observability [11] into HBISAT. This method is much more efficient for large and hard circuits than other solvers. DLM & MiniSAT (DM) [12] takes the approximation solution which is obtained by DLM as an initial input for DPLL, guides DPLL to search the subspace that the approximation solution lies first.

In this paper, firstly, based on [12], another hybrid solver is implemented by replace DLM with WalkSAT, which is called WM for short. Secondly, it is noted that the noise parameter in WalkSAT, like the Lagrangian parameter in DLM, affects the overall performance of the hybrid solver's greatly no matter for the same instance or different instances. So with the progress of WalkSAT, an Adaptive-Noise-Hybrid-SAT solver (ANHSAT) is implemented by integrating the adaptive noise mechanism [13] into the hybrid algorithm of WM to further improve the solver's overall performance.

II. Preliminaries

A. Bounded Model Checking

BMC is proposed in [1], and the model of the problem can be described as follows:

Given the finite state machine (FSM) M, the task is to check whether M satisfies a temporal property P in all paths with length less to some bound k. The general structure of BMC invariant formula is following:

$$I_0 \wedge (\wedge_{i=0}^{k-1}\rho(i,i+1)) \wedge (\vee_{i=0}^{k}\neg P_i)\,. \qquad (1)$$

Where I_0 is the initial state of FSM, $\rho(i,i+1)$ is a formula representing the transition between cycles i and $i+1$, and P_i is the property in cycle i. It is not hard to see that (1) can be SAT if and only if there is a reachable state in cycle i which contradicts the property P_i. Otherwise, M satisfies all P for every path length with a large enough k. These can be checked by a SAT solver.

B. The Implement of DPLL

DPLL is first proposed in [2], although it has a history of more than 60 years, it is still the core of most complete SAT solvers, e.g. zChaff, MiniSAT and their variants. DPLL is

* The work described in this paper is supported by the "Shuang Bai Project", the Research Program of Shenzhen, China.

mainly consisted of three parts: branching decision, Boolean Constraint Propagation (BCP), and conflict resolving, other heuristic methods could be included into these three parts. The pseudo-code of DPLL is listed in TABLE I. Branching decision consists of variable selection and polarity decision. BCP checks if any other variable assignments may be implied or a conflict has been encountered under current assignments. Conflict resolving contains conflict analysis, learning, etc.

C. The Implement of WalkSAT

WalkSAT is proposed in [6], the pseudo-code of the algorithm is listed in TABLE II. There are three parameters which can be tuned in the algorithm: *MAX-TRIES*, *MAX-FLIPS*, and noise parameter *p*. At the beginning of the algorithm, a random truth assignment is generated to every variable. From there, it takes incremental steps in the space of complete assignments, flips the value of one variable at a time. If the current assignment satisfies the formula, then the algorithm returns SAT; otherwise, the algorithm will randomly pick an unsatisfied clause and choose a variable from the clause to flip its polarity with the probability of noise parameter *p*. After reaching *MAX-FLIPS*, WalkSAT will restart by another random assignment. If the *MAX-TRIES* threshold is also reached without obtaining a solution for the problem, the algorithm will stop. However, at this situation the algorithm can't prove that the solution doesn't exist, so it is incomplete.

TABLE I DPLL Algorithm

```
define DPLL
begin
    while (true)
        if (!decide())                  //branching decision
            return SAT
        while (!bcp())                  //BCP
            if (!ResolveConflict())     //conflict resolving
                return UNSAT
end
define RosolveConflict()
begin
    d = most recent decision not tried both ways
    if (d == null)  return false        //no such d is found
    flip the value of d
    mark d as tried both ways
    undo any invalidated implications
    return true
end
```

TABLE II WalkSAT Algorithm

```
define WalkSAT
begin
    for i = 1 to MAX-TRIES
        T = a randomly generated truth assignment
        for j = 1 to MAX-FLIPS
            if (T satisfies the formula)  then return SAT
            pick an unsatisfied clause C
            v = choose a variable randomly in C with probability p
            T = flip the assignment of v in T
        end for
    end for
    return "no satisfying assignment found"
end
```

III. Our Approach

TABLE III shows a set of experimental data which is obtained via MiniSAT by setting the default polarity decision order: All True, All False, and Random, which means once a variable is selected, this polarity order will be first assigned to the variable before BCP. It can be inferred from TABLE III that different polarity decision strategies have a great effect on the performance of the solver for different instances. This is because different polarity decision will guide the decision tree to different directions and form a different search tree. DM [12] uses DLM to generate an approximation solution and uses this solution to guide the polarity decision by setting the assignment in the approximation solution as the default polarity for the choosing variable.

In our work, since WalkSAT is another recognized SLS algorithm and could be better than DLM, it is chosen as the SLS part in the hybrid solver which is called WM for short. As is mentioned before, there are three parameters which can be tuned in WalkSAT. The noise parameter *p* has a major impact on the performance of WalkSAT, and the other two parameters have little or no impact on the behavior of WalkSAT [13]. After implementing WM, we find the noise parameter in WalkSAT could also affect the hybrid solver's overall performance in a great way which could be inferred from Fig. 1.

There are three types of information that can be potentially used to adjust the noise *p*:

(1) Background knowledge provided by the algorithm designer including theoretical and empirical knowledge, this may help others have an insight into the algorithm's behavior;

(2) Syntactic information about the problem instance, e.g. the number of clauses and variables, the length of clauses;

(3) Information collected over the run of the algorithm, including information about the search space positions and objective function values encountered over the search process.

Considering these three types of information, the adaptive noise mechanism which is proposed in [13] is integrating into the algorithm framework of WM to further improve the performance of it. The adaptive noise mechanism could be described as follows:

At the beginning of the search process, let noise be zero, this would typically lead to a series of rapid improvements in the objective function value followed by local optima (unless a solution to the given problem is found). In this situation, the noise value should be increased. If this increment is not sufficient to escape from the local optima after a certain number of steps, the noise value will be further increased. Ultimately, the noise value should be high enough that escape the local opti-

TABLE III Polarity Testing

Instance	# of variables	# of clauses	Polarity	CPU Time (s)
bmc-ibm-12	39598	194778	All True	4.6363
			All False	5.3581
			Random	4.2213
bmc-ibm-13	13215	65728	All True	9.1846
			All False	3.4745
			Random	4.2135

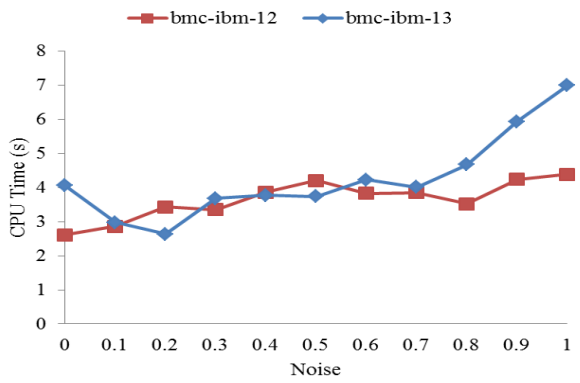

Fig. 1 The effect of the noise parameter on WM's overall performance

ma, at which point, the noise can be gradually decreased until happens to some next local optima or a solution of the given problem is found.

The above description can be formulated as following formulas:

(1) If the objective function value has not been improved over the last $\theta * m$ search steps, where m is the number of clauses of the given problem and θ is equal to 1/6, then the noise p will increase as:

$$p = p + (1 - p) * \varphi . \quad (2)$$

(2) Otherwise, for the decrement situation, p should decrease as:

$$p = p - p * \varphi / 2 . \quad (3)$$

Where φ above is equal to 1/5.

Although this adaptive noise mechanism introduces two additional parameters θ and φ to the algorithm, it is confirmed in [14] that these two parameters need not be tuned for each problem instance to achieve good performance for WalkSAT.

Thus, by integrating the adaptive noise mechanism into WM, ANHSAT is implemented and no parameter has to be manually tuned to solve a new problem. The flowchart of ANHSAT is shown in Fig. 2. The algorithm will first run the adaptive noise WalkSAT and return a SAT solution or an approximation solution of the problem. If not return SAT, the approximation solution obtained by the adaptive noise WalkSAT will be used as a heuristic polarity decision in DPLL. Each time when the DPLL algorithm selects an unassigned branching, the polarity decision module will first branch the polarity subspace in the approximation solution, and then go on other DPLL procedures. Obviously, ANHSAT is a complete solver. Due to limitations on space, the proof will not be given here.

IV. Experimental Results

ANHSAT is implemented with the following hardware and software environment:

- Hardware: host machine with Win7 platform, Intel Core2 Duo P8400 2.26GHz 2.27GHz CPU and 4GB RAM; virtual machine with Fedora15 in VMware Workstation 9.0, 1GB RAM and single core CPU;
- Software: WalkSAT v50, MiniSAT 2.2 and gcc 4.6.3.

In order to reduce the overhead of WalkSAT in ANHSAT, *MAX-TRIES* is set equal to 10 and *MAX-FLIPS* is equal to 1000. This setting can make the runtime of WalkSAT be negligible comparing with the process of DPLL. MiniSAT is chosen as the DPLL core. ANHSAT is compared with zChaff 2007, MiniSAT 2.2 and WM with static noise p equals 1/2 which may be good enough for many instances. The bmc series benchmarks are taken from SATLIB and others are circuit instances from the industrial instances of SAT Competition 2002 (see TABLE IV). Particularly, the bmc and cnt series are all BMC instances.

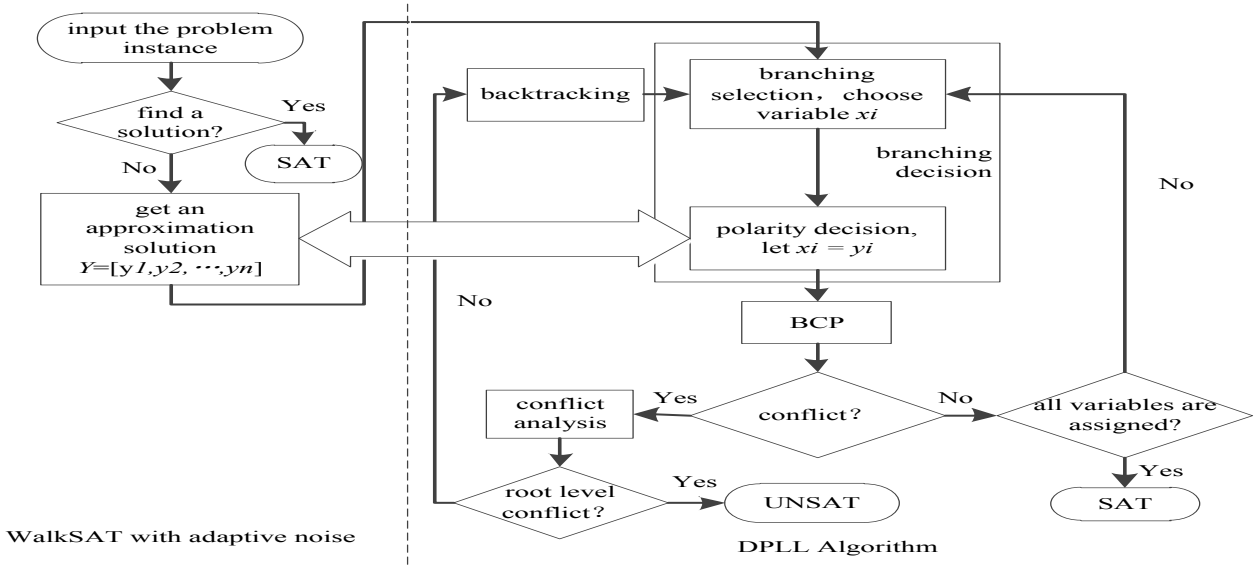

Fig. 2 The flowchart of ANHSAT

The *CPU Time* data are obtained by calculating the mean value from running all instances more than 10 times. *Speedup I* and *Speedup II* are calculated by *CPU Time* with following formulas respectively:

$$Speedup\ I = (MiniSAT - ANHSAT) / MiniSAT * 100\% . \quad (4)$$

$$Speedup\ II = (WM - ANHSAT) / WM * 100\% . \quad (5)$$

The bold numbers in TABLE IV represent the best performance. It can be noted from TABLE IV that ANHSAT is better than MiniSAT and WM in all instances, zChaff only performs well in some really simple instances. Moreover, we find that the performance of ANHSAT is outstanding in most BMC instances and some large and hard circuits.

V. Conclusions

BMC model is a variant of model checking widely used in circuit design and can be solved by SAT solver. Combining the advantages of DPLL and WalkSAT, a hybrid SAT solver ANHSAT is implemented. Different from the algorithm framework of DM, ANHSAT consists of adaptive noise mechanism to further improve the solver's performance. Our experimental results have demonstrated that ANHSAT is better than MiniSAT and WM with static noise mechanism and can efficiently solve the problems of BMC and some other large and hard circuits.

References

[1] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic Model Checking without BDDs," *Proc. of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 193-297, 1999.

[2] M. Davis, G. Logemann, and D. Loveland, "A Machine Program for Theorem-Proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394-397, 1962.

[3] Y. S. Mahajan, Z. Fu, and S. Malik, "Zchaff2004: An Efficient SAT Solver," *Proc. of the 7th International Conference on Theory and Applications of Satisfiability Testing*, pp. 360-375, 2004.

[4] N. Eén, N. Sörensson, "An Extensible SAT-Solver," *Proc. of the 6th International Conference on Theory and Applications of Satisfiability Testing*, pp. 502-518, 2003.

[5] Y. Shang, and B. W. Wah, "A Discrete Lagrangian-Based Global-Search Method for Solving Satisfiability Problems," *Global Optimization*, vol. 12, no. 1, pp. 61-99, 1998.

[6] B. Selman, H. Kautz, and B. Cohen, "Noise Strategies for Improving Local Search," *Proc. of the 12th National Conference on Artificial Intelligence*, pp. 337-343, 1994.

[7] H. Kautz, and B. Selman, "The State of SAT," *Discrete Applied Mathematics*, vol. 155, no. 12, pp. 1514-1524, 2007.

[8] B. Ferris, and J. Froehlich, "WalkSAT as An Informed Heuristic to DPLL in SAT Solving," *Department of Computer Science, University of Washington, Seattle*, 2004.

[9] L. Fang, and M. S. Hsiao, "A New Hybrid Solution to Boost SAT Solver Performance," *Proc. of the Design, Automation and Test in Europe Conference and Exhibition*, pp.1-6, 2007.

[10] X. Wang, H. Wang, and G. Ma, "Hybrid SAT Solver Considering Circuit Observability," *Proc. of the 9th International Conference on Young Computer Scientists*, pp. 65-70, 2008.

[11] Z. Fu, Y. Yu, and S. Malik, "Considering Circuit Observability Don't Cares in CNF Satisfiability," *Proc. of the Design, Automation and Test in Europe Conference and Exhibition*, pp. 1108-1113, 2005.

[12] M. Jing, D. Zhou, P. Tang, and X. Zhou, "A Heuristic Complete Algorithm for SAT Problem by Approximation Solution," *Computer Aided Design and Computer Graphics (Chinese)*, vol. 19, no. 9, pp. 1184-1189, 2007.

[13] H. Hoos, "An Adaptive Noise Mechanism for WalkSAT," *Proc. of the 18th National Conference on Artificial Intelligence*, pp. 655-660, 2002.

[14] C. Li, W. Wei, and H. Zhang, "Combing Adaptive Noise and Look-Ahead in Local Search for SAT," *Proc. of the 10th International Conference on Theory and Applications of Satisfiability Testing*, pp. 121-133, 2007.

TABLE IV Benchmark Testing Results

Instance	# of variables	# of clauses	CPU Time (s)				Speedup I (%)	Speedup II (%)
			zChaff 2007	MiniSAT 2.2	WM	ANHSAT		
bmc-ibm-1	9685	55870	0.3918	0.065	0.056	**0.044**	32.31	21.43
bmc-ibm-3	14930	72106	**0.0229**	0.1929	0.1626	0.1404	27.22	13.65
bmc-ibm-4	28161	139716	0.7019	0.212	0.1791	**0.1548**	26.98	13.57
bmc-ibm-6	51639	368352	1.0988	0.296	0.2427	**0.2334**	21.15	3.83
bmc-galileo-8	58074	294821	1.6358	0.4219	0.2624	**0.174**	58.76	33.69
bmc-galileo-9	63624	326999	1.2158	0.268	0.282	**0.2455**	8.40	12.94
bmc-ibm-10	59056	323700	6.6160	0.9749	0.7866	**0.6517**	33.15	17.15
bmc-ibm-11	32109	150027	8.5947	0.4559	0.4683	**0.3485**	23.56	25.58
bmc-ibm-12	39598	194778	18.3682	4.3243	3.1831	**2.8991**	32.96	26.39
bmc-ibm-13	13215	65728	2.5346	4.0624	2.8397	**2.162**	46.78	30.10
cnt08	4089	13531	6.419	0.9968	0.8742	**0.7911**	20.64	9.51
cnt09	9207	30678	34.0848	4.6143	4.7058	**3.4207**	25.87	27.31
ssa-sat-1	4824	48233	**0.084**	1.4708	0.5339	0.308	79.06	42.31
ca128	2282	6586	0.4429	0.154	0.133	**0.109**	29.22	18.05
ca256	4584	13236	1.8677	0.4929	0.3789	**0.3229**	34.49	14.78
cache_05	113080	431939	1.9297	18.2832	17.9395	**12.1415**	33.59	32.32
comb2	31933	112462	224.324	63.5855	69.7104	**54.6064**	14.12	21.67
comb3	4774	16331	>300	55.4536	64.2392	**48.5736**	12.41	24.39
f2clk_30	20458	59559	82.3025	16.0146	14.1691	**12.4291**	22.39	12.28
f2clk_40	27568	80439	>300	184.83	155.398	**147.56**	20.16	5.04
fifo8_100	64762	176313	10.6884	7.5479	6.9365	**4.6207**	38.78	33.39
fifo8_200	129762	353513	170.618	38.7761	38.8881	**36.0705**	6.98	7.25
fifo8_300	194762	530713	>300	120.514	101.276	**82.2905**	32.72	18.75