FPGA ROUTING ACCELERATION BY EXTRACTING UNSATISFIABLE SUBFORMULAS

Zhang Jianmin, Li Tiejun and Ma Kefan

College of Computer, National University of Defense Technology, Changsha, China

ABSTRACT

Explaining the causes of infeasibility of Boolean formulas has practical applications in various fields. A small unsatisfiable subset can provide a succinct explanation of infeasibility and is valuable for applications, such as FPGA routing. The Boolean-based FPGA detailed routing formulation expresses the routing constraints as a Boolean function which is satisfiable if and only if the layout is routable. The unsatisfiable subformulas can help the FPGA routing tool to diagnose and eliminate the causes of unroutable. For this typical application, a resolution-based local search algorithm to extract unsatisfiable subformulas is integrated into Boolean-based FPGA routing method. The fastest algorithm of deriving minimum unsatisfiable subformulas, called the branch-and-bound algorithm, is adopted to compare with the local search algorithm. On the standard FPGA routing benchmark, the results show that the local search algorithm outperforms the branch-and-bound algorithm on runtime. It is also concluded that the unsatisfiable subformulas play a very important role in FPGA routing real applications.

KEYWORDS

FPGA routing, Boolean satisfiability, unsatisfiable subformula, local search.

1. INTRODUCTION

Many real-world problems, arising in electronic design, equivalence checking, property verification, automatic placement and routing and Auto Test Pattern Generation (ATPG), can be formulated as constraint satisfaction problems, which can be translated into Boolean formulas in conjunctive normal form (CNF). Modern Boolean satisfiability (SAT) solvers, such as Chaff [1] and MiniSAT [2], which implement enhanced versions of the Davis-Putnam-Logemann-Loveland(DPLL) backtrack-search algorithm, are usually able to determine whether a large formula is satisfiable or not. When a formula is unsatisfiable, it is often required to find an unsatisfiable subformula, that is, a small unsatisfiable subset of the original formula. Localizing a small unsatisfiable subformula is necessary to determine the underlying reasons for the failure.
Explaining the causes of unsatisfiability of Boolean formulas is an essential requirement in various fields, such as electronic design automation and formal verification of hardware. A typical paradigm is fixing wire routing in FPGAs, where an unsatisfiable subformula implies that the channel is unroutable. Furthermore, we are usually interested in a small explanation of infeasibility that excludes irrelevant information. There have been considerable researches in deriving the unsatisfiable subformulas. Most of previous works are complete search approaches, essentially on the basis of enhanced versions of the DPLL backtrack-search algorithm. However, existing studies have very little concern regarding unsatisfiable subformulas extraction using incomplete local search method.

David C. Wyld et al. (Eds): ACITY, DPPR, VLSI, WeST, DSA, CNDC, IoTE, AIAA, NLPTA - 2020
pp. 73-81, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.101507

The unsatisfiable sub formulas solver plays a very important role in the automatic routing tool for deciding the routability of FPGA devices. The Boolean-based FPGA detailed routing formulation expresses the routing constraints as a Boolean function which is satisfiable if and only if the layout is routable. The Boolean-based routers have two unique features: One is simultaneous embedding of all nets regardless of net ordering; the other is ability to demonstrate routing infeasibility by proving the unsatisfiability of the generated routing constraint Boolean function. The unsatisfiable subformulas can help the FPGA routing tool to diagnose and eliminate the causes of unroutable. In general, the unsatisfiable subformulas are extracted faster, the tool completes the FPGA routing process more efficiently. Therefore, a fast algorithm of deriving the unsatisfiable subformulas, called the resolution-based local search algorithm [3], is integrated into Boolean-based FPGA routing method. We have compared two optimal algorithms of computing unsatisfiable subformulas, respectively called branch-and-bound algorithm [4] and the resolution-based local search algorithm, on the standard FPGA routing benchmark. The evaluation results show that the local search algorithm strongly outperforms the branch-and-bound algorithm on runtime. It is also shown that the unsatisfiable subformulas can help the tool to quickly diagnose the root causes of unroutability problem and eliminate the fail nets.

The paper is organized as follows. The next section gives the basic definitions and notations of unsatisfiable subformula used throughout the paper. Section 3 surveys the related work on computing unsatisfiable subformulas. Section 4 introduces the principles of the Boolean-based FPGA routing algorithm. Section 5 describes the algorithms of computing the unsatisfiable subformulas. Section 6 shows and analyzes experimental results of two algorithms on the FPGA routing benchmark. Finally, Section 6 concludes this paper.

2. PRELIMINARIES

Resolution is a proof system for CNF (Conjunctive Normal Form) formulas with the following rule:

$$\frac{(L_0 \vee a) \wedge (L_1 \vee \neg a)}{(L_0 \vee L_1)}, \tag{1}$$

where L_0, L_1 are disjunctions of literals. The clauses $(L_0 \vee a)$ and $(L_1 \vee \neg a)$ are the resolving clauses, and $(L_0 \vee L_1)$ is the resolvent. The resolvent of the clauses (a) and $(\neg a)$ is the empty clause ($\perp$). Each application of the resolution rule is called a resolution step. The above resolution step is represented as $((L_0 \vee a) \wedge (L_1 \vee \neg a)) \models (L_0 \vee L_1)$. A sequence of resolution steps, each one uses the result of the previous step or the clauses of the original formula as the resolving clauses of the current step, is called a resolution sequence.

Definition 1. (Boolean Satisfiability) Given a CNF formula $\varphi(V)$, where V is the set of variables, and a Boolean function $F(V)$: $\{0,1\}^n \rightarrow \{0,1\}$, the Boolean satisfiability problem consists of identifying a set of assignments M_v to the variables, such that $F(M_v)=1$, or proving that no such assignment exists.

Lemma 1 A CNF formula φ is unsatisfiable if and only if there exists a finite sequence of resolution steps ending with the empty clause.

It is well-known that a Boolean formula in CNF is unsatisfiable if it is possible to generate an empty clause by resolution sequence from the original clauses. The set of original clauses involved in the derivation of the empty clause is referred to as the unsatisfiable subformula.

Definition 2 (Unsatisfiable subformula) Given a formula φ, ϕ is an unsatisfiable subformula for φ if and only if ϕ is an unsatisfiable formula and $\phi \subseteq \varphi$.

Observe that an unsatisfiable subformula can be defined as any subset, which is infeasible, of the original formula. Consequently, there may exist many different unsatisfiable subformulas, with different number of clauses, for the same problem instance, such that some of these subformulas are subsets of others.

Lemma 2. The set of original clauses involved in the derivation of an empty clause is referred to as the unsatisfiable subformula.

Definition 3 (Minimal Unsatisfiable Subformula) Given an unsatisfiable subformula ϕ for a formula φ, ϕ is a minimal unsatisfiable subformula if and only if removing any clause $\omega \in \phi$ from ϕ implies that $\phi - \{\omega\}$ is satisfiable.

For Boolean formulas in CNF, an unsatisfiable subformula is minimal if it becomes satisfiable whenever any of its clauses is removed. According to the definition, a minimal unsatisfiable subformula has two features: one is unsatisfiable, the other is irreducible, in other words, all of its proper subsets are satisfiable.

Definition 4 (Minimum Unsatisfiable Subformula) Consider a formula φ and the set of all unsatisfiable subformulas for φ: $\{\phi_1, \phi_2, \ldots, \phi_n\}$. Then, $\phi_k \in \{\phi_1, \phi_2, \ldots, \phi_n\}$ is a minimum unsatisfiable subformula iff $\forall \phi_i \in \{\phi_1, \phi_2, \ldots, \phi_n\}$, $1 \le i \le n$: $|\phi_k| \le |\phi_i|$.

According to the definition, a minimum unsatisfiable subformula has the smallest cardinality of all unsatisfiable subsets of a formula. From the above definition, one may conclude that any unsatisfiable formula has at least one minimum unsatisfiable subformula.

We may observed that, the clauses, contained in the intersection of a resolution trace and the original formula, belong to some unsatisfiable subformula. From the lemmas, it is concluded that a refutation proof contains the explanation of infeasibility of the formula. In other words, the causes of unsatisfiability can be derived from the resolution sequence in the sense that removing them will correct the infeasibility. Then we illustrate the process of extracting unsatisfiable subformulas from a Boolean formula according to Lemma 1 and Lemma 2. For example, a CNF formula is

$$\varphi = (\neg a_0) \wedge (a_1) \wedge (a_0 \vee \neg a_1) \wedge (a_0 \vee \neg a_2) \wedge (\neg a_1 \vee a_2) \tag{2}$$

The above formula is refuted by a series of resolution steps ending with the empty clause. Two refutation sequences to affirm the infeasibility of the formula φ are shown as follows:

$$\Gamma_1 = \left\{ \frac{(\neg a_0) \wedge (a_0 \vee \neg a_1)}{(\neg a_1)}, \frac{(\neg a_1) \wedge (a_1)}{\bot} \right\} \tag{3}$$

$$\Gamma_2 = \left\{ \frac{(\neg a_0) \wedge (a_0 \vee \neg a_2)}{(\neg a_2)}, \frac{(a_1) \wedge (\neg a_1 \vee a_2)}{(a_2)}, \frac{(\neg a_2) \wedge (a_2)}{\bot} \right\} \tag{4}$$

From Γ_1, the resolvent $(\neg a_1)$ of the first resolution step serves as one of the resolving clauses of the second step, and the result of the second resolution step is the empty clause. Similarly, the other sequence Γ_2 of resolution steps also arrives at the empty clause. According to Lemma 2, the

original clauses included in the proof of infeasibility belong to the unsatisfiable subformula. More specifically, two unsatisfiable subformulas respectively corresponding to the refutation proofs Γ_1 and Γ_2 are

$$\phi_1 = (\neg a_0) \wedge (a_0 \vee \neg a_1) \wedge (a_1) \quad (5)$$

$$\phi_2 = (\neg a_0) \wedge (a_0 \vee \neg a_2) \wedge (a_1) \wedge (\neg a_1 \vee a_2) \quad (6)$$

In summary, this Boolean formula example demonstrates that the resolution-based local search algorithm to extract the unsatisfiable subformulas is essentially based on Lemma 1 and Lemma 2.

3. BOOLEAN-BASED FPGA ROUTING ALGORITHM

In electronic design automation fields, one of the motivating applications is a Boolean-based FPGA detailed routing formulation on island-style FPGA architecture. The Boolean-based router expresses the routing constraints as a Boolean function which is satisfiable if and only if the layout is routable. The Boolean-based routers have two unique features: One is simultaneous embedding of all nets regardless of net ordering; the other is ability to demonstrate routing infeasibility by proving the unsatisfiability of the generated routing constraint Boolean function.

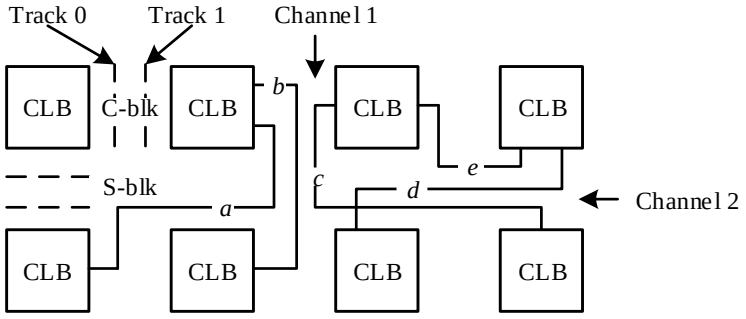

Figure 1. An FPGA routing example

Consider the small FPGA routing problem shown in Figure 1. As depicted in Figure 1, an island-style FPGA consists of a two-dimensional array of Configurable Logic Blocks (CLB), Connection Blocks (C-blk), and Switching Blocks (S-blk). In this figure, there are five nets named *a* through *e* to be routed over an FPGA fabric that has two tracks numbered 0 and 1 in each routing channel. For instance, a net *c* is modeled by two binary variables c_0 and c_1 that indicate its track assignment: $c_i = 1$ expresses that net *C* is assigned to track *i*. With this way of encoding, the routing requirements are now formulated as a set of CNF clauses that fall into one of two categories: Liveness constraints are to ensure that each net is assigned to at least one routing track; Exclusivity constraints are to guarantee that each track is assigned to at most one net.

The resulting set of constraints is listed in Table 1 along with four minimal unsatisfiable subformulas. There are five liveness constraints, one per net, and twelve exclusivity constraints, six for routing channel 1 and six for routing channel 2. The minimal unsatisfiable subformula 1 and minimal unsatisfiable subformula 2 respectively correspond to routing channels 1 and 2. As a means of diagnosing the causes of unroutability, the minimal unsatisfiable subformula 1 pinpoints the conflicting requirements of trying to route three nets *a*, *b* and *c* in a 2-track channel. Similarly, the minimal unsatisfiable subformula 2 indicates the impossibility of three routing nets *c*, *d* and *e* in channel 2. Beyond pointing out a routing channel whose capacity is exceeded, these

minimal unsatisfiable subformulas are able to find a crucial net, namely *c*, which contributes to the unroutability of both channels. This can be located by noting that the *c* variables occur more frequently in these unsatisfiable subformulas than all other variables. This simple example shows that the unsatisfiable subformulas might play a very important role in diagnosing and eliminating the causes of failure.

Table 1. The constraints and minimal unsatisfiable subformulas

Constraints and MUSes		**Expressions in CNF**
Liveness constraints		$L_0=(a_0 \vee a_1)$, $L_1=(b_0 \vee b_1)$, $L_2=(c_0 \vee c_1)$, $L_3=(d_0 \vee d_1)$, $L_4=(e_0 \vee e_1)$
Exclusivity constraints	Channel 1	$E_0=(\neg a_0 \vee \neg b_0)$, $E_1=(\neg a_0 \vee \neg c_0)$, $E_2=(\neg b_0 \vee \neg c_0)$, $E_3=(\neg a_1 \vee \neg c_1)$, $E_4=(\neg a_1 \vee \neg c_1)$, $E_5=(\neg b_1 \vee \neg c_1)$
	Channel 2	$E_6=(\neg c_0 \vee \neg d_0)$, $E_7=(\neg c_0 \vee \neg e_0)$, $E_8=(\neg d_0 \vee \neg e_0)$, $E_9=(\neg c_1 \vee \neg d_1)$, $E_{10}=(\neg c_1 \vee \neg e_1)$, $E_{11}=(\neg d_1 \vee \neg e_1)$
Minimal unsatisfiable subformula 1		$L_0 \vee L_1 \vee L_2 \vee E_0 \vee E_1 \vee E_2 \vee E_3 \vee E_4 \vee E_5$
Minimal unsatisfiable subformula 2		$L_2 \vee L_3 \vee L_4 \vee E_6 \vee E_7 \vee E_8 \vee E_9 \vee E_{10} \vee E_{11}$
Minimal unsatisfiable subformula 3		$L_0 \vee L_1 \vee L_2 \vee L_3 \vee L_4 \vee E_0 \vee E_4 \vee E_5 \vee E_6 \vee E_7 \vee E_{11}$
Minimal unsatisfiable subformula 4		$L_0 \vee L_1 \vee L_2 \vee L_3 \vee L_4 \vee E_1 \vee E_2 \vee E_3 \vee E_8 \vee E_9 \vee E_{10}$

4. ALGORITHMS OF EXTRACTING UNSATISFIABLE SUBFORMULA

There have been many different contributions to research on unsatisfiable subformulas extraction in the last few years, owing to the increasing importance in numerous practical applications. Some research works, based on a relationship between maximal satisfiability and minimal unsatisfiability, have developed some sound techniques for finding a minimum unsatisfiable subformula [4], or all minimal unsatisfiable subformulas [5].

CoreTimmer [6] iterates over each internal node that consumes a large number of clauses and attempts to prove them without these clauses. In [7], the authors presented the algorithms which tracks minimal unsatisfiable subformulas according to the trace of a failed local search run for consistency checking. Two new resolution-based algorithms are proposed in [8]. These algorithms are used to compute a minimal unsatisfiable subformula or, if time-out encountered, a small non-minimal unsatisfiable subformula. Based on these algorithms, seven improvements are proposed in [9], and the experiments have shown the reduction of 55% in run time and 73% in the size of the resulting subformula.

In [10], the authors also proposed two algorithms, one is to optimize the number of calls to a SAT solver, the other is to employ a new technique named recursive model rotation. An improvement to model rotation called eager rotation [11] is integrated in resolution-based minimal unsatisfiable subformulas algorithm. Belov et al. [12] have proposed some techniques to trim CNF formulas using clausal proofs. A new algorithm [13] is to exploit the minimal unsatisfiable subformulas (mus) and minimal correction sets connection in order to compute a single mus and to incrementally compute all muses. An algorithm [14] is proposed to improve over previous methods for finding multiple muses by computing its muses incrementally. The authors [15] aimed to explore the parallelization of partial MUS enumeration. The evaluation results show that the full parallelization of the entire enumeration algorithm scales well.

A technique [16] implements a model rotation paradigm that allows the set of minimal correction subsets to be computed in a heuristically efficient way. The authors [17] proposed a new algorithm for extracting minimal unsatisfiable subformulas and correction sets simultaneously. Liu et al. [18] introduced an algorithm for extracting all MUSes for formulas in the field of

propositional logic and the function-free and equality-free fragment of first-order logic. Luo et al. [19] proposed a method for accelerating the enumeration of MUSes based on inconsistency graph partitioning. A novel algorithm [20] is presented for computing the union of the clauses included in some MUSes, by developing a refined recursive enumeration of MUSes based on powerful pruning techniques.

Bendik et al. [21] firstly approximated MUS counting procedure called AMUSIC, combining the technique of universal hashing with advances in QBF solvers along with a novel usage of union and intersection of MUSes to achieve runtime efficiency. They proposed a novel maximal satisfiable subsets enumeration algorithm called RIME [22]. The experimental results showed that RIME is several times faster than existing tools. In [23], they focused on the enumeration of MUSes, and introduced a domain agnostic tool called MUST. This tool outperforms other existing domain agnostic tools and is even competitive to fully domain specific solutions.

In recent years, the complete methods have made great progress in solving many real life problems including Boolean satisfiability, but they usually cannot scale well owing to the extreme size of the search space. One way to solve the combinatorial explosion problem is to sacrifice completeness, thus some of the best known methods using this incomplete strategy are local search algorithms. In general, the local search strategy starts from an initial solution, which may be randomly or heuristically generated. Then the search moves to a better neighbor according to the objective function, and terminates if the goal is achieved or no better solution can be found. Local search methods are underlying some of the best-performing algorithms for certain types of problem instances, both from an empirical as well as from a theoretical point of view. Consequently, this stochastic strategy is adopted to tackle the problem of finding unsatisfiable subformulas, and in general it has better performance than DPLL-based complete algorithms, especially on 2-SAT and 3-SAT problem instances. According to the rules described in Section 3, the FPGA detailed routing problem can be translated to the Boolean formulas with a number of 2-literal and 3-literal clauses. Therefore, we integrated a resolution-based local search algorithm [3] into the Boolean-based FPGA detailed routing method. The local search algorithm to extract the unsatisfiable subformulas from the Boolean formulas is based on the Lemma 1 and Lemma 2 introduced in Section 2.

5. EXPERIMENTAL RESULTS AND ANALYSIS

To experimentally evaluate the efficiency between two algorithms of computing unsatisfiable subformulas on FPGA routing, we have selected a suit of typical paradigm of the Boolean-based FPGA routing problem. As described above, the FPGA routing benchmark suite is derived from the problem of Boolean-based FPGA detailed routing formulation on island-style FPGA architecture, which is one of the typical applications for unsatisfiable subformulas. The Boolean-based router expresses the routing constraints as a CNF formula which is unsatisfiable if and only if the layout is unroutable. The benchmark includes 10 instances. We have compared the resolution-based local search algorithm with the branch-and-bound algorithm, which is the fastest tool to compute an exactly minimum unsatisfiable subformula. The experiments were conducted on a 2.5 GHz Athlon*2 machine having 2 GB memory and running the Linux operating system. The limit time was 1800 seconds.

The experimental results of the branch-and-bound algorithm and the resolution-based local search algorithm on the 10 formulas of FPGA routing problem are listed in Table 2. Table 2 shows the number of variables (vars) and the number of clauses (clas) for each Boolean formula. The fourth column gives the total number of minimal unsatisfiable subformulas contained in every formula (MUSes). For generating all minimal unsatisfiable subsets we use the CAMUS algorithm [5]. However there are five instances which run out of time, and we mark them with time out in the

table. Table 2 also provides the runtime in seconds (BaBA time) of branch-and-bound algorithm, and the number of clauses (BaBA size) in the derived minimum unsatisfiable subformula. The next three columns present the runtime of the resolution-based local search algorithm in seconds (RbLSA time), and the memory consumption in MB (RbLSA mem), and the size of the unsatisfiable subformula (RbLSA size). The last column shows the percentage of clauses in the unsatisfiable subformula occupying the original formula (Per %).

Table 2. Experimental results on FPGA routing benchmark

Benchmarks	**vars**	**clas**	**MUSes No.**	**BaBA**		**RbLSA**			**Per** (%)
				time	size	time	mem	size	
fpga_routing1	10	17	4	<0.001	9	<0.001	0.12	9	52.9
fpga_routing2	14	25	11	0.02	9	<0.001	0.29	9	36.0
fpga_routing3	18	33	26	0.18	9	0.08	0.63	9	27.3
fpga_routing4	22	41	57	2.63	9	1.2	1.15	9	21.9
fpga_routing5	26	49	120	62.7	9	27.6	3.25	9	18.4
fpga_routing6	30	57	time out	time out		182.5	10.6	9	15.8
fpga_routing7	34	65	time out	time out		358.0	17.5	9	13.8
fpga_routing8	38	73	time out	time out		617.0	23.4	9	12.3
fpga_routing9	42	81	time out	time out		1040.1	28.0	9	11.1
fpga_routing10	46	89	time out	time out		1690.0	36.5	9	10.1

From Table 2, we may observe the following. The resolution-based local search algorithm outperform the branch-and-bound algorithm for all of 10 formulas. For the instances of fpga_routing6 through fpga_routing10, the branch-and-bound algorithm failed to extract the unsatisfiable subformula within the timeout, but the resolution-based local search algorithm succeeded in obtaining it. Moreover, the local search algorithms find the minimum unsatisfiable subformula for each formula of the FPGA routing benchmark suite.

Therefore, the following conclusion can be reached that the runtime of resolution-based local search algorithm strongly exceeds the branch-and-bound algorithm, although the local search algorithm cannot guarantee obtaining the exact minimum unsatisfiable subformulas. The causes include three aspects: The first is that the function of deriving unsatisfiable subformula is coupled tightly with the satisfiability checking procedure of the formula. While the resolution is proceeding, the refutation is recorded, and the parsing tree is constructed simultaneously, then the unsatisfiable subformula is computed very efficiently. The second reason is that the decision of satisfiability is implemented simply and performs many more moves per second. The third cause is there are many powerful heuristics in the local search algorithm, especially for unit clauses and binary clauses. The formulas translated by the Boolean-based FPGA routing problem contain many 2-literal clauses.

From the last column of Table 2, we may observe the following. For 10 formulas, the percentage of clauses in the unsatisfiable subformulas is quite small, in most cases from 10% to 30%. In general, the unsatisfiable subformulas can provide more succinct explanations of infeasibility, and is very valuable for a variety of practical applications. In the automatic routing tool of FPGA chips, the unsatisfiable subformulas can help the tool to quickly diagnose the root causes of unroutability problem, and eliminate the fail nets, and then finish the FPGA routing process much more efficiently.

6. CONCLUSIONS

An unsatisfiable subformula generally provides the most accurate explanation of infeasibility and is valuable for FPGA routing application. The automatic routing process of FPGA devices is very difficult and time-consuming. Therefore, two algorithms of deriving unsatisfiable subformulas, respectively called the resolution-based local search algorithm and the branch-and-bound algorithm, are employed to accelerate the FPGA routing tool. The standard FPGA routing problem instances are adopted as the benchmark. The results show that the resolution-based local search algorithm outperforms the branch-and-bound algorithm on runtime. We have also analyzed that the unsatisfiable subformulas play a very important role in automatic routing tool of FPGA chips.

ACKNOWLEDGEMENTS

The authors would like to thank all peer reviewers for their valuable comments and suggestions. This work is supported by the National Key Research and Development Program of China under grant No. 2018YFB0204301, and the National Natural Science Foundation of China under grant No. 62072464 and U19A2062.

REFERENCES

[1] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang & S. Malik, (2001) "Chaff: engineering an efficient SAT solver", *Proc. of the 38th Design Automation Conf.*, Las Vegas: ACM Press, pp530-535.

[2] N. Een & N. Sorensson, (2003) "An extensible SAT-solver", *Proc. of the 6th Intl. Conf. on Theory and Applications of Satisfiability Testing*, LNCS 2919, Heidelberg: Springer-Verlag, pp502-518.

[3] J. Zhang , S. Shen , & S. Li, (2009) "Tracking Unsatisfiable Subformulas from Reduced Refutation Proof", *Journal of Software*, Vol. 4, No. 1, pp42-49.

[4] M.H. Liffiton, M.N. Mneimneh, I. Lynce, Z.S. Andraus, J.P. Marques-Silva, & K.A. Sakallah, (2009) "A branch and bound algorithm for extracting smallest minimal unsatisfiable formulas", *Constraints*, Vol. 14, No. 4, pp415-442.

[5] M.H. Liffiton, & K.A. Sakallah, (2008) "Algorithms for computing minimal unsatisfiable subsets of constraints", *Journal of Automated Reasoning*, Vol. 40, pp1-30.

[6] R. Gershman, M. Koifman, & O. Strichman, (2008) "An approach for extracting a small unsatisfiable core", *Formal Methods in System Design*, Vol. 33, No. 1, pp1-27.

[7] E. Gregoire, B. Mazuer, & C. Piette, (2009) "Using local search to find MSSes and MUSes", *European Journal of Operational Research*, Vol. 199, No. 3, pp640-646.

[8] A. Nadel, (2010) "Boosting minimal unsatisfiable core extraction", *Proc. of 10th Intl. Conf. Formal Methods in Computer Aided Design*, pp221-229.

[9] V. Ryvchin & O. Strichman, (2011) "Faster extraction of high-level minimal unsatisfiable cores", *Proc. of 14th Intl. Conf. Theory and Applications of Satisfiability Testing*, pp174-187.

[10] A. Belov, I. Lynce, & J. Marques-Silva, (2012) "Towards efficient MUS extraction", *Journal AI Communications*, Vol. 25, No. 2, pp97-116.

[11] A. Nadel, V. Ryvchin, & O. Strichman, (2013) "Efficient MUS extraction with resolution", *Proc. of 13th Intl. Conf. Formal Methods in Computer Aided Design*, pp197-200.

[12] Anton Belov, Marijn Heule, & Joao Marques-Silva. (2014) "MUS extraction using clausal proofs", *Proc. of 17th Intl. Conf. on Theory and Applications of Satisfiability Testing*, pp48-57.

[13] F. Bacchus & G. Katsirelos, (2015) "Using minimal correction sets to more efficiently compute minimal unsatisfiable sets", *Proc. of the 27th Intl. Conf. on Computer Aided Verification*, pp70-86.

[14] F. Bacchus & G. Katsirelos, (2016) "Finding a collection of MUSes incrementally", *Proc. of 13th Intl. Conf. on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp35-44.

[15] W. Zhao & M. H. Liffiton, (2016) "Parallelizing partial MUS enumeration", *Proc. of IEEE 28th Intl. Conf. on Tools with Artificial Intelligence*, pp464-471.

[16] E. Gregoire & Y. Izza, (2018) "Boosting MCSes Enumeration", *Proc. of the 27th Intl. Joint Conf. on Artificial Intelligence*, pp1309-1315.
[17] N. Narodytska, N. Bjorner, M.C. Marinescu, & M. Sagiv, (2018) "Core-guided minimal correction set and core enumeration", *Proc. of the 27th Intl. Joint Conf. on Artificial Intelligence*, pp1353-1361.
[18] S. Liu & J. Luo, (2018) "FMUS2: An efficient algorithm to compute minimal unsatisfiable subsets", *Proc. of 2018 Intl. Conf. on Artificial Intelligence and Symbolic Computation*, pp104-118.
[19] J. Luo & S. Liu, (2019) "Accelerating MUS enumeration by inconsistency graph partitioning", *Science China Information Sciences*, Vol. 62, pp212104.
[20] C. Mencia, O. Kullmann, A. Ignatiev, & J. Marques-Silva, (2019) "On computing the union of MUSes", *Proc. of 22nd Intl. Conf. on Theory and Applications of Satisfiability Testing*, pp211-221.
[21] J. Bendik & M.S. Kuldeep, (2020) "Approximate counting of minimal unsatisfiable subsets," *Proc. of the 32nd Intl. Conf. on Computer Aided Verification*, pp1-23.
[22] J. Bendik & C. Ivana, (2020) "Rotation based MSS/MCS enumeration", *Proc. of the 23rd Intl. Conf. on Logic for Programming, Artificial Intelligence and Reasoning*, pp120-137.
[23] J. Bendik & C. Ivana, (2020) "MUST: minimal unsatisfiable subsets enumeration tool", *Proc. of the 26th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, pp135-152.

AUTHORS

Zhang Jianmin was born in China in 1979. He is an associate professor in computer science at National University of Defense Technology. He received Ph. D. degree in computer science from National University of Defense Technology, Changsha, China, in 2008. His major fields of study include SAT-based formal verification and FPGA acceleration algorithms.

Li Tiejun was born in China in 1977. He is a professor in computer science at National University of Defense Technology. He received Ph. D. degree in computer science from National University of Defense Technology, Changsha, China, in 2005. His research interests include functional verification and high performance CPU design.

Ma Kefan was born in China in 1985. He is a Ph.D in computer science at National University of Defense Technology. He received Ph. D. degree in computer science from National University of Defense Technology, Changsha, China, in 2019. His research interests include FPGA accelerated SAT solver and formal verification.