Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

An assertion graph based abstraction algorithm in GSTE and Its application

Desheng Zheng [a,b,1], Xiaoyu Li [b,*], Guowu Yang [d], Hai Wang [e], Lulu Tian [c,1]

[a] School of Computer Science, Southwest Petroleum University, Chengdu 610500, China
[b] School of Information & Software Engineering, University of Electronics Science and Technology of China, Chengdu, 611731, China
[c] School of Automation Engineering, University of Electronics Science and Technology of China, Chengdu, 611731, China
[d] School of Computer Science & Engineering, University of Electronics Science and Technology of China, Chengdu, 611731, China
[e] School of Microelectronics & Solid State Electronics, University of Electronics Science and Technology of China, Chengdu, 611731, China

ARTICLE INFO

Keywords:
GSTE
State explosion
Assertion graph
Abstraction algorithm

ABSTRACT

Generalized Symbolic Trajectory Evaluation (GSTE) is an alternative model checking technique based on particular automata to specify the properties. Despite the success of GSTE, its state explosion remains a major hurdle when applying it to large industrial designs. This paper presents two efficient theoretical underpinning abstraction algorithms based on assertion graph to combat the state explosion problem. We implement these two algorithms as a prototype system for discrete models. Experimental results show that the prototype system is 10× faster than the former without abstraction.

1. Introduction

After decades of research and development, microprocessor has evolved into complex system, which is composed of components such as out-of-order execution, register-renaming, pipelining, speculative, and multi-level caching. It is a big challenge to ensure the correctness and fault tolerance in such a complex system. Formal verification [1,2] techniques based on the mathematical methodology have been proposed to resolve these problems, with functions including model checking [3,4], theorem proving [5,6], and equivalence proving [7,8].

Symbolic trajectory evaluation (STE) [9] is one of the efficient symbolic model checking (SMC) algorithms, which is especially suited to verifying the large data-path designs' properties and also has been proven in verifying fairly large industrial hardware designs [10]. STE has been used in word-level verification [9], However, the property with indefinitely long time interval cannot be expressed in STE. Thus, generalized symbolic trajectory evaluation (GSTE) [11,12] has been proposed to resolve this weakness. GSTE is an advanced technique developed by combining STE and traditional SMC, which will be introduced in Section 2 in details.

This paper presents two theoretical underpinning abstraction algorithms, *Minimum Preserved Abstraction* (MPA) and *Optimized Minimum Preserved Abstraction* (OMPA), based on assertion graphs of GSTE [11,12]. In order to verify if a model satisfies an assertion graph, the classical GSTE procedure checks whether $\varphi(e) \subseteq cons(e)$ [11]. If such condition is not satisfied, a counterexample will be listed which is a path on the assertion graph. As one of the classical model checking methods, GSTE is based on reduced order binary decision diagrams techniques (ROBDD) [13]. When the state count is up to 10^{120} [14], state explosion will be its bottleneck. Our approach involves abstracting states according to $cons(e)$ with two new algorithms named MPA and OMPA. These two algorithms search the minimum intersection from all the $cons(e)$ and split them into independent sets, which are abstracted into the unique new states. We implement these two algorithms as a prototype system for discrete models. Experimental results show that our approach is highly efficient and feasible.

The structure of this paper is organized as follows: Section 2 introduces the related work of GSTE, assertion graph and abstraction. Section 3 gives a brief review of some necessary definitions and notations in GSTE, such as model, assertion graph, abstraction function and strong satisfiability definitions. Section 4 defines a new abstraction function different from the abstraction function defined by *Yang* [11] and presents three theorems as the theoretical basis for the next two algorithms (MPA and OMPA). Section 5 tests feasibility and efficiency of our algorithms for time and memory usage. Section 6 concludes the paper and briefly comments on future work.

* Corresponding author.
E-mail addresses: xiaoyuuestc@uestc.edu.cn (X. Li), desheng619@gmail.com (D. Zheng).
[1] Desheng Zheng and Lulu Tian have contributed equally to this work.

https://doi.org/10.1016/j.vlsi.2018.03.009
Received 15 December 2017; Accepted 17 March 2018
Available online XXX
0167-9260/© 2018 Published by Elsevier B.V.

2. Related work

2.1. STE and GSTE

Symbolic Trajectory Evaluation (STE) [15] is the main alternative to symbolic model checking. It has shown great promise in verifying medium to large scale industrial hardware designs with a high degree of automation at both the gate level and the transistor level. But the specification language of STE has limited expressiveness where only properties over finite time intervals are allowed. *Jain* [16] developed a generalized STE algorithm to check this form with its assertions, which was mathematically clarified by *Chou* [17].

Generalized Symbolic Trajectory Evaluation (GSTE) [11,12] is a significant extension of STE that has the power to verify all ω-regular properties but at the same time preserve the benefit of the original STE. As a powerful, new model-checking approach, GSTE combines the industrially-proven scalability and capacity of STE with the expressive power of temporal-logic model checking. *Yang* presented a case study on FIFO verification to illustrate the strength of GSTE and demonstrate its methodology in specifying and verifying large scale designs [18].

2.2. Assertion Graph

The three key aspects of GSTE are: assertion graphs for property, a symbolic simulation based on model checker, and a four-value lattice based on circuit abstraction [19]. An assertion graph is a labeled flow graph where each edge in the graph is labeled with an antecedent/consequent pair. The assertion graph provides sequential stimuli to drive the symbolic simulation of the circuit, and specify the responses expected from the circuit. *Yang et al.* have presented a new model characterization for an assertion graph with important properties [20]. *Hu et al.* have done some work on reasoning about assertion graphs themselves [21]. In our previous work [20,22], we have presented a novel implication technique relying on direct Boolean reasoning on each edge (and vertex) of an assertion graph for assertion graphs, thus avoiding the reach ability computation in GSTE. Compared with the above, we do not change the structure of the assertion graph, but only abstract the states based on assertion graphs in this work.

2.3. Abstraction

Due to state explosion problems, numerous methods have been proposed to reduce the number of states in a model, such as symmetry reductions [23–25], partial order reductions [26,27], and abstraction techniques [28–30]. Abstraction techniques are considered to be the most flexible and regular method to handle the state explosion problem. *Clark et al.* [31] divide them into several classes by how they control the information loss after abstraction: *over-approximation* [28,31], *under-approximation* [32,33], *abstract interpretation* [29,34] and *3-valued logics* [35,36]. Besides the above methods, many optimizations have been conceived, such as techniques based on ALL-SAT [37–39] and later extended to the SMT case [40]. *Clark et al.* [31] extend a new counterexample-guided abstraction technique based on their general framework introduced in Ref. [28]. *Graf* and *Saïdi* have proposed the *predicate abstraction* techniques that abstract an infinite system (concrete system) into a finite system (abstract system) [30]. For such predicate abstraction techniques, the user should supply the predicates and properties [29,41]. Our method abstracts states according to assertion graph automatically without any predicates.

GSTE is a model checking technique and greatly benefits from advanced abstraction methods. *Yang* and *Seger* have presented an abstract interpretation technique to reduce states [42]. Currently, abstraction is mostly a manual process that requires considerable creativity and insight [31]. To generalize GSTE to large industrial designs, automatic abstracting techniques are needed. *Chen et al.* [43] give automatic abstraction refinement algorithms that can quickly converge to an appropriate level of abstraction. AutoGSTE [43] is the first automatic abstraction refinement framework for GSTE that completely eliminates false negatives caused by abstraction imprecision, which is a comprehensive approach to automatic abstraction refinement for GSTE. According to our algorithms, we can also realize the automatic abstraction. *Chen et al.* present a suite of optimizations targeting automatic abstraction refinement for GSTE [44]. This paper focuses on the abstraction based on assertion graphs to reduce the time and memory usage.

3. Preliminaries

This section gives a brief review of the necessary definitions and notations in GSTE from Ref. [11], such as, assertion graph, model, satisfiability and abstraction function.

3.1. Assertion graph

Definition 1. *(Assertion Graph) [11] An assertion graph is a quintuple $G = (V, v_I, E, ant, cons)$ where V is a finite set of vertices, $v_I \in V$ is the initial vertex, $E \subseteq V \times V$ is a set of edges, satisfying $\forall u \in V, \exists v \in V, (u, v) \in E$, $ant : E \to 2^S$ labels each edge $e \in E$ with an antecedent $ant(e)$, $cons : E \to 2^S$ labels each edge $e \in E$ with a consequent $cons(e)$.*

Remark 1. *$G = (V, v_I, E, ant, cons)$ is a directed graph. The ant and cons are the two functions of the assertion graph, which are labeled on the edge. Note that $ant(e)$ and $cons(e)$ are the subset of S.*

Definition 2. *(Path) [11] A path in the assertion graph is an edge sequence ρ such that for all $1 \le i < |\rho|$, $\rho[i]$ ends at a vertex from which $\rho[i+1]$ starts.*

Definition 3. *(start(e), end(e), in(e), out(e)) [11] Given all the edges $e \in E$, start(e) denotes the vertex e starts from; end(e) denotes the vertex e ends at; in(e) denotes the set of the edges ending at start(e); out(e) denotes the set of the edges starting from end(e).*

Remark 2. *In assertion graph $G = (V, v_I, E, ant, cons)$, we can know that for all $e \in E$, $in(e) = \{e_i | end(e_i) = start(e)\}$, $start(e) = \{u | (u, v) = e\}$, $end(e) = \{v | (u, v) = e\}$, $out(e) = \{e_i | end(e) = start(e_i)\}$.*

3.2. Model

Definition 4. *(Transition Relation) [11] A relation $T \subseteq S \times S$ is a transition relation if $\forall s \in S, \exists s' \in S, (s, s') \in T$, where S is a non-empty set of finite states.*

Definition 5. *(Transition Relation Induced Model) [11] The model M induced by the transition relation T is the pair $(pre, post)$ where the pre-image transformer $pre: 2^S \to 2^S$ is defined as $pre(Q) = \{s | \exists s' \in Q, (s, s') \in T\}$ for all $Q \in 2^S$, and the post-image transformer $post: 2^S \to 2^S$ is defined as $post(Q) = \{s' | \exists s \in Q, (s, s') \in T\}$ for all $Q \in 2^S$.*

Remark 3. *Model $M = (S, T)$ can be expressed as $M = (pre, post)$, which is a directed graph. Pre and post are two functions of model M. Note that $pre(s) = pre(\{s\}), \forall s \in S$ and $post(s) = post(\{s\}), \forall s \in S$.*

Definition 6. *(Trace) [11]. A trace in $M = (pre, post)$ is a state sequence σ such that $\sigma[i+1] \in post\{\sigma[i]\}$, for all $1 \le i \le |\sigma|$.*

3.3. Satisfiability

Definition 7. *(Path Satisfiability) [11] Let $G = (V, v_I, E, ant, cons)$ be an assertion graph, and let $M = (pre, post)$ be a model. Given an edge labeling $\gamma : E \to 2^S$ where γ is either ant or cons, A trace σ in M satisfies a path ρ of the same length under γ, denoted by $(M, \sigma) \vDash_\gamma (G, \rho)$, iff $\sigma[i] \in \gamma(\rho[i]), 1 \le i \le |\sigma|$. A trace satisfies a path, denoted by $(M, \sigma) \vDash (G, \rho)$, iff $(M, \sigma) \vDash_{ant} (G, \rho) \Rightarrow (M, \sigma) \vDash_{cons} (G, \rho)$.*

Download English Version:

https://daneshyari.com/en/article/11020902

Download Persian Version:

https://daneshyari.com/article/11020902

Daneshyari.com