Chapter 8
SAT Sweeping with Local Observability Don’t-Cares

Qi Zhu, Nathan B. Kitchen, Andreas Kuehlmann, and Alberto Sangiovanni-Vincentelli

Abstract Boolean reasoning is an essential ingredient of electronic design automation. AND-INVERTER graphs (AIGs) are often used to represent Boolean functions but have a high degree of structural redundancy. SAT sweeping is a method for simplifying an AIG by systematically merging graph vertices from the inputs toward the outputs using a combination of structural hashing, simulation, and SAT queries. Due to its robustness and efficiency, SAT sweeping provides a solid algorithm for Boolean reasoning in functional verification and logic synthesis. In previous work, SAT sweeping merges two vertices only if they are functionally equivalent. In this chapter we present a significant extension of the SAT-sweeping algorithm that exploits local observability don’t-cares (ODCs) to increase the number of vertices merged. We use a novel technique to bound the use of ODCs and thus the computational effort to find them, while still finding a large fraction of them. Our reported results based on a set of industrial benchmark circuits demonstrate that the use of ODCs in SAT sweeping results in significantly more graph simplification with great benefit for Boolean reasoning with a moderate increase in computational effort.

8.1 Introduction

Boolean reasoning is a key part of many tasks in computer-aided circuit design, including logic synthesis, equivalence checking, and property checking. Circuit graphs such as AND-INVERTER graphs (AIGs) [8] are often used to represent
Boolean functions because their memory complexity compares favorably with other representations such as binary decision diagrams. In many Boolean reasoning problems, circuit graphs have a high degree of structural redundancy [9]. The redundancy can be reduced by the application of SAT sweeping [7]. SAT sweeping is a method for simplifying an AND-INVERTER graph by systematically merging graph vertices from the inputs toward the outputs using a combination of structural hashing, simulation, and SAT queries. Due to its robustness and efficiency, SAT sweeping provides a solid algorithm for Boolean reasoning in functional verification and logic synthesis.

In previous work, SAT sweeping merges two vertices only if they are functionally equivalent. However, functional differences between vertices are not always observable at the outputs of a circuit. This fact can be exploited to increase the number of vertices merged. In this chapter we present a significant extension of the SAT-sweeping algorithm that uses observability don’t-cares (ODCs) for greater graph simplification. Taking observability into account not only increases the effectiveness of SAT sweeping but also increases its computational expense. In order to find a balance between effectiveness and efficiency, we introduce the notion of local observability, in which only paths of bounded length are considered.

When observability is taken into account, the equivalence-class refinement approach of the original SAT-sweeping algorithm cannot be used. We introduce a new method of comparing simulation vectors to identify merging candidates. Although the number of comparisons is theoretically quadratic, on average near-linear complexity is observed in our results.

SAT sweeping interleaved with SAT queries to check a property can be utilized as an efficient engine for equivalence and bounded property checking [7, 9] (see also Section 8.4.5). Similarly, SAT sweeping can be applied to perform the equivalence-class refinement in van Eijk’s algorithm for sequential equivalence checking [4]. In each iteration it can prove or disprove the functional equivalences of the candidates. In all these applications, the use of ODCs for SAT sweeping described in this chapter can increase the number of vertices merged and thus improve their overall reasoning power.

This chapter is structured as follows: Previous work is discussed in Section 8.2. Section 8.3 introduces some preliminary concepts, including AIGs and SAT sweeping. Section 8.4 explains the details of SAT sweeping with local ODCs. Section 8.5 presents the experimental results, and Section 8.6 contains conclusions.

### 8.2 Previous Work

The combined application of random simulation and satisfiability queries for finding functionally equivalent circuit components has been proposed in multiple publications [2, 6, 10–12]. A particular implementation on AIGs that combines these methods with structural hashing and circuit rewriting was described in [7] for simplifying the transition relation for bounded model checking (BMC).