Abstract

With the size increase of VLSI, current designs are firstly written in high-levels, such as system-level, behavioral level, or register transfer level (RTL). High-level designs are typically verified by simulation. However, since simulation can only check patterns being input, some design bugs in corner cases may not be detected with it. Then, formal verification is used as a complement of simulation for such a case.

Currently, two problems can be considered on formal verification of high-level design. One is about performances of verification methods and tools. The other is the high-barrier to apply formal verification methods to actual designs. In this thesis, four methods are proposed for these problems. The first two methods improve verification performances, and the other two methods related to interfaces or preprocesses of formal verification methods. The first two methods are based on an approach which separates control and data portions in designs. Then, control portions and data portions can be analyzed separately, and word-level methods such as symbolic simulation can be applied effectively.

The first proposed method improves bounded model checking by decomposing one large bounded model checking into small pieces. It is very difficult for traditional bounded model checking methods which can only be verified with short bounds to detect deep bugs. In the proposed method, since the bound of each decomposed bounded model checking is small, the computation amount can be dramatically reduced in successful cases. In addition, symbolic simulation is applied to a control path of each counter example to support the connections between those decomposed bounded model checkings. When a connection fails, the former bounded model checking is retried after refining the condition not to get similar counter examples.

Experimental results showed that the proposed method can improve the performance of bounded model checking even with the simplest two-level method.

The second method proposed in this thesis improves equivalence checking between designs before and after behavioral optimization or high-level synthesis. The proposed method applies a preprocess that makes the data portions of the target designs identical. This is performed by tentatively synthesizing behavioral designs into virtual controllers and virtual datapaths. When the target designs are designs before and after high-level synthesis, the virtual datapath is identical to the datapath of the RTL design. When datapaths of two designs are identical, the same control signals are guaranteed to be equivalent in bit-level. Then such control signals can be replaced with uninterpreted functions, and word-level equivalence checking techniques can be applied with bit-level accuracy. In addition, a word-level rule-based equivalence checking method is proposed. The method uses pre-defined rules of equivalence to propagate input equivalences which are given by users to outputs. Since the rule based approach topologically traverses control FSMs, designs which include many conditional branches and loops can be verified faster than symbolic simulation based methods.

The third method proposed in this thesis is a preprocess for hardware/software co-design to solve the three problems in formal verification of hardware/software co-design in lower level than system-level, such as their size, the difference of hardware and software representations, and the interactions between hardware and software portions. The proposed method translates both hardware and software portions into a set of concurrent Finite State Machine with Datapaths (FSMDs). After the translation, the interactions between hardware and software portions are abstracted. Then, a sequentialization method which converts concurrent FSMDs into a single sequential FSMD and handles interruptions is applied. After the sequentialization, control states which do not have data dependences each other are merged. The experimental results showed that the proposed method could make formal verification more than 20 times faster than existing methods.

The last method proposed in this thesis is an useful intermediate representation of high-level designs for verification. In the proposed intermediate representation ExSDG, complicated syntax elements and structures are removed in preprocess steps. Since ExSDG has different representation levels correspond to untimed behavioral level, timed behavioral level, and register transfer level, respectively, various existing design representations in high-level can be directly translated into ExSDG. Therefore, verification tool developers only have to deal with ExSDG to support those representations. In addition, System Dependence Graph (SDG) and Control Flow Graph are integrated with Abstracted Syntax Tree (AST) in ExSDG, and users can directly access such information from the AST tree. An SDG edge shows a dependency relation between two portions of a design. Many researches use ExSDG as a tool implementation environment, and this fact shows the effectiveness of ExSDG.

With the four methods proposed in this thesis, formal verification in high-level can achieve more performances, wider range of designs can be verified with them, and tool implementations of them will be easier.