論文の内容の要旨

論文題目　Formal Verification of High-Level Design Based on Control/Data Separation
(高位設計に対する制御と演算の分離による形式的検証)

氏　名　西原　佑

With the size increase of VLSI, currently, designs are firstly written in high-levels, such as system-level, behavioral level, or register transfer level (RTL). Then, incremental design refinements and syntheses are performed to the designs to translate or convert them into low levels. High-level designs are typically verified by simulation. However, since simulation can only check the patterns which are input, some design bugs in corner cases may not be detected with it. Then, formal verification technique which can perform exhaustive analysis is used as a complement of simulation for such a case.

Currently, two problems can be considered on high-level design formal verification. One is the performance of verification methods and tools. More efficient methods are always required since the computation amount of formal verification methods increases exponentially with the size of target designs, and large designs still cannot be fully verified. The other is the high-barrier to apply formal verification methods to actual designs. Though most formal verification methods target on simple representation such as finite state machine (FSM), the industry designs includes designs represented in various representations, such as SystemC and SpecC in system level, and Verilog-HDL and VHDL in register transfer level (RTL). Since those representations are complicated, they must be translated into more simple representations to apply formal verification methods. Moreover, some designs include not only hardware portions but also software portions. Such hardware/software co-designs are other examples that difficult to be verified formally.

Four methods are proposed in this thesis for those problems. The first two methods improve verification performance, and the other two methods related to the interface or preprocess of formal verification methods. The first two methods 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 method proposed is an improvement of bounded model checking method by decomposing one large bounded model checking into small pieces. Model checking is a formal verification method which proves that a target design satisfies a property. Target designs are given as FSMs, and those states are traversed to check the reachability from the initial states to the states where the properties are violated. Even with symbolic model checking technique, typically designs include flip-flops up to 100 can be verified since the number of states in an FSM increases exponentially with the size of the target design. Bonded model checking is an extension of model checking. It limits the number of searched states by giving a fixed bound which is a maximum number of state transitions from the initial states. If the bound is small, even large designs can be verified since the number of searched states is exponential to the bound. However, verifications with large bounds are still difficult and deep bugs that exist in states far from the initial states cannot be detected by bounded model checking. The proposed method in this thesis is a heuristic to solve such a problem, and other heuristics and efficient methods, such as abstraction-refinement and incremental SAT solving, can be applied together with the proposed method.

The proposed method mainly consists of three steps. The first step applies bounded model checking without considering the initial states which means all states in the state space are considered as initial states. This can be realized by removing the initial state condition from the bounded model checking formula. If no property violations are found in this step, there are no states violating the property. Then the property is proved without applying the further steps. On the other hand, if a property violation is found, a counter example from a state to the state where the property is violated is generated. Though the final goal is to check the reachability from the initial states to the first state of the counter example, symbolic simulation is applied on the counter example as a preprocess step to enhance the performance. This symbolic simulation generates a condition to violate the property on the control path of the counter example. Since the generated condition corresponds to multiple states, the final step, the reachability checking from the initial states, can be much easier. This reachability analysis can be performed by replacing the property condition in bounded model checking with the generated condition. If the condition is reachable from the initial states, a counter example is generated. Such a counter example can be concatenated with the counter example generated at the first step after it is modified with the symbolic simulation result. Then a complete counter example from an initial state to a

state where the property is violated is generated. On the other hand, if the condition is not reachable, the condition is proved unreachable from the initial states. In such a case, the initial state condition is refined not to include the unreachable condition. This corresponds to removing some states from the initial state space which initially covers all states. With the symbolic simulation step, multiple states instead of just a single state can be removed from the initial state condition. This refinement is applied iteratively until a complete counter example is found or no states remain in the initial state condition. Since the third step, reachability analysis from the initial states is also a bounded model checking, that step can be decomposed by applying this method recursively to realize arbitrary numbers of decompositions (levels). 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. Equivalence checking is another formal verification method which compares two designs and proves the equivalence between them. Conventional equivalence checkings in high-level are performed in bit-level where each signal or operation is handled as a bit-array or bit-wise operation. Then combinational or sequential equivalence checking methods for RTL and gate-level designs are applied. However, the computation amount of such bit-level analyses is too high to verify even middle size designs. To overcome such a problem, word-level techniques where each signal or operation in designs is handled as a symbol or symbolic expression are performed. This technique is called symbolic simulation. In symbolic simulation, identical symbols or symbolic expressions can be decided to be equivalent without interpreting their actual bit-level functions. Then large designs can be handled. However, computation amount or memory usage of symbolic simulation is doubled for each control conditional branch, and it also requires the guarantee that each pair of corresponding symbols or operators in two designs is equivalent in bit-level.

To solve those problems in symbolic simulation, 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. On the other hand, when the target designs are designs before and after behavioral synthesis, an arbitrary datapath is given, and both the virtual datapath will be identical to the given datapath. NISC compiler can be used for these syntheses since NISC compiler can

generate a set of control signals for an arbitrary datapath, and such a set of control signals can be directly translated into a control FSM. 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 the proposed method verifies two designs with an identical datapath by a new word-level equivalence checking method. That method uses pre-defined rules of equivalence to propagate input equivalences which are given by users to outputs. If the output equivalences are proved, then the two designs are guaranteed to be equivalent. 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 experimental results showed that the proposed rule-based method can actually be used as a complement of symbolic simulation methods.

The third method proposed in this thesis is a preprocess for hardware/software co-design. There are mainly three problems on formal verification of hardware/software co-design in lower level than system-level. The first is the size problem. Since hardware/software co-designs includes both hardware and software portions, sum of the hardware and software portions are larger than either the hardware or software portion. The second is the difference of hardware and software representations. Typically, software is written in a program code, and hardware is written in RTL. The third is the interaction between hardware and software portions. Such interactions are mainly done by memory-mapped I/O and interruption driven I/O. Both of them must be considered on formal verification.

The proposed method translates both hardware and software portions into a set of concurrent Finite State Machine with Datapaths (FSMDs). Since both the portions are translated into a same representation, the second problem is solved. After the translation, some simplification and state reduction techniques are applied to the FSMDs. The first one is the abstraction of the interaction between hardware and software portions. Each memory-mapped I/O transaction is replaced with an access to a shared variable. Interruption related portions are removed from the FSMD after the information of interruptions is stored. Then a reduction technique, sequentialization, is applied to the FSMDs. The sequentialization method converts concurrent FSMDs into a single sequential FSMD without changing the function of the design. The original sequentialization method was proposed for SpecC. The proposed method extends it, and can handle interruption relations. In the proposed method, synchronization or execution

order relations among concurrent FSMDs include interruption relations are solved by decision procedure such as SMT solver. After the synchronization, the last reduction technique which merges control states which do not have data dependence each other. The experimental results showed that the proposed method could make formal verification more than 200 times faster.

The last method proposed in this thesis is a useful intermediate representation of high-level designs for verification. Front-end parts are important portion in formal verification tool implementation since formal verification techniques can only be applied to simple representations. Then formal verification unfriendly portions must be removed from the original representation. In the proposed intermediate representation ExSDG, such complicated syntax and structures are removed in preprocess steps. Since ExSDG has different versions correspond to untimed behavioral level, timed behavioral level, and register transfer level, respectively, various existing design representations, such as SystemC, SpecC, SystemVerilog, Verilog-HDL, and VHDL, can be directly translated into ExSDG. Then verification tool developers only have to deal with ExSDG to support those representations. In addition, System Dependence Graph (SDG) and Control Flow Graph (CFG) are integrated with Abstracted Syntax Tree (AST) in ExSDG, and users can directly access such information from the AST tree. SDG edges show dependency relations between two portions of a design. Related portions can be extracted by traversing SDG, and it can reduce computation amount of formal verification methods. Since similar method can be performed on net-list representation which is a common representation for verification and analysis of gate-level and RTL designs, SDG can be considered as a corresponding representation in high-level. 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 performance, wider range of designs can be verified with them, and tool implementation of them will be easier.