Enhancing BMC-based Protocol Verification Using Transition-By-Transition FSM Traversal

Minh D. Nguyen, Dominik Stoffel, Wolfgang Kunz

nguyen@eit.uni-kl.de

Abstract: We present a technique to automatically derive reachability information from designs to enhance Bounded Model Checking style verification. The method is well suited for typical protocol applications where the operation is controlled by a main finite state machine (FSM) interacting with a hierarchy of sub-FSMs. The algorithm traverses the global state space of the design based on single transitions of the main finite state machine. This *transition-by-transition* decomposition of the state space distinguishes our approach from previous ones. The experimental results clearly show the value of the new method. In our experiments, false negatives could be completely avoided and the overall verification effort was drastically reduced.

1 Introduction

Implementations of protocols and protocol controllers are typically designed as a hierarchy of interacting finite state machines (FSMs). There is usually one (top-level) FSM which controls the overall behavior of the design and assigns sub-tasks to the other FSMs. The global state space of such designs contains many unreachable states. In verification methodologies based on Bounded Model Checking (BMC) [BCC+03], the *request/acknowledge* properties often specify unreachable states in their *request* part. Due to this, false negatives are very likely to occur. In practice, verification engineers and designers need to manually decompose the state space of these designs and supply the verification tool with additional reachability constraints derived from the design.

In this paper we present a technique to automate this process. The method is capable of deriving reachability information from the design which enables BMC provers to complete the verification task. It is specially suited to a design style where a particular FSM controls the design behavior and interacts with a hierarchy of sub-FSMs. The core algorithm analyzes correlations between the main FSM and the sub-FSMs. It caclulates sets of reachable states corresponding to main states of the design. This can be viewed as a decomposition of the global state space into sets of states corresponding to main states.

The sets of reachable states are calculated exactly by considering state transitions in the main FSM (*main transitions*). Our traversal algorithm therefore is named *Transition By Transition* (TBT). By considering the main transitions the transition relation of the circuit can be greatly simplified. The memory requirements for manipulating the transition relation are greatly reduced. Using the main FSM to guide the decomposition procedure clearly distinguishes our method from previous decomposition methods such as [CCLQ97, CHM+96, GDHH98, SSTV04][1].

[1] For reason of space, some references are omitted. Refer to [NSK05] for complete references

2 Decomposing the state space by the main FSM

Let C be a sequential circuit with a set of *present-state variables* $V = \{v_1, \ldots, v_m\}$. A circuit C is modeled as an encoded *finite state machine* whose the *state transition graph* (STG) is a graph (S_0, S, R) where the vertices are given by the set of states S, the source vertices correspond to the set of initial states S_0, the edges are given by the transition relation $R \subseteq S \times S$. The set of states $S \subseteq B^m$ is encoded by the state variables. The value of a state variable v_i in a state s is denoted by $s(v_i)$. We consider a sub-vector U of state variables, $U = (v_j, \ldots, v_k)$, where $1 \leq j < k \leq m$. The boolean vector of the values of the state variables in U for state s is denoted by $s(U)$.

Definition 1 *Let the main FSM be encoded by a sub-vector $\hat{V} = (v_p, \ldots, v_q)$ of V with $1 \leq p < q \leq m$. The STG of the main FSM of the circuit is a tuple $(\hat{S}, \hat{S}_0, \hat{R})$ where*

- $\hat{S} = \{\hat{s} \in B^{|\hat{V}|} \mid \exists s \in S : s(\hat{V}) = \hat{s}\}$ *is the set of main states*
- $\hat{S}_0 = \{\hat{s}_0 \in \hat{S} \mid \exists s_0 \in S_0 : s_0(\hat{V}) = \hat{s}_0\}$ *is the set of initial main states*
- $\hat{R} = \{(\hat{s}_1, \hat{s}_2) \in \hat{S}^2 \mid \exists s_1, s_2 \in S : (s_1, s_2) \in R \wedge s_1(\hat{V}) = \hat{s}_1 \wedge s_2(\hat{V}) = \hat{s}_2\}$ *is the main transition relation.*

The main FSM of the circuit can be easily extracted from the RTL description (commercial tools exist for this task). Note that the extraction technique must yield a main FSM conforming to Definition 1, i.e., all main states must be identified and no state must be missed. Since the main FSM is very small, in practice, this is easy to guarantee[2].

Whenever the main FSM is in one of its states, the circuit may be in one of many different states. Each single state of the main FSM therefore corresponds to a set of states of the complete circuit C. The *set of states corresponding to* the main state $\hat{s}$ is

$$S_{\hat{s}} = \{s \in S \mid s(\hat{V}) = \hat{s}\}$$

In the same way, we can describe the transitions of the complete design C in terms of transitions of the main FSM. When the main FSM moves from one of its states, $\hat{s}_1$, to another state, $\hat{s}_2$, the circuit C correspondingly moves from some state in $S_{\hat{s}_1}$ to another state in $S_{\hat{s}_2}$. If we consider all possible transitions between such states corresponding to $\hat{s}_1$ and $\hat{s}_2$, we can define the design's transition relation corresponding to a transition of the main FSM. The *transition relation corresponding to* the main transition $(\hat{s}_1, \hat{s}_2)$ is

$$R_{\hat{s}_1 \to \hat{s}_2} = \{(s_1, s_2) \in R \mid s_1 \in S_{\hat{s}_1} \wedge s_2 \in S_{\hat{s}_2}\}$$

Given this definition of the transition relation $R_{\hat{s}_1 \to \hat{s}_2}$, we can define *image computation* corresponding to a transition $\hat{s}_1 \to \hat{s}_2$ of the main FSM. The *constrained img operation* of the set of states *From* $\subseteq S$ corresponding to the main transition $(\hat{s}_1, \hat{s}_2)$ is

$$\mathit{img}_{\hat{s}_1 \to \hat{s}_2}(\mathit{From}) = \{s_2 \in S \mid \exists s_1 \in \mathit{From} \wedge (s_1, s_2) \in R_{\hat{s}_1 \to \hat{s}_2}\}$$

The following two lemmas show how the state transition graph of a circuit C can be explored based on single transitions of the main finite state machine in the design.

[2]It is straightforward to devise a formal method for this purpose. In our work, we chose to rely on the commercial solution.

```
0 : procedure TBT_TRAVERSAL(R, S_0, Ŝ, R̂, Ŝ_0) {
1 :     queue = {ŝ | ŝ ∈ Ŝ_0};
2 :     for each ŝ ∈ Ŝ
3 :       if (ŝ ∈ Ŝ_0)   S_ŝ = S_0;   else   S_ŝ = ∅;
4 :     while (queue ≠ ∅) {
5 :       ŝ = dequeue(queue);
6 :       for each ŝ' where (ŝ, ŝ') ∈ R̂ {
7 :         S_ŝ' = S_ŝ' ∪ img_{ŝ→ŝ'}(S_ŝ);
8 :         if (S_ŝ' changes) enqueue(ŝ', queue); }}
9 :     return({S_ŝ}); }
```

Figure 1: TBT traversal algorithm

Lemma 1 *The set of states of the circuit can be decomposed into disjoint sets of states corresponding to main states:* $S = \bigcup_{\hat{s} \in \hat{S}} S_{\hat{s}}$

Lemma 2 *Let* $From_{\hat{s}_1}$ *be a set of states corresponding to a main state* $\hat{s}_1$*. The next states of* $From_{\hat{s}_1}$ *can be calculated as:* $img(From_{\hat{s}_1}) = \bigcup_{\hat{s}_2:(\hat{s}_1,\hat{s}_2) \in \hat{R}} img_{\hat{s}_1 \rightarrow \hat{s}_2}(From_{\hat{s}_1})$

For the reason of space, we omit the proofs of the lemmas. The reader can refer to [NSK05] for the proofs. Lemma 1 states that the states of the circuit can be computed implicitly and exactly as the union of states corresponding to main states. By Lemma 2 the next states of a set of states corresponding to a main state can be calculated using the *constrained img* operation for every outgoing transition of the corresponding main state. This is the basis of our *transition-by-transition* FSM traversal algorithm.

The algorithm in Figure 1 calculates reachable states of a circuit by decomposing the state space based on the main FSM. Procedure *TBT_TRAVERSAL* (*transition by transition*) takes the transition relation, the set of initial states and the main FSM of the circuit as inputs. It returns the sets of reachable states corresponding to main states.

At the beginning, the sets of states corresponding to the main states are empty except for the sets of states corresponding to the initial main states which contain the given initial states. A *queue* is used to keep track of the main states that need to be processed next. First, the queue contains the initial main states. A main state $\hat{s}$ is added to the queue if and only if its corresponding set of states changes. In this case, the sets of states corresponding to the next states of $\hat{s}$ have to be recalculated. For a main state $\hat{s}$ in the queue, the algorithm considers all its next states $\hat{s}'$. The set of states $S_{\hat{s}'}$ corresponding to main state $\hat{s}'$ is extended by the set of next states correspoding to the transition $\hat{s} \rightarrow \hat{s}'$ which is calculated by the *constrained img* operation. If this actually adds any new states to $S_{\hat{s}'}$ then $\hat{s}'$ is entered into the queue. When the queue is empty, the sets of states corresponding to the main states are unchanged and the algorithm terminates.

Theorem 1 *Procedure* TBT_TRAVERSAL *traversal calculates all reachable states of the circuit.*