社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2005－SLDM－122 (34)
2005／12／2

Structural Coverage of Traversed Transitions for Symbolic Model Checking

Xingwen Xu† Shinji Kimura† Kazunari Horikawa‡ and Takehiko Tsuchiya‡

† Graduate School of IPS, Waseda University 2-7 hibikino, Wakamatsu, 808-0135 Japan
‡ System LSI Design Division, Toshiba Corporation Sawasaki, 210-8520 Japan
E-mail: † xuxingwen@fuji.waseda.jp

Abstract Coverage estimation for model checking has become an important issue in practical formal verification. Transition traversal coverage focuses on the transition characteristics of CTL operators and calculates which transitions are traversed during the model checking process of properties. One limitation of the method is the lack of the correspondence between the circuit structure and transitions. One transition might be covered no matter which part of the circuit is checked (or not) related to the transition. This leads to the overestimation of the coverability of properties. In this paper, we enhance the transition traversal coverage by analyzing the structural coverage of each traversed transition. We consider which variables are checked explicitly or implicitly on the traversed transitions. Thus, we deduce which part of the circuit is checked by properties for each traversed transition. The importance is that we can analyze which part of the circuit has not been verified. The accuracy of the transition traversal coverage is enhanced by our technique. We show the effectiveness of the proposed method by experiments.

Keyword structural coverage, transition traversal, symbolic model checking

1. Introduction

Model checking [1], [2] proves whether a system satisfies a set of properties, which are manually specified in a property specification language such as Computation Tree Logic (CTL) [3]. The validation is exhaustive only for functions specified by the properties. A design error might not be detected by model checking if the erroneous behavior is not checked by the specified properties. Thus, to validate the whole functions of a design by model checking, a set of properties specifying all intended behaviors is desired. Specifying such a set of complete properties is a very challenging task because it is very hard to determine the completeness. As model checking becomes more and more widely used for industrial hardware verification, properties completeness analysis has become an important issue and has been addressed by a few recent articles [5]-[12].

Usually, the completeness problem is addressed by coverage methods as used in simulation-based verification methodology [13]. The properties are compared with the implementation according to certain metrics to reveal the behaviors of the implementation which have not been checked by the properties. Such unexplored behaviors indicate the incompleteness (coverage holes) of the properties. Additional efforts then should be taken to clarify the specification and enhance the properties. In other words, coverage methods try to ensure that there is no unknown behavior in the implementation since such behavior might contain design bugs. Assisting the model checking process with coverage estimation can achieve higher degree of confidence in the verification results.

Several coverage methods have been proposed for model checking. The simulation preorder is used in [5] to compare the implementation model and the tableau model translated from properties. High-level fault models such as stack-at fault are used to analyze the completeness of properties of the coverage methods in [9], [10]. The state coverage method evaluates the coverage of states in the circuit finite state machine (FSM). It checks whether the value of the selected observed signal on a state is critical to the satisfaction of properties. Transition-based coverage metrics enable more comprehensive and precise coverage analysis. Chockler et.al. have presented coverage metrics based on omitting or mutating transitions (or paths) of a FSM [11]. However, the practicality of their techniques has not been fully established.

We have proposed a transition traversal coverage method for a subset of CTL [14]. It focuses on the transition characteristics of properties. A property is interpreted as a set of computation paths which must be traversed to prove the property. Transitions on such paths are considered as covered. The set of covered transitions is obtained by traversing transitions of the circuit FSM for each property according to the semantics of CTL. One weak point of the transition traversal coverage is the overestimation due to the lack of correspondence between the circuit structure and transitions. For example, a local

property describing functions of a sub-circuit might be checked by traversing all transitions of the whole circuit. This will overestimate the coverability of the specification for the whole circuit. Further more, even for global properties, not all the circuit is verified for each traversed transition. Usually, along a computation path of a property, different transitions are related with different parts of the circuit when considering the functions verified by the property.

In this paper, we enhance the preciseness of the transition traversal coverage by evaluating the structural coverage for each traversed transitions. A property is viewed as a set of computation paths during transition traversal. We consider which variables are checked on each transition in the computation paths. We not only consider the explicitly asserted variables by properties, but also consider the implicitly checked variables. When a register variable is checked on a transition, its fan-in registers are implicitly checked on the previous transition in the computation path. Thus, we deduce which part of the circuit is checked by properties for each traversed transition. The importance is that we can analyze which part of the circuit has not been verified for each state-transition.

2. Preliminaries

A sequential circuit is usually modeled as a FSM to represent its functionality. A Finite State Machine (Moore type) is a 6-tuple $M=<I, O, S, \delta, \lambda, S_0>$, where

- S is the set of states,
- I is the input space,
- O is the output space,
- δ: $S \times I \rightarrow S$ is the state transition function,
- λ: $S \rightarrow O$ is the output function,
- S_0 is the set of initial states.

When we talk about transition coverage, we refer to the transitions of the FSM. On the other hand, model checking is usually performed on the Kripke structure. Let AP be a set of atomic propositions. A Kripke structure over AP is a four tuple $K=<S, S_0, R, L>$, where

- S is a finite set of states,
- $S_0 \subseteq S$ is the set of initial states,
- $R \subseteq S \times S$ is a complete transition relation,
- L: $S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions true in that state.

The circuit FSM can be translated to a kripke structure so that input signals are labeled on states. For a FSM $M=<I, O, S, \delta, \lambda, S_0>$, the corresponding Kripke structure can be derived as $K=<S\times I, S_0\times I, R, L>$, where for any $s, s' \in S$ and any $i, i' \in I$:

- '$(<s,i>, <s',i'>) \in R$ *iff* $\delta(s,i)=s'$',
- $L(<s,i>)=i \cup s \cup \lambda(s)$.

From the translation, it is clear that each transition in the circuit FSM corresponds to a state in the Kripke structure and vice versa. Transition traversal coverage computation is based on such translation and interprets each state in the Kripke structure as a transition of the circuit FSM.

The transition traversal coverage is defined for the following subset of CTL:

- If b is a propositional formula, b is within the subset;
- If f and g are within the subset, so are $b \rightarrow g$, $f \wedge g$, AXf, AGf, $AfUg$, $AfRg$.

The set of covered (traversed) transitions for a CTL property within this subset is shown in Table 1 where

- ϕ represents the empty set,
- $Fwd(S)$ represents the set of forward image of S,
- $Sat(f)$ represents the set of states satisfying f,
- $Rch(S)$ represents all reachable states from S.

Table 1 The set of traversed Transitions.

$C(S, f)$	Set of Covered Transitions
$C(\phi, f)$	ϕ
$C(S, b)$	ϕ
$C(S, g_1 \wedge g_2)$	$C(S, g_1) \cup C(S, g_2)$
$C(S, b \rightarrow g)$	$C(S \cap Sat(b), g)$
$C(S, AXg)$	$S \cup C(Fwd(S), g)$
$C(S, AGg)$	$C(Rch(S), g)$
$C(S, A(g_1 U g_2))$	$C(S \cap Sat(g_2), g_2) \cup$ $C(S \cap Sat(\neg g_2), g_1 \wedge AX(A(g_1 U g_2)))$
$C(S, A(g_1 R g_2))$	$C(S \cap Sat(g_1), g_1 \wedge g_2) \cup$ $C(S \cap Sat(\neg g_1), g_2 \wedge AX(A(g_1 R g_2)))$

3. Motivative Example

In this section, we present a simple example to explain the limitation of the transition traversal coverage and show the concept of structural coverage for state-transitions.

Our example is a synchronous modulo-8 counter as shown in Fig 1. Consider a property that whenever R_1 is "1", it will be "0" after two clock cycles:

$AG(R_1=1 \rightarrow AX\,2\,(R_1=0))$.

According to the transition traversal coverage, this property will traverse the four transition sequences as shown in Fig 2. Six transitions in these sequences are covered then. The problem is for these traversed transitions, which part of the circuit is really checked by this property? In order words, is there any part of the

circuit not checked on the traversed transitions.

It is obvious that this property dose not verify the

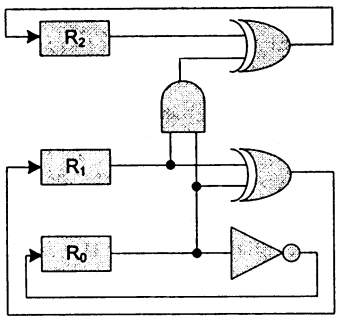

Figure 1 A Synchronous modulo-8 counter.

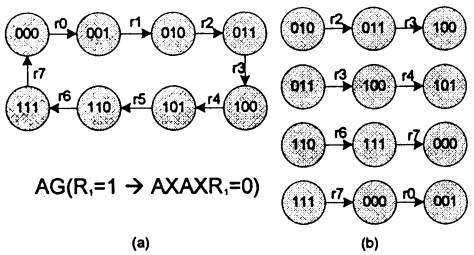

Figure 2 (a) is the FSM of the counter and a satisfied property; (b) is the traversed transition sequences by this property. Six transitions {r0,r2,r3,r4,r6,r7} are covered by this property.

function of the sub-circuit for R_2. But every transition involves the function of R_2. For example, the traversed transition $r7$ shows when R_2=1, R_1=1, R_0=1, R_2 will be "0" in the next state. Although $r7$ is traversed, we can not say such a transition is thoroughly verified since the function of R_2 is not checked at all.

Then what is checked for a traversed transition? According to the property, the value of R_1 should be "0" on the reached state of transition of $r7$. Then the function of R_1 on $r7$ is checked. Furthermore, since the values of R_1 and R_0 decide the value of R_1 in the next clock cycle, when R_1 is checked on $r7$, R_1 and R_0 is also checked on the previous traversed transition $r6$.

From this example, we observe that due to lack of observability, the transition traversal coverage dose not answer why a transition is traversed. The relation between the transition traversing and the verified functions is very loose. This might lead to the overestimation of properties' completeness. A transition will be traversed if any function of this transition is checked. But since a transition is about the functions of the whole circuit, many functions (for sub-circuits) of a traversed transition might have not been checked. In this paper, we propose a novel structural coverage of traversed transitions. Our objective is to analyze whether the function of each sub-circuit is well verified.

4. Structural Coverage of Traversed Transitions

In this section, we present our techniques for analyzing the structural coverage of traversed transitions. We first explain the circuit structure representation for our coverage. Then we present our definition for the checked register on a traversed transition. The intuition of the structural coverage metric is explained afterwards.

4.1. Structure Representation

A synchronous circuit can be represented by its transition relation. Take the counter of Fig. 1 as an example. $V=\{R_2, R_1, R_0\}$ is the set of register variables for this circuit. We use them to represent current-state variables. Let $V'=\{R_2', R_1', R_0'\}$ represents the set of next-state variables for these registers. Then the function of each register is given by:

$R_2' = (R_1 * R_0) \otimes R_2$

$R_1' = R_1 \otimes R_0$

$R_0' = \sim R_0$

They can be used to define the relations:

$\mathcal{R}_2 = (R_0' \Leftrightarrow \sim R_0)$

$\mathcal{R}_1 = (R_1' \Leftrightarrow R_1 \otimes R_0)$

$\mathcal{R}_3 = (R_2' \Leftrightarrow (R_1 \wedge R_0) \otimes R_2)$

The function of the counter is represented by the transition relation:

$\mathcal{R}(V,V') = \mathcal{R}_2 \wedge \mathcal{R}_1 \wedge \mathcal{R}_3$

In the general case of a synchronous circuit with n registers, let $V=\{R_0, \ldots R_{n-1}\}$ and $V'=\{R_0', \ldots, R_{n-1}'\}$. For each register R_i' there is a Boolean function f_i such that

$R_i'=f_i(V)$.

These functions are used to define the relations

$\mathcal{R}_i(V,V')=(R_i' \Leftrightarrow f_i(V))$.

The whole circuit is then represented by the conjunction of these formulas

$\mathcal{R}(V,V') = \mathcal{R}_2 \wedge \cdots \wedge \mathcal{R}_{n-1}$

From the circuit representation, it is clear that the function of each register $f_i(V)$ is the basic element of the circuit transition relation. It describes the function of the fan-in logic of the register. Since every transition is a set of evaluation of each register function, in order to analyze the specific verification intent on a traversed transition, we can analyze which register functions on the transition is checked. In other words, we use registers as

elements of circuit structure for analyzing the coverage on traversed transitions.

For each register R_i, and its Boolean function $R_i'=f_i(V)$.

We call a register R_j is a fan-in register of R_i iff:

$\exists V\ (\partial f_i(V)/\partial R_j)=1$

where $\partial f/\partial R$ is the logical differential.

4.2. Registers checked on traversed transitions

A property usually specifies how particular states should be reached through transitions and the correctness conditions for certain circuit signals on the reached states. To simplify discussion, we assume the property only involves register variables. Now we talk about two kinds of checks for registers on traversed transitions by properties. We use the 3-bit counter example of Section 3 and the property $AG(R_1=1 \rightarrow AXAX\ (R_1=0))$ for our explanation.

Explicitly Checking

A register is explicitly checked on a transition if the property specifies the correctness condition about the register on the state reached by that transition.

In the counter example, look at the transition traversal paths of the property. For the first-step traversed transitions ($r2,r3,r6,r7$) by the first AX operator, the property dose not specifies any signal's correctness conditions on the reached states. But for the second AX operator, the property specifies that register R_1 should be "0" on the reached states. So on transitions traversed at this step ($r3,r4,r7,r0$), register R_1 is explicitly checked. The meaning of kind of checking is straightforward. It is just the verification intent of the property.

To compute the set of checked registers and their corresponding transitions, we just need to judge whether the property specifies a correctness condition on each traversed transition. Since the correctness condition is specified by propositional formulas, during traversing for a CTL operator, we just need to check if this CTL operator leads any propositional formula and find out on which transitions this propositional formula is satisfied. Registers involved in the propositional formula are explicitly checked on such transitions.

Implicitly Checking

When a register is checked (explicitly or implicitly), its fan-in registers are implicitly checked on the previous transitions in the traversed paths.

Look at the counter example again, take a traversed path $\{r6 \rightarrow r7\}$, since register R_1 is checked (explicitly) on $r7$, then its fan-in registers $\{R_0, R_1\}$ are implicitly checked on the previous transition $r6$. Registers on such transitions are also considered as checked because their values determine the values of explicitly checked registers through the transition relation.

The computation is more complex than explicitly checking. We start backward traversing from transitions on which a register is explicitly checked. Then we identify the transitions which are in the traversal paths leading to the explicit checking. The fan-in registers of the explicitly check register are then implicitly checked on such transitions. These transitions and registers are then taken as start point for the next-step backward traversing and so no.

4.3. Coverage Metric

The structural coverage of a traversed transition is defined as which registers are checked on that transition. The intuitive meaning of the structural coverage metric is explained as follows.

Consider the function of a register $R'=f(V)$ of the circuit C. For a given transition $r(Vr,Vr')$, we introduce a perturbed function for the register R:

$$fp(V)=\begin{cases} f(V) & \text{if } V\neq Vr \\ \sim f(V) & \text{if } V=Vr \end{cases}$$

By replacing f with fp, a new circuit C' can be constructed. Note that distinct with state perturbation where only the labels on states are changed, we change the circuit transition relation by redirecting r to another state where only the value of R is different, as shown in Fig. 3. The structural coverage indicates that if the register R is covered on the transition r by a property, then the property might get failed on the new circuit C'. More precisely, if the register is not covered on the transition by any property, then all property will pass on the perturbed circuit C'.

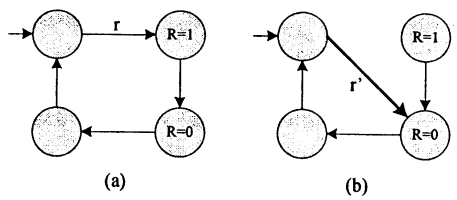

Figure 3 The transition perturbation. (a) is the original circuit FSM; (b) is the perturbed one on transition r for register R.

Proof: Consider a traversed transition sequence ($r \rightarrow rj \rightarrow ... \rightarrow ri \rightarrow rc$) and the property explicitly checks the value of register Rc on state Sc. Suppose register R is not covered on transition r. In order to prove the property will pass on perturbed circuit, consider the same length of transition sequence from r' ($r' \rightarrow rj' \rightarrow ... \rightarrow ri' \rightarrow rc'$), we should prove the value of Rc on state Sc' is same with that on Sc. Assume Rc has different value on Sc'. Then there is at least one fan-in register Ri of Rc which has different values on the previous states Si and Si'. Ri is covered on transition ri because it is implicitly checked. For the same reason, there is at least on register Rj which is implicitly checked on transition rj and has different values on states Sj and Sj'. Because Rj has different values on Sj and Sj' and only register R has different values on the previous states Sr and Sr', R must be a fan-in register of Rj. Because Rj is implicitly checked on rj, R is also implicitly checked on the previous transition r. Then R is covered on transition r, which conflicts with the given condition that R is not covered on r. So Rc can not have different value on Sc and Sc'.

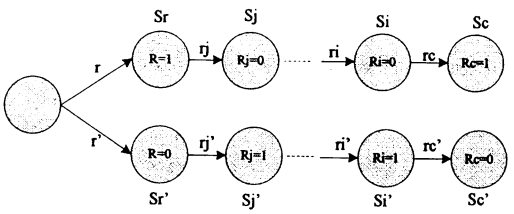

Figure 4 Transition r is redirected to a state where only register R has different value on Sr and Sr'. Consider a transition path from r' with same length as the traversed path, if the explicitly checked register Rc has different value on state Sc', register R will be covered on transition r.

Now let's consider several coverage conditions. First, we can analyze which registers are covered for a transition. Attention should be paid to those transitions on which only one or a few registers are covered. The low register coverage indicates many registers are don't-care on this transition. This could be true don't-care conditions according to design specification. But this also might be missed functions in the properties. Second, we can analyze on which transitions a register is covered. It is a serious coverage hole if a register is never covered on any transition. This means changing the sub-circuit of this register will not falsify any property. So the function of this register is totally irrelative to the satisfaction of properties and the function of such a register is not verified at all. If a register is covered only on a few transitions, we should also analyze don't-care or incompleteness.

5. Experiments

Fig. 5 is a simple three-stage float-point multiplier and we have a property that when multiplication begins and the multiplicands have the same sign values, the sign value of the production will be "0" after three clocks. It can be specified by the following CTL property:

$AG((begin=1 * X[7]==Y[7]) \rightarrow AX:3\ (Z[7]=0))$.

Transition traversal coverage will report that 50% transitions are covered. However, we have no idea about which part of the circuit is related with the covered transitions. As a matter of fact, this property only checks the computation of sign-bit and only part of the circuit is responsible for it. Other part of the circuit like the computation of significand and exponent of the production is totally irrelative to this property. However the 50% transition coverage dose not indicate such structural relation.

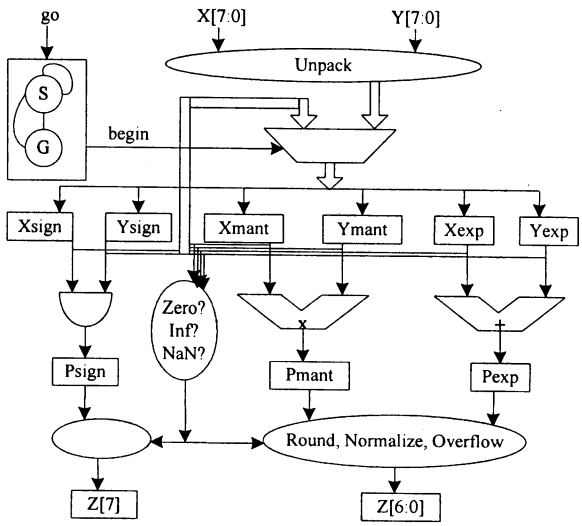

Figure 5 A three-stage float-point multiplier

Now let's look at the structural coverage on traversed

transitions. We check that on how many transition each register is covered. Among 43 registers, the transition coverage for each register is shown as follows: *Z[7]* 50%; *Pmant*, *Pexp*, *Z[6:0]* 0%; *Xsign*, *Ysign* 0.03%; other registers 43.6%. The results clearly indicate that only the computation for sign-bit is covered, and the function of *Xsign* and *Ysign* is not well verified.

6. Conclusions

Properties completeness analysis is an important issue for the deployment of model checking in real designs. Based on the transition traversal of CTL properties, we propose a structural coverage method to analyze which part of the circuit is actually related with the traversed transitions. As a combined method of FSM-based and circuit-based coverage metrics, our techniques are able to analyze whether some part of the circuit is not verified even when all transitions are traversed.

References

[1] E.M. Clarke, O. Grumberg, and D. Peled, Model Checking, MIT Press, 1999.

[2] J.R. Burch, E.M. Clarke, D.E. Long, K.L. McMillan, and D.L. Dill, "Symbolic model checking for sequential circuit verification," IEEE Trans. CAD, vol.13, no.4, pp.401-424, 1994.

[3] M. Ben-Ari, A. Pnueli, Z. Manna, "The Temporal Logic of Branching Time," Acta Informatica, Vol.20, pp.207-226, 1983.

[4] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Computers, vol.35, no.8, pp.677-691, 1986.

[5] S. Katz, O. Grumberg, and D. Geist, "Have I written enough properties? - A method of comparison between specification and implementation," Proc. ACM Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARM'99), pp.280-297, 1999.

[6] Y. Hoskote, T. Kam, Pei-Hsin Ho, and Xudong Zhao, "Coverage estimation for symbolic model checking," Proc.DAC'99, pp.300-305, 1999.

[7] N. Jayakumar, M. Purandare, and F. Somenzi, "Do's and don'ts of CTL state coverage estimation," Proc. DAC'03, pp.292-295, 2003.

[8] H. Chockler, O. Kupferman, R. Kurshan and M.Y. Vardi, "A Practical Approach to Coverage in Model Checking," Proc. International Conference on Computer Aided Verification (CAV'01), pp.66-78, 2001.

[9] F. Fummi, G. Pravadelli, A. Fedeli, U. Rossi, and F. Toto, "On the use of a high-level fault model to check properties incompleteness," Proc. ACM/IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE'03), pp.145-152, 2003.

[10] S. Das, et al., "Formal Methods for Analyzing the Completeness of an Assertion Suite against a High-Level Fault model," Proc. International Conference on VLSI Design,pp.201-206, 2005.

[11] H. Chockler, O. Kupferman, and M.Y. Vardi, "Coverage Metrics for Formal Verification," Proc. ACM Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARM'03), pp.111-125, 2003.

[12] S. Das, et al., "Formal Verification Coverage: Computing the Coverage Gap between Temoral Specifications". Proc. ICCAD'04, pp.198-203, 2004.

[13] S. Tasiran, and K. Keutzer, ¥Coverage metrics for functional validation of hardware designs," IEEE J. Design & Test of Computers, vol.18 no.4, pp.36-45, 2001.

[14] X. Xu, S. Kimura, K. Horikawa, and T. Tsuchiya. "Transition Traversal Coverage Estimation for Symbolic Model Checking". In Proceedings of the 3rd ACM/IEEE international Conference on Formal Methods and Models for Co-Design (MEMOCODE), page 259-260, 2005.