논문 2005-42SD-10-7

불완전명세 상태천이그래프상에서 미정의상태를 이용한 동기순차회로의 테스트용이화 합성

(Synthesis for Testability of Synchronous Sequential Circuits Using Undefined States on Incompletely-Specified State Transition Graph)

최호용*, 김수현**

(Ho-Yong Choi and Soo-Hyun Kim)

요 약

본 논문에서는 불완전명세(incompletely-specified)를 가진 상태전이그래프(state transition graph: STG)상에서 리던던트 고장(redundant faults)수를 줄여 테스트를 용이하게 하기 위한 새로운 동기 순차회로의 합성방법을 제안한다. 이 STG 합성법에는 1) 구별전이(distinguishable transition)을 이용하여 무정의상태(undefined states)와 불완전명세된 입력전이를 추가하고, 2) 가능한 한 강연결(strongly-connected)이 되도록 하는 방법을 사용한다. 제안된 방법을 이용하여 MCNC 벤치마크 회로에 대해 실험한 결과, 대부분의 회로에 대해 무해 고장의 수가 현격히 줄어들어 높은 고장검출을 얻었다.

Abstract

In this paper, a new synthesis method for testability of synchronous sequential circuits is suggested on *an incompletely-specified state transition graph (STG)* by reducing the number of *redundant faults*. In the suggested synthesis method, 1) a given STG is modified by adding *undefined states* and unspecified input transitions using *distinguishable transition*, 2) the STG is modified to be strongly-connected as much as possible. Experimental results with MCNC benchmark show that the number of redundant faults of gate-level circuits synthesized by our modified STGs are reduced, and much higher fault coverage is obtained.

Keywords : synthesis for testability, undefined states, redundant faults, distinguishable transition

I. Introduction

A fault in a sequential circuit is defined to be redundant if the fault-free and faulty machines are equivalent. In general, it may require long test generation time to identify such redundant faults, especially these faults are caused by *illegal states*, where illegal states are defined as states which are not part of any strongly connected component of the circuits[1-4]. Because a sequential test generator may waste a lot of time trying to justify illegal states in addition to exhaustive searching for a test sequence without finding any. Hence, for a circuit that has lots of redundant faults, the number of redundant faults needs to be decreased for both high fault coverage and short generation time.

There are two research directions to solve the problems related to redundant faults[1,4-9]. One direction is to identify or remove redundant faults in synthesized gate-level circuits[1,4-7], and to be used for test generation or logic optimization. These

* 평생회원, 충북대학교 전기전자컴퓨터공학부
(School of Electrical and Computer Eng., Chungbuk National University)
** 정회원, 삼성SDI
(Samsung SDI Co.)

※ This work was supported by the research grant of the Chungbuk National University in 2004 and by the tools of IDEC.
접수일자: 2005년7월14일, 수정완료일: 2005년10월4일

approaches can easily detect redundant faults resulting to shortened test generation time. However, they have no effect on fault coverage itself, and have limitation to reduce test generation time.

The other direction is the synthesis for testability (SFT) approach, which has been introduced to solve testing problems in connection with redundant faults[9]. Main idea of this approach is to prevent occurrence of redundant faults by modifying the specification of circuits, so called state transition graphs (STGs) or state transition tables (STTs). Pomeranz *et al.* proposed an SFT method on incompletely-specified STG[9], in which an STG is synthesized to be strongly connected using unspecified input transitions on the STG. However, this method does not consider undefined states which have possibility to be synthesized as illegal states.

In this paper, we propose an synthesis-for-testability approach to obtain high fault coverage by reducing the number of redundant faults on an incompletely-specified STG. In the proposed algorithm, an incompletely-specified STG is modified to be strongly-connected as much as possible using distinguishable transition to add undefined states.

This paper is organized as follows. Section II presents preliminaries necessary for better understanding of this paper. In Section III, details of our synthesis approach are explained. Section IV shows the experimental results with MCNC benchmark and Section V concludes the paper.

II. Preliminaries

A finite state machine (FSM) M is defined such as 5-tuples :

$M=(I,S,O,\delta,\gamma)$, where

$-\ I$ is a set of input vectors,

$-\ S$ is a set of states,

$-\ O$ is a set of output vectors,

$-\ \delta: I\times S\rightarrow S$ is the next state function, and

$-\ \gamma: I\times S\rightarrow O$ is the output function.

If the state transitions and outputs for all input combinations are specified for each state in an FSM, then the machine is said to be completely-specified. Otherwise, the machine is incompletely-specified[10].

Each directed arc in a state transition graph (STG) is called a transition denoted by

$T_{i,j}[I/O]=(I_{i-j},S_i,S_j,O_{i-j})$, where

$-\ I_{i-j}\in I$ is a primary input from state S_i to state S_j,

$-\ S_i\in S$ is a present state,

$-\ S_j=\delta(I_{i-j},S_i)$ is a next state from state S_i for I_{i-j}, and

$-\ O_{i-j}=\gamma(I_{i-j},S_i)$ is a primary output from state S_i for I_{i-j}.

In a graphical representation, a state is *a defined state* if it is described in a specification given by an FSM. A state is *an undefined state* if not described in the specification. If for every pair of states S_i, S_j of machine M there exists an input sequence which takes M from S_i to S_j, then M is said to be strongly connected. Further, let's denote an SCC (Strongly Connected Component) is a set of states in which any state is reachable from any other states in that set. And a state is *legal* if it is a part of an SCC. Otherwise, a state is to be *illegal.*

Definition 1: For an FSM $M=(I,S,O,\delta,\gamma)$ and a transition $T_{i-j}[I/O]$ is a distinguishable trnsition $dT_{i-j}[I/O]$ with a given input I_{i-j}, if a primary output from a present states S_i is different from outputs from other present states, i.e., $O_{i-j}(=\gamma(I_{i-j},S_i))\neq O_{i-j}'(=\gamma(I_{i-j},S_i'))$ for any $S_i\neq S_i'$.

Such a defined distinguishable transition $dT_{i,j}[I/O]$ is used for enhancing observability of an STG in adding a new transition.

Ⅲ. Synthesis for Testability on an Incompletely Specified Using Undefined States

In this section, we introduce a new approach for synthesis for testability using undefined states and transitions on incompletely-specified STG in order to obtain high fault coverage by reducing the number of redundant faults.

In an incompletely-specified STG, there may exist undefined states and defined states which have unspecified transitions, denoted by incompletely-specified (IS) states. In the proposed SFT method, an incompletely-specified STG is modified to be strongly connected as much as possible by adding undefined states, and unspecified transitions of defined states are newly assigned to connect from defined states to undefined states.

Figure 1 shows how to make a strongly connected STG by inserting undefined states forcedly for three types of STGs we consider in this paper. A strongly connected STG (type 1: original in Fig. 1(a)) is modified to be strongly connected using undefined states (modified in Fig. 1(a)). To add an undefined state as a legal state, we used an unspecified transition of a defined state. That is, an unspecified transition of a defined state is newly specified to an undefined state. Also, an unspecified transition of the undefined state is specified to a defined state. So, the modified STG also becomes strongly connected.

If the given STG is not strongly connected, there are two types. Type 2 is an STG composed of an SCC and a reachable (from the SCC) illegal state (original in Fig. 1(b)), and type 3 is composed of an SCC and an unreachable (from the SCC) illegal state (original in Fig. 1(c)). For both types, illegal IS states can be included in an SCC by adding transitions from and to legal states of the given STG via undefined states. The modified STG for type 2 is shown at modified in Fig. 1(b). An unspecified transition of an illegal state is specified to an undefined state, and an unspecified transition of an undefined state is specified to a defined legal state. Hence, the illegal state and the added undefined state are included in an SCC resulting modified STG to be strongly connected. For type 3, the modified STG is shown at modified in Fig. 1(c). An unspecified transition of an legal state is specified to an undefined state, and an unspecified transition of undefined state is specified to an illegal state, so the modified STG becomes strongly connected.

Algorithm 1 shows how to perform the SFT on an incompletely-specified STG. At first, an undefined state S_{un} is selected to be added on the STG. Next, a state S_{in} that will be connected to S_{un} is selected. To make the given STG a strongly connected graph, we consider following conditions to select S_{in}. If the given STG is a strongly connected graph, any defined

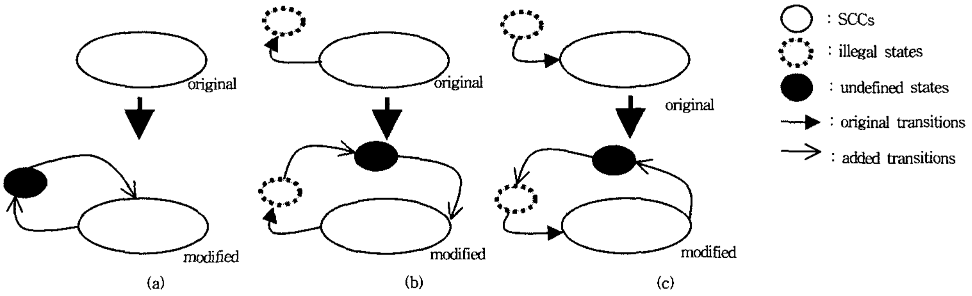

그림 1 3가지 STG 형태에 대한 수정된 강연결 STG: (a) 형태 1: 강연결 STG, (b) 형태 2: 도달가능한 illegal 상태를 가진 STG, (c) 형태 3: 도달불가능한 illegal 상태를 가진 STG

Fig. 1. Modified strongly connected STGs for three types of STGs: (a) type 1: a strongly connected STG, (b) type 2: an STG with a reachable illegal state, and (c) type 3: an STG with an unreachable illegal state.

IS state can be selected as S_{in} (Fig. 1(a)). If not, it checks whether there exists a defined IS state, which is illegal and reachable from defined states. If it exists, it is selected as an S_{in} (Fig. 1(b)). Otherwise, any legal IS state can be selected as an S_{in} (Fig. 1(c)). Then, the distinguishable transition from S_{in} to S_{un}, $dT_{in,un}$[I/O], is added to the STG. Again, a state S_{out}, which will be connected from S_{un}, is selelcted and distinguishable transition from S_{un} to S_{out}, $dT_{un,out}$[I/O], is added to the STG. This algorithm is finished when all undefined states are added. Note here that strongly-connectivity of an STG may be breakable even if the original STG is strongly connected. Generally, logic synthesis tools assign undefined states for achieving an optimized circuit. Hence undefined states are likely to be synthesized as illegal states during logic synthesis, more specifically in logic optimization and/or hazard optimization. In

[Algorithm]

```
input : given STG
output : the modified STG to minimize the redundant faults using undefined states
  main (STG) {
      $U$ := the set of undefined states ;
        do {
          $S_{un}$ := an arbitrary selected state in $U$ ;
            $U$ := $U$ - $S_{un}$ ;
            if ((chk_SCG(STG) == FALSE) & (( $S_1$ = chk_illegal_input(STG)) != NULL))
                $S_{in}$; := $S_1$ ;
            else  $S_{in}$ := an arbitrarily defined IS state ;
            new_STG := add_tr(STG, $S_{in}$, $S_{un}$);
            if ((chk_SCG(STG) == FALSE) & (( $S_2$ = chk_illegal_output(STG) != NULL))
                $S_{out}$ := $S_2$ ;
            else $S_{out}$ := an arbitrarily defined state ;
            new_STG := add_tr(new_STG, $S_{un}$, $S_{out}$) ;
            return new_STG ;
        } while (( $U$ != $\emptyset$ ) }

  procedure chk_SCG(STG) {
     if ( STG is Strongly_Connected_Graph )
            return TRUE ;
     else return FALSE ; }

  procedure chk_illegal_input(STG) {
       $S_I$ := the set of defined states which have unspecified inputs ;
       if ( there exist a state $S_{il} \in S_I$ which is illegal and reachable from defined states )
            return $S_{il}$ ;
       else return NULL ; }

  procedure add_tr(STG, $S_{src}$, $S_{dst}$) {
       $In_{src-dst}$ := an arbitrarily unspecified input of $S_{src}$ to $S_{dst}$ ;
       find $dT_{src,dst}$[In/O] = ( $I_{nsrc\text{-}dst}$, $S_{src}$, $S_{dst}$, $O_{src-dst}$) ;
       add $dT_{src,dst}$[In/O] to the STG and return STG ; }

  procedure chk_illegal_output(STG) {
       if ( there exist a state $S_{il}$ which is unreachable from defined states )
            return $S_{il}$ ;
       else return NULL ; }
```

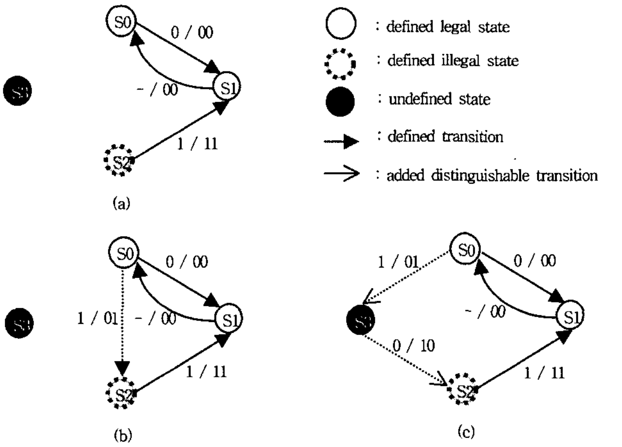

그림 2. Ref.[9]와 우리 방법과의 차이: (a) 주어진 STG, (b) Ref.[9]에 의해 수정된 STG, (c) 우리 방법에 의해 수정된 STG

Fig. 2. Difference between Ref. [9] and our method : (a) a given STG, (b) the modified STG by Ref.[9], and (c) the modified STG by our method.

our method, undefined states are included in an SCC by enforcing a synthesized STG to be strongly connected, differently from the reference [9].

Figure 2 shows a simple example to represent the difference between the reference [9] and our method. The original STG is given in Fig. 2(a). This STG is composed of three defined states (S0, S1, S2) with one illegal state (S2), and one undefined state (S3). Figure 2(b) shows the modified STG by the method in [9]. As mentioned earlier, the method in [9] does not consider undefined states; hence unspecified transition of state S0 is specified to state S2. As a result, all defined states are included in an SCC. However, we cannot guarantee whether the undefined state S3 will be included in an SCC or not after the synthesis procedure. Figure 2(c) shows the modified STG by our method. In our method, unspecified transition of state S0 is specified to the undefined state S3, and unspecified transition of state S3 is newly specified to the state S2. Hence, all states including defined states and undefined states are included in the SCC resulting in the synthesized STG to be strongly connected.

Figure 3 shows a simple example of synthesis for an incompletely-specified STG. This STG is composed of three defined states (S0, S1, S2) with one IS state S2, and one undefined state (S3). At first, an undefined state S3 is selected as an Sun. Next, among defined states, S2 is selceted as an Sin because S2 is the only IS state. Then a new distinguishable transition, dT2,3[0/10] = (0, S2, S3, 10), is added to the STG. Because the given STG is a strongly connected graph, Sout is arbitrarily selected among defined states. In this example, S2 is selected as an Sout. Again, a new distinguishable transition, dT3,2[0/11] = (0, S3. S2, 11), is added to the STG. Because there are no more undefined states, the STG modification for testability procedure is finished.

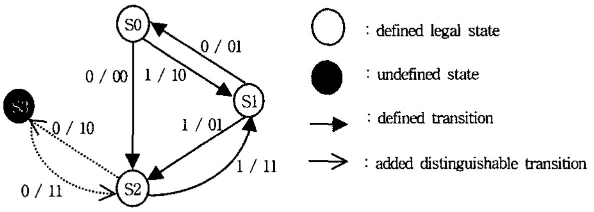

그림 3. 불완전명세 STG의 수정

Fig. 3. Modification of an incompletely-specified STG.

Ⅳ. Experimental Results

Experiments to verify the proposed SFT method are conducted on Sun Sparc Ultra-I with 128 megabytes RAM. Overall procedure for our experiments is as follows.

[Overall procedure]

input : For a given circuit in MCNC benchmarks

1) To obtain a modified STG by applying our proposed procedure to the given STG.
2) To obtain a gate-level circuit by performing logic synthesis to the modified STG using SIS[11].
3) To perform test generation using HITEC[12] to the synthesized gate-level circuit.

output : Fault coverage obtained by 3)

The statistics of the MCNC benchmarks are shown in Table 1 in terms of the number of inputs (Input), the number of outputs (Output), the number of defined states (Defined states), the number of undefined states (Undefined states), and the type of circuits (Classification).

Table 2 shows experimental results for incompletely-specified STGs. The results for original circuits are shown in column 2 through 6, with respect to the number of total single stuck-at faults (flts), the number of redundant faults (red), fault coverage (f.c. (= detected faults / total faults)), test generation time (time), and the number of synthesized gates (area). The results of the SFT method in [9] are given in the next three columns. The results of the proposed SFT method are given in the next six columns. Also, comparisons of fault coverage between original and proposed methods are given (f.c. inc.). For the original circuits or the modified circuits by reference [9], we have obtained non-effective results exhibiting very low fault coverage and long test generation time. However, for the modified circuits by Proposed method, the number of redundant faults has been reduced resulting much higher fault coverage and reduced test generation time. Compared to original results, fault coverage of Proposed method has been increased 34.3% in terms of average. Hence,

표 1. 예제 회로
Table 1. Example statistics.

circuit	input	output	defined states	undefined states
beecount	3	4	7	1
ex1	9	19	20	12
ex2	2	2	19	13
ex3	2	2	10	6
ex4	6	9	14	2
ex5	2	2	9	7
ex7	2	2	10	6
lion9	2	1	9	7
pma	8	8	24	8
tma	8	8	24	8
train11	2	1	11	5

표 2. 불완전명세 STG의 실험결과
Table 2. Experimental results of incompletely-specified STG.

Circuit	Original					Method of [9]			Proposed Method					
	flts	red	f.c. (%)	time (sec)	area	flts	red	f.c. (%)	flts	red	f.c. (%)	f.c.inc	time (sec)	area
beecount	101	0	100	0.16	30	115	2	98	101	0	100	0	0.15	36
ex1	465	437	6	18.58	151	533	333	38	620	136	78	72	14.75	220
ex2	262	256	2	0.08	92	279	279	0	334	334	0	-2	0.05	113
ex3	141	115	18	39.47	46	131	36	73	167	33	80	62	2.45	57
ex4	161	155	4	0.27	49	173	150	13	189	11	94	90	2.23	62
ex5	119	113	5	0.08	39	147	107	27	192	38	80	75	0.05	52
ex7	134	128	5	0.08	47	171	161	5	172	157	9	4	9.67	54
lion9	43	5	88	0.13	10	68	5	93	74	7	91	3	1.28	19
pma	397	397	0	0.06	134	419	414	1	419	414	1	1	2.02	141
tma	255	255	0	0.05	82	352	352	0	313	104	67	67	0.06	108
train11	70	6	91	0.48	21	70	2	97	92	3	97	6	0.87	30

it is noteworthy that the proposed SFT method is effective in terms of fault coverage. Also, for most circuits, test generation time has been reduced. That is, the decrease of redundant faults makes lower complexity of test generation. However, for some circuits, results of Proposed method are equivalent or worse than those of original or Ref. [9]. The reason can be rationalized as follows; faults detected easily with the other methods are hardly detected with our modified STGs because of forced assignment of transitions and undefined states. Trade-off of proposed SFT method is the slight increase of area. General synthesis tools assign undefined states for achieving optimized circuits such as minimum circuit size or delay. In the proposed SFT method, those assigning is utilized for testability.

V. Conclusion

We proposed a method for synthesis for testability using undefined states on an incompletely-specified state transition graph to obtain high fault coverage by reducing the number of redundant faults. In our approach, distinguishable transitions are added between undefined states and defined states to make a given STG into strongly-connected graph. We obtained high fault coverage and decreased test generation time in cost of slightly increased circuit area. As a future work, we will consider SFT methods which decrease both redundant faults and area overhead.

References

[1] S. Devadas, H.-K. Tony Ma, A. R. Newton, and A. Sangiovanni-Vincentelli, "Irredundant sequential machines via optimal logic synthesis," *IEEE Trans. on CAD,* vol. 9, pp. 8-18, Jan. 1990.

[2] I. Pomeranz and S. M. Reddy, "Classification of faults in synchronous sequential circuits," *IEEE Trans. on Computer,* vol. 42, no. 9, Sep. 1993.

[3] S. M. Reddy, I. Pomeranz, X. Lin, and N. Basturkan, "New procedures for identifying undetectable and redundant faults in synchronous sequential circuits," *Proc. of VLSI Test Symposium,* pp. 275-281, 1999.

[4] D. E. Long, M. A. Iyer, and M. Abramovici, "Identifying sequentially untestable faults using illegal states," *Proc. VLSI Test Symp.,* pp. 4-11, May 1995.

[5] K.-T. Cheng, "On removing redundancy in sequential circuits," *Proc. of 28th Design Automation Conference,* pp. 164-169, June 1991.

[6] K.-T. Cheng, "Redundancy removal for sequential circuits without reset states," *IEEE Trans. on CAD,* vol. 12, no. 1, Jan. 1993.

[7] X. Lin, I. Pomeranz, and S. M. Reddy, "On finding undetectable and redundant faults in synchronous sequential circuits," *Proc. of Int'l. Conf. on Computer Design,* Oct. 1998.

[8] T. E. Marchok, A. El-Maleh, W, Maly, and J. Rajski, "A complexity analysis of sequential ATPG," *IEEE Trans. on CAD,* vol. 15, no. 11, pp. 1409-1423, Nov. 1996.

[9] I. Pomeranz and S. M. Reddy, "Design and Synthesis for Testability of Synchronous Sequential Circuits Based on Strong-Connectivity," The 20th Int'l. Symp. on FTCS-23. Digest Papers., pp. 492-501, Aug. 1993.

[10] A. Ghosh, S. Devadas, and A. R. Newton, Sequential Logic Testing and Verification, Kluwer Academic Publishers, 1978.

[11] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vinventelli, "SIS : A system for sequential circuit synthesis," *Electronics Research Laboratory Memorandum,* no.UCB/ERL M92/41, 1992.

[12] T. M. Niermann and J. H. Patel, "HITEC : A test generation package for sequential circuits," *Proc. of EDAC,* pp. 132-135, May 1994.

저자소개

최 호 용(평생회원)
1980년 2월 서울대학교 전자공학과 졸업(공학사)
1982년 2월 한국과학기술원 전기 및 전자공학과 졸업 (공학석사)
1994년 3월 오오사카대학교 전자공학과 졸업(공학박사)
1980년 3월~1985년 7월 삼성반도체연구원, 선임연구원
1985년 8월~1996년 8월 부경대학교 전임강사, 조교수, 부교수
1996년 9월~현재 충북대학교 전기전자컴퓨터공학부 교수
<주관심분야 : VLSI설계, DFT & Testing>

김 수 현(정회원)
1996년 2월 광운대학교 컴퓨터과학과 졸업(이학사)
1998년 2월 광주과기원 정보통신공학과 졸업(공학석사)
2005년 2월 광주과기원 정보통신공학과 졸업(공학박사)
2005년 3월~현재 삼성SDI 책임연구원
<주관심분야 : 비동기회로테스팅, Synthesis for Testability, DFT>