초기화가 불가능한 플립플롭을 이용한 시험 불가능 고장 검출에 관한 연구

이 재 훈† · 조 진 우†† · 민 형 복†

요 약

본 논문에서는 순차회로에서의 시험 불가능 고장을 찾는 새로운 알고리즘을 제시하였다. 이 알고리즘에서는 초기화가 불가능한 플립플롭을 먼저 찾으면서 이 과정에서 플립플롭의 초기화를 막는 고장, FPI를 찾고 이 고장의 전파 경로를 검색한다. 또한 이 알고리즘을 ISCAS89 벤치마크 회로를 대상으로 적용하여 시험 불가능한 FPI의 갯수를 제시하였다. 테스트 생성에 소요되는 시간의 대부분이 시험 불가능고장의 검출에 사용되는 것을 고려할 때, 이 알고리즘을 테스트 생성기의 전처리 과정으로 사용하면, 테스트 생성기의 효율을 크게 높일 것으로 기대된다.

A Study on Identifying Undetectable Faults Using Uninitializable Flip-Flops

Jae Hoon Lee† · Jin Woo Cho†† · Hyoung Bok Min†

ABSTRACT

Undetectable faults in a digital circuit are faults that no input patterns can detect. Identifying these faults in test generation process is very time-consuming especially for sequential circuits. In this paper we present a new algorithm to identify undetectable faults in sequential circuits. In the algorithm, we identify uninitializable flip-flops and then, faults that prevent initialization of the flip-flops(FPIs) are identified, finally propagation path of the FPI is checked. Time complexity of this algorithm is proportional to the product of the number of flip flops with at least a self loop and the number of gates in the circuit. Experiments were performed on the ISCAS89 benchmark circuits to show the feasibility of the proposed algorithm. We could identify large amount of undetectable faults (up to 50% of the number of flip-flops) in circuits with uninitializable flip-flops. Considering that most of the time in test generation is consumed in identifying undetectable faults, performance of test generator can be improved by using this algorithm as a pre-processing of test generation.

1. 서 론

회로의 테스트 복잡도(complexity)는 테스트과정에서 비롯되는 모든 비용(costs)으로 대신해서 말할 수 있다. 이 비용에는 여러 가지가 있는데, 테스트 패턴 생성(test pattern generation)비용, 결함 시뮬레이션(fault simulation)[1] 비용과 결함의 위치정보 발생(generation of fault location information)비용, 테스트 장비(test

※본 연구는 1996년도 한국과학재단 연구비 지원에 의한 결과임(과제번호 96-0102-16-01-3).

† 정 회 원:성균관대학교 전기공학과
†† 정 회 원:Motorola Korea Limited

논문접수:1997년 2월 1일, 심사완료:1997년 4월 15일

equipment)비용, 테스트과정 자체에 관련된 비용, 즉 결함(fault)을 찾는데 드는 시간과 그 결함을 제거하는데 드는 시간 등이 포함된다.

전체 테스트 비용의 대부분을 차지하는 테스트 생성기(test generator)에서 결과는 "시험 가능한 고장검출율(detectable fault coverage)"로 표시된다. 이것은 전체 고장 중에서 시험 불가능 고장을 미리 제거하고 계산된 결과이다. 이 "시험 가능한 검출율"은 매우 의미있는 수치인데, 왜냐하면 예를 들어 99 퍼센트의 고장 검출율을 목표로 할 때 이 목표는 회로 전체고장의 2 퍼센트가 시험 불가능고장(undetectable fault)인 회로에서는 결코 얻을 수 없기 때문이다. 100 퍼센트의 시험 가능한 고장검출율은 하나의 완전한 일 처리(job)를 의미한다. 왜냐하면, 고려된 각각의 모든 고장에 대해서 테스트 생성기가 테스트 패턴(test pattern)을 찾아냈거나 또는 그 고장이 시험 불능하다는 것을 증명했다는 뜻이기 때문이다. 그 생성된 테스트 집합(test set), 또는 테스트 시퀀스(test sequence)는 "완벽"하다고 말할 수 있는 것이 모든 각각의 시험가능 고장을 검출할 수 있기 때문이다.[1]

회로의 시험 불가능고장이란 그 고장을 검출할 입력 패턴이 없는 고장을 말한다. 이 고장은 테스트 생성(test generation)을 어렵게 하고, 특히 순차회로의 경우는 더욱 심하다. 테스트 생성의 성능을 증가시키려면, 이런 시험 불가능고장을 테스트 생성과정 중보다는 그 전에 미리 검색할 수 있어야 한다.

시험 불가능고장을 두 가지 종류로 분류할 수 있다. 즉, 조합회로의 시험 불가능고장(redundant fault)과 순차회로의 시험 불가능고장이다. 조합회로에서의 redundant 고장은 한 타임 프레임(one time frame) 내의 테스트 생성과정에서 검색할 수가 있다.

그러나, 순차회로에서의 시험 불가능고장은 다시 redundant 고장과 redundant 하지 않지만 시험 불가능한 고장(irredundant-but-undetectable fault)으로 세분화된다. 여기서 순차회로의 redundant 고장은 원래 회로의 기능동작에는 아무런 영향을 미치지 않는다. 즉, 입력 시퀀스를 인가했을 때, 고장이 없는 회로와 고장회로(fault-free and faulty circuit)가 똑같은 출력 응답을 보인다. 그러나, irredundant 하지 않지만 시험 불가능한 고장은 주로, 고장회로나 고장이 없는 회로에서 플립플롭을 초기화하지 못하는 데서 기인하고, 이 고장은 테스트 생성기가 테스트 시퀀스를 만드는 것을 방해한다.

테스트 생성과정 동안에, 순차회로의 시험 불가능 고장을 검색하는데는 많은 시간 프레임이 소요되며, 때때로 제한된 시간 내에 성공하지 못할 수도 있다. 이런 고장을 찾는데는 많은 시간 소모와 테스트 생성 과정의 효율성을 저하시킨다.

앞에서도 언급하였듯이, 순차회로에서의 시험 불가능고장으로 인해 테스트 생성에 어려움을 겪고 있다. 그리고, 대부분의 시험 불가능고장이 주로 상태 초기화 실패, 즉 FPI(Faults that Prevent Initialization)로 인한 것이다[1]. 따라서, 고장이 존재하는 경우에서 상태 초기화가 되지 않는 플립플롭을 찾아내는 것이 전체 테스트 생성 시간 및 고장 검출율을 높이는데 매우 중요하다.

FPI를 찾는 알고리즘은 이미 제시되었다[1][2]. [1]에서는 FPI를 찾는 알고리즘을 처음으로 제시 하였다. 그러나 이 알고리즘은 모든 초기화 시퀀스(sequence)를 찾아 내기가 어렵고, 컴퓨터 계산 시간이 많이 소모되는 단점이 있다. 이를 개선하기 위하여 [2]에서 새로운 알고리즘을 제시하였는데 여기서는 Taboo 논리 시스템[3]을 도입하여 체계적인 알고리즘을 만들었고, 알고리즘을 고속화 하였다. 본 연구에서는 시험 불가능고장의 대부분을 차지하는 FPI를 새로운 방법[2]을 기반으로 시험 불가능한 FPI 테스트 패턴 생성의 전처리 과정으로 수행하는 알고리즘을 제시하고자 한다.

따라서, 본 논문에서는

첫째, 회로 소자의 uncontrollable logic state를 효율적으로 표현하기 위해 개선된 taboo 논리 시스템[3]을 도입한 초기화가 불가능한 플립플롭을 찾는 알 고리즘[2]을 소개하고,

둘째, 초기화가 불가능한 플립플롭을 찾는 알고리즘을 이용하여 시험 불가능고장을 찾 는 알고리즘을 제시하고 그것을 구현하고자 한다.

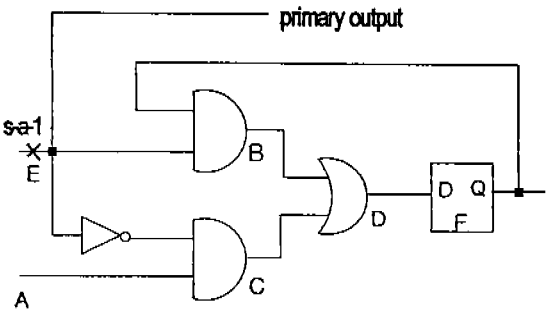
2. FPI와 플립플롭의 초기화

1) 결함 시뮬레이션은 고장 시뮬레이션이라고도 한다.

순차회로에는 초기화가 불가능한 플립플롭이 존재할 수 있다. 본 논문에서는 이러한 플립플롭의 검출 알고리즘이 사용되는데 이 알고리즘에 대해서는 이미 발표되었다.[2] 그러나 본 논문의 완성도를 높이기 위해 초기화가 불가능한 플립플롭을 찾는 알고리즘을 2장에 소개한다.

2.1 FPI(Faults that Prevent Initialization)

FPI를 검출하는 알고리즘은 Abramovici과 Parikh [1]에 의하여 제안되었다. Cheng[4]에 의하면 FPI는 궤환루프가 있는 순차회로에서만 발생한다. (그림 1)에서 고장이 없는 경우, 플립플롭 F는 (E, A)=00 또는 (E, A)=01에 의해서 초기화가 가능하다. 즉, (E, A)=00인 경우는 신호선 E에 논리값 0를 배정하고, 신호선 A에는 논리값 0을 배정한다. 신호선 E에 0을 배정하게 되면, 신호선 B 는 나머지 입력단에 무관하게 항상 논리값 0을 갖게 되고, 신호선 C는 신호선 A에 따라 결정되게 된다. 신호선 B의 논리값 0은 신호선 D의 값을 신호선 C에 따라 결정되게 만든다. 결국 신호선 D의 값은 신호선 A에 따라 결정되어, 신호선 A에 0을 배정하면 플립플롭 F는 0으로 초기화가 된다. 마찬가지로 (E, A)=01로 논리값을 배정하면 플립플롭 F는 1로 초기화가 된다.



(그림 1) 플립플롭의 초기화를 막는 고장
(Fig. 1) Faults that prevent initialization of flip-flops

그러나, 신호선 E에 stuck-at-1[2] 고장이 존재할 경우, 플립플롭 F는 초기화될 수 없다. 왜냐하면, 신호선 E에 stuck-at-1이 존재하게되면, 신호선 C는 신호선 A의 값에 무관하게 항상 논리값 0을 갖게되고, 신호선 B는 플립플롭 F의 출력단에 연결된 나머지 입력단의 논리값에 따라 결정된다. 신호선 C의 논리값 0은 신호선 D의 값을 신호선 B에 따라 결정되게 만든다. 결국 플립플롭 F 입력단의 논리값은 신호선 D에 따르고, D는 신호선 B에 따르고, B는 플립플롭 F의 출력단 Q에 따르게 된다. 즉 신호선 E의 stuck-at-1에 의해서 다른 주입력단 A에 무관하게, 항상 플립플롭 F의 출력단의 uncontrollable state 논리값이 궤환(feedback) 경로를 통하여 플립플롭 F의 입력단에 전달되고 주입력단에서는 플립플롭 F를 논리값 0또는 1로 초기화할 수 없게 된다. 따라서 신호선 E의 stuck-at-1은 플립플롭의 초기화를 막게되고, 이러한 고장을 FPI라 한다.

2.2 초기화가 불가능한 플립플롭

2.2.1 FPI가 존재하기 위한 충분 조건

본 논문의 목적은 ATPG(Automatic Test Pattern Generator)[3]의 전처리(pre-process)단계로 순차회로에서의 시험 불가능고장을 찾는 것이다. 그러므로, 모든 FPI를 검출하기보다는 고장 회로(faulty circuit)에서 초기화가 불가능한 플립플롭을 찾고, 주어진 플립플롭이 FPI를 갖고있는지를 검증하고 시험 불가능고장인지를 판단하고자 한다. 여기서는 주어진 플립플롭에 FPI가 존재하기 위한 충분조건을 소개한다[2].

[정리 1]

임의의 노드(node) P에 논리값 V(0 or 1)를 고정한 것이, 다른 노드의 논리값에 관계없이 회로내의 플립플롭 F의 un-initialized output 상태를 그 플립플롭 F의 입력단으로 전파시키는 그런 노드 P가 하나라도 존재하면, 그 플립플롭 F에 대한 FPI가 존재한다.

---

2) stuck-at-1 고장이란, 회로의 고장모델중 하나로, 임의의 신호선에 논리값 1이 고정또는 고착되있는 고장을 말한다. (그림 1)에서는 s-a-1으로 표시하였다. stuck-at-0 고장도 있으며, stuck-at 고장을 고착고장이라고도 번역한다.

3) 여기서 ATPG란, 회로에 주어진 임의의 고장들에 대하여 그 고장들을 검출할 수 있는 주입력단의 입력값(input value)패턴을 자동으로 생성해주는 발생기를 말한다.

[증 명]

그런 노드 P가 고정된 논리값을 갖고 있으면, 플립플롭 F의 다음 상태는 계속 un-initialized 상태를 유지하게 된다. 따라서 노드 P의 stuck-at-V은 FPI가 된다.

[정리 2]

[정리 1]에 의해서 찾은 초기화가 불가능한 모든 플립플롭을 고장 회로에서 초기화가 가능하게 만들면, FPI에 의해 초기화가 불가능한 다른 플립플롭은 없게 된다

[증 명]

[정리 1]에서 찾은 초기화가 불가능한 플립플롭은 [정리 1]에서 찾지 못하는 다른 플립플롭의 초기화도 불가능하게 만들기 때문이다.

[정리 2]로부터 초기화가 불가능한 플립플롭을 이용하여 시험 불가능고장을 검출하고 ATPG의 전처리 단계로 사용하는데 당위성을 찾을 수 있다. 즉, 대부분의 시험 불가능고장이 상태 초기화 불능에 기인하고[4], 이러한 고장은 고장 검출율이나 컴퓨터 계산 시간면에서 테스트 생성의 성능(test generation performance)을 치명적으로 저하시킬 수 있기 때문이다.

2.2.2 Taboo Logic Value

Taboo logic value는 constraint propagation 문제를 처리하기 위하여 도입되었다[3]. Taboo logic value는 근본적으로 unknown (or un-initialized) value를 표시하며 constraint에 의하여 어떠한 논리적(logical) implications들을 얻을 수 없는가에 대한 의미를 갖는다. 즉, 본 논문에서는 플립플롭의 초기화되지 않는 논리값을 의미한다.

회로 전체의 노드를 4 가지 그룹으로 분류할 수 있다.

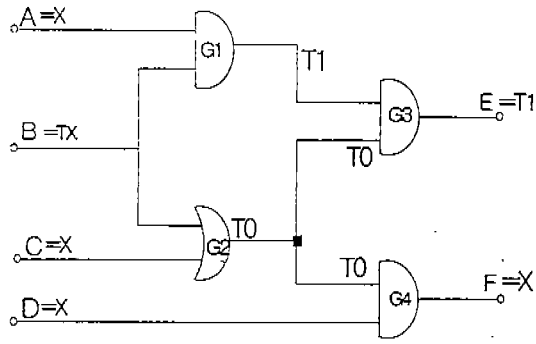
그룹 1. unknown(uncontrollable) 값을 갖는 노드, 즉 논리값으로 1이나 0을 가질 수 없는 노드를 의미하며, TX로 표시한다.

그룹 2. 논리값 0이 배정될 수 없는 노드들로, T0로 표시한다.

그룹 3. 논리값 1이 배정될 수 없는 노드들로, T1로 표시한다.

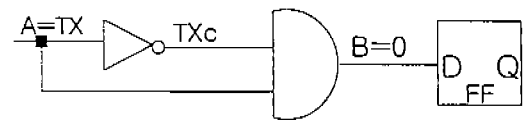
그룹 4. constraint에 무관하게 논리값 0나 1 모두 배정될 수 있는 노드들 로, X로 표시한다.

Taboo logic value TX를 적용한 결과를 보이면 (그림 2)과 같다.



(그림 2) Taboo logic value를 이용한 Constraints의 전파
(Fig. 2) Propagation of constraints using taboo logic value

Taboo logic value는 근본적으로 정보손실(loss of information) 문제로 인한 부정확한 논리 연산을 할 수 있다. 이러한 현상은 three-valued logic system(0, 1, U)에서 발생하는 문제와 근본적으로 같다. 이러한 문제는 (그림 3)과 같이 TX의 보수(complement) 논리값인 TXc를 사용하여 방지할 수 있다. TXc는 TX와 같은 의미이나 TXc의 보수로 정보손실을 방지할 수 있다. 그러나, TXc는 회로내에 constraint가 하나만 존재할 때에 한하여 TX에 의한 부정확한 논리 연산을 방지할 수 있음에 주의하여 한다. 본 논문에서는 constraint가 플립플롭의 출력단에 한 개만 존재하므로 TXc를 도입할 수 있다.



(그림 3) TXc 논리값의 사용
(Fig. 3) Use of the TXc logic value

이 Taboo logic value는 기존의 논리값 시스템과 손쉽게 통합 사용 가능하다. 즉, 전체적으로 seven-valued

4) 이렇게 말할 수 있는 근거를 본 논문의 제 4장 실험 및 결과의 표 4에서 제시한다.

logic system(0, 1, X, T0, T1, TX, TXc)이 되다. 이 seven-valued logic calculus는 다음 표와 같다.

〈표 1〉 AND calculus
〈Table 1〉 AND calculus

	0	1	X	T0	T1	TX	TXc
0	0	0	0	0	0	0	0
1	0	1	X	T0	T1	TX	TXc
X	0	X	X	X	T1	T1	T1
T0	0	T0	X	T0	T1	TX	TXc
T1	0	T1	T1	T1	T1	T1	T1
TX	0	TX	T1	TX	T1	TX	0
TXc	0	TXc	T1	TXc	T1	0	TXc

〈표 2〉 OR calculus
〈Table 2〉 OR calculus

	0	1	X	T0	T1	TX	TXc
0	0	1	X	T0	T1	TX	TXc
1	1	1	1	1	1	1	1
X	X	1	X	T0	X	T0	T0
T0	T0	1	T0	T0	T0	T0	T0
T1	T1	1	X	T0	T1	TX	TXc
TX	TX	1	T0	T0	TX	TX	1
TXc	TXc	1	T0	T0	TXc	1	TXc

〈표 3〉 NOT calculus
〈Table 3〉 NOT calculus

0	1	X	T0	T1	TX	TXc
1	0	X	T2	T0	TXc	TX

2.2.3 초기화가 불가능한 플립플롭을 찾는 알고리즘

FPI의 검출은 하나의 플립플롭에 대하여 그 출력단의 초기화되지 않는 논리값을 궤환루프를 통하여 입력단으로 강제적으로 전달시키는 고정된 논리값을 갖는 신호가 있는 지를 조사함으로써 가능하다. 다른 표현으로는, stuck-at 고장에 의하여 플립플롭이 초기화되지 않으려면, 회로 내에 한 개의 신호선이 0 또는 1의 고정된 값을 가짐으로써 플립플롭의 출력단의 초기화되지 않는 논리값이 그 플립플롭의 입력단까지의 신호전달 경로를 sensitize시켜야 한다. 여기서 sensitize라 함은, 입력 테스트 시퀀스 T에 의한 어떤 신호선의 논리값이 고장에 의해서 바뀌게 될 때 그 신호선은 그 고장에 의해 sensitize 되었다고 하고, 이런 sensitize된 신호선들로 이루어지 경로를 sensitized path 라고 한다.

이를 위해 PODEM[5]도 비슷한 방법을 쓰는데, 초기화되지 않는 플립플롭을 검출하기 위하여 ATPG에서 사용하고 있는 forward/backward implication 과정에 taboo logic value를 함께 사용한다. 궤환루프가 있는 각 플립플롭에 대하여 FPI가 존재하는지 여부를 조사하려한다. 알고리즘은 아래 (그림 4)와 같다.

```
==========================================================================
 Find_Uninitializable_FF( input: a f/f F, output: init_flag)
 {
1  Assign TX at the output of F and perform forward implication from F.
2  Create a TX-frontier T;
3  for each logic element G in T
4    for each fanin_gate of G with logic value X{
5      init_flag= Backward_implication(fanin_gate, a non-controlling value V of G)
6      if ( init_flag = NO) , return ( NO ); //un-initializable
     }
7  return (YES); //initializable
 }

  Backward_implication(G, V)
  {
```

```
8    if G is a fanout stem or a Primary Input {
       do forward_implication with V;
9      if the input of F is either TX or TXc, return (NO); //un-initializable
10     else undo the implicated values;
     }
11   if G is not a Primary Input {  //recursive call
12   init_flag=Backward_implication(fanin_gate of G, a logic value to justify V at G);
     if (init_flag=NO), return (NO);
     8}
13 return (YES);
   }
```

(그림 4) 초기화가 불가능한 플립플롭 검출 알고리즘.
(Fig. 4) An algorithm of identifying unintializable flip-flops

궤환루프상의 각 플립플롭에 대하여 Find_Uninitailizable_FF()이 수행된다. (그림 4) 알고리즘에서 forward 및 backward implication이 수행되는데, implication은 time frame을 무시하고 forward implication 시 플립플롭의 입력 논리값이 출력으로 전달된다. 또한, backward implication 시에도 플립플롭의 출력 신호값이 입력단으로 justify된다.

알고리즘의 2 줄에서 TX-frontier를 다음 세 가지조건을 만족하는 모든 논리 소자로 정의한다. 1) 출력단 논리값이 0 또는 1로 고정되지 않는다. 2) 입력단에 TX 또는 TXc를 갖는다. 3) 그 논리 소자로부터 대상 플립플롭까지의 경로가 존재한다.

3. 시험 불능고장의 검출 알고리즘

앞에서도 언급하였듯이, 시험 불능고장을 검출하는 것은 대단히 중요한 일이다. 제 3장에서는 초기화가 불가능한 플립플롭을 이용하여, 시험 불가능고장을 찾는 새로운 방법을 제시하고자 한다. 즉, 초기화가 불가능한 플립플롭을 찾는 알고리즘을 약간 변경하여 플립플롭의 초기화를 막는 고장, FPI를 찾는다. 물론 여러 개의 플립플롭이 하나의 FPI에 의해 초기화가 불가능해지는 경우도 있다. 다음으로 각각의 FPI에 대하여 그 FPI에 의해 초기화가 되지 않는 플립플롭들을 지나지 않고 주출력단으로 전파경로(propagation path)가 있는지를 조사한다. 만약 전파경로가 없다면 이 고장은 시험 불가능하다고 말할 수 있다.

이 알고리즘은 다음 (그림 5)과 같다.

Step 1에서는, 주입력단에서 제어 가능(controllable)하고 주출력단에서 관찰 가능(observable)한 자기궤환(self-loop)이 있는 플립플롭에 대하여 초기화가 불가능한 플립플롭을 검출한다. Step 2에서는, Step 1에서 찾은 각각의 플립플롭에 대한 FPI를 테이블로만든다. 이때 하나의 FPI에 의해 여러 플립플롭이 초기화가 불가능할 경우도 있다. 그 예를 다음 (그림 6)에서 볼 수 있는데[5], 신호선 a의 stuck-at-0 고장에 의하여 플립플롭 F1과 F2가 모두 초기화가 불가능하게 된다. 왜냐하면, 신호선 a에 0이 고정되면, 신호선 $b=0$, $c=0$, $d=1$, $f=0$, $j=0$, $k=0$, $n=0$이 배정되고, 신호선 $b=0$은 신호선 i의 보수 논리값을 그대로 전달시키고, 신호선 $d=1$은 신호선 e의 논리값을 그대로 전달시키고, 신호선 $f=0$은 신호선 g의 보수 논리값을 그대로 전달시킴으로써, 플립플롭 F1의 초기화되지 않은 (un-initialized)상태를 플립플롭의 입력단으로 그대로 전달시키고, 결국 플립플롭 F1은 초기화가 불가능하게 된다. 마찬가지로, 신호선 $k=0$은 신호선 p의 보수 논리값을 그대로 전달시키고, 신호선 $d=0$은 신호선 l의 논리값을 그대로 전달시키고, 신호선 $n=0$은 신호선 m의 보수 논리값을 그대로 전달시킴으로써, 플립플롭 F2의 초기화를 불가능하게 한다. Step

5)(그림 6)은 실제 s344.bench 회로의 일부분을 보인 것이다.

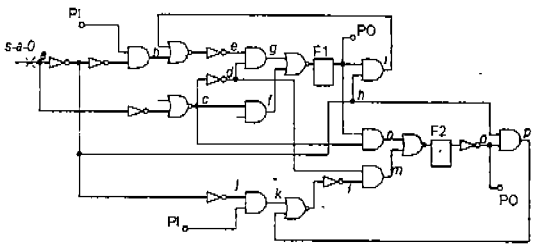
```
Step 1. for each flip-flop with  self-loop
        perform  Find_Uninitializable_FF() and identify un-initializable flip-flops;

Step 2. Make fault-list that prevents initialization of flip-flops found in the Step 1.

Step 3. for each fault {
          check propagation path;
          if( reach PO only through un-initializable flip-flops)
            the fault is undetectable;
        }
```

(그림 5) 시험 불능고장 검출 알고리즘
(Fig. 5) An algorithm of identifying undetectable faults

3에서는, Step 2에서 찾은 각각의 FPI에 대하여 그 FPI에 의해 초기화가 불가능한 플립플롭을 통해서만 주출력단으로 전파하는지를 체크한다. (그림 6)에서 신호선 a의 stuck-a-0 고장은 꼭 플립플롭 F1과 F2를 거쳐야지 만 주출력단으로 전파될 수가 있다. 다시 말하면, F1과 F2를 거치지 않고는 주출력단까지의 경로가 없고, 결국 이 stuck at 고장은 시험 불가능한 고장이 된다.



(그림 6) 시험 불능고장(undetectable FPI)의 예
(Fig. 6) Example of the undetectable FPI

4. 실험 및 결과

본 논문의 시험 불가능고장을 찾는 알고리즘은 C 언어로 구현되었으며, 32MB의 주메모리를 갖은 Sun Sparc-20 workstation 에서 수행되었다. ISCAS89 benchmark 회로에 대한 결과를 아래 〈표 4〉에 보였다. 여기서는 1991년에 변경된 새로운 ISCAS89(이름 끝에 .1이 덧붙어 있다)를 사용하였다.[6]

〈표 4〉에서 두 번째 열(column)의 [# total ffs]는 각 회로에 있는 모든 D 플립플롭의 개수를 의미하고, 세 번째 열의 [# self loop ffs]는 그 D 플립플롭 중에서 자기궤환을 갖고 있는 플립플롭의 개수이다. 이 자기궤환을 갖고 있는 플립플롭이 Find_Uninitializable_FF()의 입력(input)으로 들어가게 된다.

네 번째 열의 [# uiffs]는 자기궤환을 갖는 플립플롭 중에서 초기화가 불가능한 플립플롭의 개수이고, 다섯 번째 열의 [# FPI]는 초기화가 불가능한 플립플롭을 만드는 FPI의 개수를 의미한다. s208.1.bench 에서 초기화가 불가능한 플립플롭의 수는 8개이고, FPI는 8개이다. 그러나, s344.bench 회로에서는 초기화가 불가능한 플립플롭은 8개이고, FPI는 2개이다. 이는 곧, 하나의 FPI가 여러개의 플립플롭의 초기화를 불가능하게 만들 수 있음을 의미하는 예이다.

앞에서도 언급하였듯이, 다섯 번째 열에서 찾은 모든 FPI가 시험 불능한 고장은 아니다. 여섯 번째 열의 [# undetectable faults]는 FPI중에서 시험 불능한 고장의 개수를 보여주고 있다. 이것이 본 논문에서 제안한 알고리즘을 이용하여 찾은 시험 불가능고장이다.

〈표 4〉 실험 결과
〈Table 4〉 Experimental results

circuit name	# total ffs	# self loop ffs	# uiffs	# FPIs	# undetectable faults	CPU time(sec)
s208.1	8	8	8	8	1	0.34
s298	14	14	1	1	0	0.69
s344	15	15	8	2	1	0.41
s349	15	15	8	2	1	0.45
s382	21	15	0	0	0	0.98
s386	6	6	0	0	0	1.71
s420.1	16	16	16	16	3	0.36
s444	21	15	0	0	0	1.08
s510	6	6	2	1	0	4.38
s526	21	21	1	1	0	2.15
s526n	21	21	1	1	0	2.09
s641	19	15	1	1	0	0.42
s713	19	15	1	1	0	0.43
s820	5	5	0	0	0	6.76
s832	5	5	0	0	0	7.59
s838.1	32	32	32	32	7	0.65
s953	29	6	5	5	0	1.18
s1196	18	0	0	0	0	0.28
s1238	18	0	0	0	0	0.27
s1423	74	71	12	4	2	5.99
s1488	6	6	0	0	0	9.79
s1494	6	6	0	0	0	9.36
s5378	179	0	0	0	0	0.63
s9234.1	211	90	41	11	4	101.86
s13207.1	638	274	161	39	20	345.44
s15850.1	534	369	219	29	11	823.48
s35932	1728	288	0	0	0	590.46
s38417	1636	1042	898	898	896	931.13
s38584.1	1426	1072	14	13	0	6379.01

마지막 일곱 번째 열에서는, ISCAS89 회로를 읽어 회로의 내부구조(structure)를 만드는데 걸린 CPU 시간(time)은 제외하고, 회로의 자귀궤환을 갖은 플립플롭을 찾는 데부터 시험 불가능고장을 찾는데 까지 걸린 CPU 시간(second)을 기록한 것이다.

이 표에서 의미하는 것은, 물론 본 연구에서의 알고리즘을 구현한 결과를 보이고자 한 것도 있지만, 일반적으로 순차회로의 시험 불가능고장을 검출할 때 순차회로용 ATPG를 이용하여 많은 시간적 소모와 낮은 효율을 수반하는 문제점에 대해 본 연구에서 제안한 새로운 알고리즘으로 순차회로에서의 시험 불능고장을 미리 찾아내고 순차회로용 ATPG를 그 후에 사용할 수 있다는 중요한 의미를 지니고 있다.

5. 결 론

본 연구에서는 시험 불가능고장의 상당부분을 차지하는 시험 불가능한 FPI가 되기 위한 충분조건을

소개하고 이를 기반으로 시험 불가능한 FPI를 찾는 새로운 알고리즘을 제시하였다. 이 알고리즘에서는 초기화가 불가능한 플립플롭을 먼저 찾으면서 이 과정에서 플립플롭의 초기화를 막는 고장, FPI를 찾은 후 그 고장의 전파 경로를 검색한다. 또한 이 알고리즘을 ISCAS89 벤치마크 회로를 대상으로 적용하여 시험 불가능한 FPI의 갯수를 제시하였으며, 실험 결과 일부회로에서는 플립플롭 갯수의 절반까지 시험 불가능한 FPI를 찾을 수 있었다.

이 방법은 테스트 생성기에 의존하지 않을 뿐만 아니라 time complexity도 회로내의 게이트 수와 회로의 자기궤환 플립플롭의 수의 곱에 비례하므로 비교적 효율적이다. 테스트 생성에 소요되는 시간의 대부분이 시험 불가능고장의 검출에 사용되는 것을 고려할 때, 이 알고리즘을 테스트 생성기의 전처리 과정으로 사용함으로써 테스트 생성기의 효율을 높일 것으로 기대되며, 멀리는 전체 테스트 비용을 줄이는데 기여할 것으로 기대된다.

참 고 문 헌

[1] M. Abramovici, and P.S. Parikh, "WARNING: 100% fault coverage may be misleading !!," *Proceedings of International Test Conference*, pp. 662-668, 1992.

[2] E. S. Park, and H. B. Min, "An Algorithm for Partial Scan Flip-Flop Selection Using Functional and Topological Analysis," CAD 및 VLSI 설계 연구회 학술발표회 논문집, pp. 37-42, 1996

[3] E.S. Park, "A Pragmatic Test Pattern Generation System for Scan-Dsigned Circuits with Logic Value Constraints," *IEEE Asian Test Sysmposium*, pp. 2-7, November 1993.

[4] K-T. Cheng, "On removing redundancy in sequential circuits," *Proceedings of Design Automation Conference*, pp. 164-169, 1991.

[5] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Transactions on Computer*, Vol.C-30, pp. 215-222, March 1981.

[6] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Proceedings of International Symposium on Circuits and Systems*, pp. 1929-1934, May 1989.

이 재 훈

1991년 2월 성균관대학교 공과대학 전기공학과(공학사)

1993년 2월 성균관대학교 공과대학 전기공학과(공학석사)

1993년 1월~1996년 2월 LG전자기술원 주임연구원

1996년 3월~현재 성균관대학교 전기공학과 박사과정 재학

관심분야: VLSI CAD/Testing, Low power design

조 진 우

1995년 2월 성균관대학교 공과대학 전기공학과(공학사)

1997년 2월 성균관대학교 공과대학 전기공학과(공학석사)

1997년 1월~현재 Motorola Korea Limited

관심분야: VLSI CAD/Testing

민 형 복

1980년 2월 서울대학교 공과대학 전자공학과(공학사)

1982년 2월 한국과학기술원 전기 및 전자공학과(공학석사)

1990년 12월 The University of Texas at Austin 전기 및 컴퓨터 공학과 공학박사

1982년 3월~1985년 4월 금성통신(주) 연구소 주임연구원

1985년 8월~1986년 7월 미국 Columbia대학교 연구원

1991년 3월~현재 성균관대학교 전기공학과 부교수

관심분야: VLSI Testing, CAD 시스템