유사조합 회로로의 변환에 기초한 부분 스캔 기법을 이용한 디지탈 순차 회로의 테스트 기법 연구

論文
43~3~18

Test Generation of Sequential Circuits Using A Partial Scan Based on Conversion to Pseudo-Combinational Circuits

閔 炯 福*
(Hyoung-Bok Min)

Abstract—Combinational automatic test pattern generators (CATPG) have already been commercialized because their algorithms are well known and practical, while sequential automatic test pattern generators (SATPG) have been regarded as impractical because they are computationally complex. A technique to use CATPG instead of SATPG for test generation of sequential circuits is proposed. Redesign of sequential circuits such as Level Sensitive Scan Design (LSSD) is inevitable to use CATPG. Various partial scan techniques has been proposed to avoid full scan such as LSSD. It has been reported that SATPG is required to use partial scan techniques. We propose a technique to use CATPG for a new partial scan technique, and propose a new CATPG algorithm for the partially scanned circuits. The partial scan technique can be another choice of design for testability because it is computationally advantageous.

Key Words: Test(시험), Automatic Test Pattern Generation(자동 테스트 생성), Partial Scan(부분 스캔), Pseudo-Combinational Circuits(의사 조합 회로)

1. 서 론

모든 반도체 소자는 테스트를 거쳐서 출고한다. 그 이유는 반도체 소자의 수율이 100%가 되지 못하기 때문이다. 즉 모든 반도체 소자는 여러 가지 원인에 의하여 항상 상당한 불량률을 보이기 때문에 테스트를 거쳐서 합격된 소자만을 출고하여야 한다. 그런데 이 테스트 기법에는 여러 가지가 있을 수 있으나 근래에는 인건비의 상승, 디지탈 IC의 고집적화 등으로 인하여 사람의 눈으로 하나하나 확인하여 테스트하는 것이 지나친 비용이 들게 되었다. 따라서 적절한 테스트 벡터를 디지탈 IC에 인가하여 그 반응을 살피는 테스트 장비(ATE, Automatic Test Equipment)의 사용이 불가피하게 되었다. 그런데 문제는 이 테스트 장비에 사용할 테스트 벡터를 어떻게 생성 시킬수 있는가 하는 것이다. 숙련된 테스트 엔지니어가 그 경험에 의하여 테스 벡터를 생성시켜 주는 것도 한 방법이지만 테스트 엔지니어의 훈련도 문제이거니와, 현재의 고집적도를 가진 디지탈 IC 테스트의 경우(특히 ASIC이나 마이크로 프로세서, 각종 콘트롤러등) 숙련된 테스트 엔지니어 조차도 테스트 벡터를 생성시키는 것이 불가능하게 되었다. 따라서 각종 수학적 알고리즘을 이요하여 테스트 벡터를 자동으로 생성시키야 하게 되었다. 즉 디지탈 회로를 입력으로 넣어 주면 그 디지탈 회로를 테스트하기 위한 일련의 테스트 벡터를 자동으로 생성시켜 주는 소프트웨어(혹은 드물게는 하드웨어)가 개발되었다[1,2,3]. 지난 10여년간 이러한 소프트

*正 會 員：成均館大 工大 電氣工學科 助敎授·工博
接受日字：1993年 7月 26日
1 次修正：1993年 12月 14日

웨어에 관한 연구가 각광을 받게 된 것은 같은 기간 디지탈 IC의 집적도가 폭발적으로 증가하여 테스트 벡터를 생성시키는 작업 자체의 비용이 폭발적으로 증가하였기 때문이다.

최근에는 테스트 벡터 자동 생성 시스템이 상업용 소프트웨어 시스템으로 판매되고 있기도 하다. 그러나 문제는 상업용 소프트웨어들은 모두 조합 회로(combinational circuit)를 그 대상으로 하는 자동 테스트 패턴 생성기(Combinational automatic test pattern generator, CATPG)인 반면, 고집적도를 가진 대부분의 디지탈 IC들은 순차 회로(sequential circuit)라는데 문제가 있다. 즉 상업용 소프트웨어들을 그대로 적용시킬 수 없다는 점이다. 이에 순차 회로에 스캔 기법을 이용하여 조합 회로로 변환해 주는 기법[4]이 개발되어 현재 널리 이용되고 있다. 즉 순차 회로의 기억소자(플립플롭)를 쉬프트레지스터의 형태로 설계하여 그 기억소자에 직접 접근할 수 있도록 변환하는 것이다. 그리하여 사실상 기억소자의 입출력이 회로 전체의 입출력 처럼 동작하도록 함으로서 순차 회로를 조합 회로로 변환하는 것이다.

그러나 이 기법을 사용하기 위해서는 기억소자를 보통의 플립플롭과 쉬프트레지스터의 이중으로 사용할 수 있도록 설계되어야 하기 때문에 기억소자가 차지하는 면적이 상당히 증가하여(area overhead) 디지탈 IC의 고집적화에 방해요인이 되며, 두가지 용도로 사용하기 때문에 부가되는 논리 소자들이 유발하는 지연 시간으로 인하여 디지탈 IC의 고속동작에 방해가 되어(performance overhead)디지탈 IC 설계자들이 스캔 기법의 사용을 꺼리는 것이 현실이다. 이 난점을 부분적으로나마 개선하기 위하여 최근 부분 스캔 기법이 연구되어 왔다. 즉 순차 회로의 모든 메모리 소자에 스캔 기법을 적용(완전 스캔 : full scan)하지 아니하고, 선택된 일부의 메모리 소자에만 스캔 기법을 적용(부분 스캔 : partial scan)함으로써 오버헤드를 상당히 줄이자는 것이다. 지금까지 개발된 방법들[5,6,7]은 기대대로 오버헤드를 줄일 수 있으나 새로운 문제가 발생되었다. 부분 스캔된 순차 회로는 조합 회로로 변환되지 아니하고 순차 회로로 남기 때문에 실제 사용자의 입장에서는 이미 투자된 상업용 소프트웨어를 사용할 수 없다는 점이다. 또한, 순차 회로에 사용되는 자동 테스트 벡터 생성 시스템[8,9]은 아직도 대학에서 실험 개발하는 단계에 있으며, 확립된 알고리즘이 존재하지 않아서 상업용 소프트웨어도 존재하지 않는 것이 사실이기 때문에 이미 투자된 상업용 소프트웨어를 사용할 수 없다는 것은 대단히 심각한 문제이다.

본 연구는 이를 해결하는 방법에 대한 연구이다. 즉 부분 스캔 기법을 적용하되 부분 스캔을 적용할 메모리 소자를 선택하는 새로운 방법을 개발함으로써 회로가 순차 회로이면서도 조합 회로와 유사하게 되는 유사 조합 회로(pseudo-combinational circuit)개념을 개발하고, 이 개념을 이용하여 변환된 회로에 상업용 소프트웨어에서 채용되는 CATPG 기법을 사용할 수 있음을 보이려는 것이다. 즉 기억소자를 선택할 때 선택된 기억소자를 제외한 나머지 기억소자들 간에 궤환(feedback)이 생성되지 않게 함으로써 유사 조합회로를 형성시키는 것이 본 연구의 핵심 사항이다.

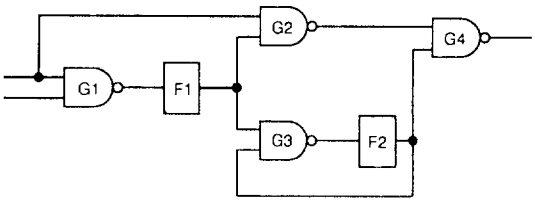
2. 문제 제기-기존 자동 테스트 패턴 생성 알고리즘의 문제점

오늘날 대부분의 디지탈 회로는 순차 회로로 이루어져 있으므로 이러한 회로에 대한 테스트 패턴 생성 방법이 널리 연구되었다. 그 대표적인 예가 스캔 설계 기법 [4]이다. 이 방법은 순차회로내의 플립플롭들을 모두 특수한 형태(스캔 가능한 플립플롭)로 설계하여 회로를 조합 회로로 변환시킨 후 이 회로에 대하여 조합 회로를 위한 테스트 패턴 생성 기법을 적용시키는 방법이다. 이 기법은 간단하지만 특수한 형태의 플립플롭이 칩의 면적을 많이 차지하고 회로 지연시간을 증가시켜서 경우에 따라 심각한 문제점으로 부각되었다. 이러한 문제를 해결하기 위하여 회로내의 플립플롭중 일부만 스캔 가능한 플립플롭으로 설계하는 부분 스캔(partial scan)기법[5, 6,7]이 제안되었으나 이 경우에는 변환된 회로가 순차 회로가 되는 것이 문제이다.

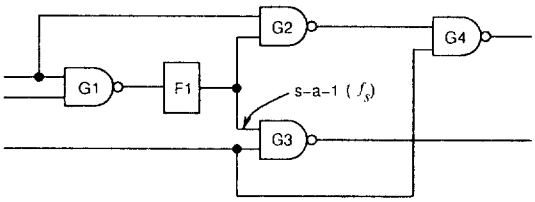
이제 이 문제를 좀더 자세히 검토하여 보기로 한다. 그림 1(a)는 테스트하려는 순차회로의 한 예이다. 이와 같은 회로를 테스트하기 위하여 우선 결함모델을 선정하여야 한다. 결함모델은 일반적으로 디지탈 회로에 이용되는 실리콘 반도체 소자에서 널리 사용되고 있는 stuck-at 결함

모델을 사용하기로 한다. stuck-at 결함모델의 경우에도 잘 알려진 바와 같이[10], 순차회로용 테스트 패턴 생성기(Sequential automatic test pattern generator, SATPG)는 테스트벡터를 만들어 낸다는 보장도 없고, 비효율적이며 상업용 소프트웨어가 존재하지도 않는다. 기업의 입장에서는 이런 이유로 완전 스캔 기법을 이용하거나 랜덤 테스트 기법을 이용하였다. 그러나 칩의 집적도가 증가함에 따라 랜덤 테스트 기법은 한계를 보이고 있으며 완전 스캔은 그 오버헤드 때문에 문제가 된다. 그래서 부분 스캔이 필요하게 되었다. 즉, 더 크기가 큰 스캔 가능한 플립플롭의 사용량을 감소시킴으로써 오버헤드를 줄일 수 있다.

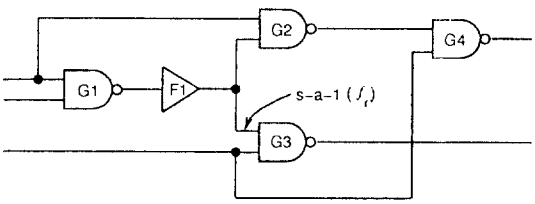
그림 1(b)에 부분 스캔 기법을 이용한 그림을 보였다. 그림 1(a)에 있는 2개의 플립플롭 중에서 F2에만 스캔가능한 플립플롭을 사용함으로써 F2



(a) A Finite State Machine Under Test



(b) Type-S Circuit (Pseudo-Combinational Circuit)



(c) Type-C Circuit Derived from (b)

그림 1 부분 스캔된 회로 예제

Fig. 1 An Example of Partially Scanned Circuit

의 입/출력 핀은 전체 회로의 입/출력으로 변환되고 F1는 그대로 남는다. 이렇게 변환된 그림 1(b)의 회로를 살펴보면 이 회로에는 궤환 경로가 존재하지 않는다. 물론 플립플롭 내부에는 궤환 경로가 존재할 것이나, 플립플롭을 하나의 회로 소자로 해석하면 이 회로에는 궤환 경로가 존재하지 않는다. 즉, 이 회로는 조합회로에 가까운 형태를 취한다. 따라서 상업용 소프트웨어로 판매되고 있는 조합 회로를 위한 자동 테스트 벡터 발생 프로그램을 사용할 수 있을 것이라는 결론이 도출된다. 다만 플립플롭이 여전히 회로내에 남아 있으므로 이 플립플롭을 어떻게 취급할 것인가의 문제가 남는다.

이것이 기존의 테스트 패턴 생성 알고리즘을 수정하여야만 이 문제에 적용할 수 있는 이유이다. 이 답에 대해서는 Gupta의 논문이 힌트를 주고 있다[11]. 논문에서는 이런 플립플롭을 그대로 전선(wire)으로 모델하고 있다. 이와 같은 전선 모델은 적극 검토할 만한 대상이다. 그러나 Gupta의 논문에서 다루고 있는 회로는 본 연구에서 다루고자 하는 유사 조합 회로와는 거리가 있다. 즉 그 논문의 회로는 각 회로 레벨마다 D 플립플롭이 들어 있어서, D 플립플롭의 존재가 별 영향을 주지 아니하므로 그대로 전선으로 대치할 수 있으나, 본 논문에서 제안하는 회로는 회로의 각 레벨에 D 플립플롭이 들어갈 수도 있고 들어가지 않을 수도 있는, 즉 Gupta의 논문에서 다룬 회로보다 더욱 일반적인 회로이다. 이와 같은 회로를 대상으로 하는 이유는 물론 실제적인 적용이 가능한 기법을 개발하기 위함이다. 즉 Gupta의 논문에서 다룬 바와 같이 회로 레벨마다 D 플립플롭이 규형적으로 들어가는 회로는 실제 거의 없다고 해도 과언이 아닐 것이다.

3. 순차 회로의 유사 조합 회로로의 변환

제안된 부분 스캔 기법과 일반적인 순차 회로를 부분 스캔된 회로로 변환하는 기법을 여기에서 기술하려 한다. 여기서 부분 스캔된 회로는 플립플롭 내부에만 궤환이 존재하고 외부에는 궤환이 존재하지 않는다. 이러한 디지탈 회로를 유사 조합 회로라고 부르기도 한다.

3.1 부분 스캔된 순차 회로

순차 회로의 부분 스캔 기법은 여러 가지가

제안된 바 있으나, 여기서 제안하는 방식에 가장 근접한 것은 Cheng[7]에 의하여 제안된 것이다. 그러나 Cheng의 방식은 여전히 조합 회로를 위한 ATPG를 사용할 수 없는 방식이다. 부분 스캔된 디지탈 회로의 정의를 다음과 같이 내려보고자 한다.

[정의 1] Type-S 회로

Type-S회로는 플립플롭 외부에 궤환 경로가 존재하지 않는 동기 순차회로(Finite State Machine : FSM)로 정의한다.

[정의 2] Type-C회로

Type-C회로는 Type-S회로의 D 플립플롭을 버퍼 게이트로 대치한 회로이다.

위와 같이 정의된 회로 중에서 Type-S회로가 순차 회로를 부분 스캔한 유사 조합 회로이다. 앞으로는 유사 조합 회로라는 용어 대신에 Type-S회로라는 용어를 사용하기로 한다. 여기서 Type-C회로는 완전한 조합 회로임에 유의하여야 한다. 이와 같은 Type-C회로에는 조합 회로를 위한 ATPG 기법(CATPG)을 그대로 적용할 수 있다. Type-S회로와 Type-C회로의 예는 이미 보인바 있다. 제 2절에서 보인 그림 1(b)가 Type-S회로이며 이 회로의 D 플립플롭을 버퍼 게이트로 바꾸면 Type-C 회로가 된다.

이제 부분 스캔된 회로를 정의하였으므로 일반적인 동기 순차 회로를 Type-S 회로로 변환하는 방법과 변환된 Type-S 회로에 대하여 자동 테스트 패턴 생성 기법을 제시하기로 한다.

3.2 Type-S 회로로의 변환

동기 순차 회로를 Type-S회로로 변환하는 과정은 회로 내의 플립플롭을 스캔 가능한 플립플롭으로 대치하여 서로 연결함으로써 이루어진다. 이와 같이 형성된 스캔 고리(scan chain)가 부분 스캔 기법을 이용한 테스트의 하드웨어적인 근간을 이룬다. 따라서 Type-S회로로의 변환이란 궤환을 제거할 수 있는 스캔 고리에 들어갈 플립플롭을 선택하는 알고리즘의 개발을 의미한다.

우선 문제를 정확히 정의해 보자. G=(V,E)를 동기 순차 회로를 나타내는 회로 그래프(graph)라 하자. 여기서, V는 게이트나 플립플롭 같은 회로소자를 가리키는 그래 의 노드이고, E는 회로소자들 사이의 연결을 위한 선을 의미하는 그래프의 연결선(edge)을 나타낸다.

$F=\{f_1, f_2, f_3, \cdots, f_n\}$은 G에 들어 있는 플립플롭들의 집합이라 하자. 이때, F는 G의 부분 집합임이 명백하다. $C=\{C_1, C_2, \cdots, C_m\}$이 그래프(회로) 내의 모든 싸이클(cycle)의 집합이라 하자. 집합 F의 플립플롭, f_1가 scan chain에 들어갈 때 회로에서 제거되는 싸이클의 집합을 C_1라 하고, 집합 S는 $S=\{C_1, C_2, \cdots, C_n\}$으로 정의한다. 이제 S의 부분 집합 $T=\{T_1, T_2, \cdots, T_k\}$의 원소의 합집합이 C가 된다면 T의 원소에 해당하는 D 플립플롭들의 집합이 우리가 선택하고자 하는 플립플롭들이다. 이때 문제는 T의 원소의 갯수를 어떻게 최소화 하느냐가 된다. 즉, T의 원소의 합집합이 C가 되도록 하는 원소의 개수가 가장 적은 T가 무엇인가 하는 문제이다.

이 문제는 잘 알려져 있는 바와 같이 minimum set cover 문제라 하며, 이 문제는 NP-hard 문제로 잘 알려져 있다[12]. 즉 다항식 해(polynomial time solution)는 존재하지 않을 것으로 믿어진다. 따라서, 최소해(minimal solution)를 주는 것으로 알려진 greedy 알고리즘을 이용하여 문제를 풀고자 한다.

우선 동기 순차 회로 내의 싸이클을 알아내기 위하여 Tarjan [13]에 의하여 발표된 깊이 우선 탐색 알고리즘인 CLASSIFY를 사용한다. CLASSIFY 알고리즘은 방향성 그래프의 연결선들을 4가지로 분류하며, 그 중에서 frond라고 이름 붙여진 연결선들이 궤환을 나타내므로 이 연결선이 싸이클을 나타내므로 이를 궤환 경로라고 부르기로 한다. 싸이클을 없애기 위해서는 이 궤환 경로에 연결된 플립플롭 스캔 고리에 넣으면 싸이클이 없어진다. 따라서 greedy 알고리즘에서는 모든 궤환 경로를 조사하여 궤환 경로의 집합에 가장 자주 나타나는 플립플롭부터 스캔 고리에 넣도록 하면 된다. Greedy 기법을 이용하여 플립플롭을 선택하는 알고리즘을 다음 그림 2와 같이 쓸수 있다. 이 알고리즘은 테스트할 대상이 되는 동기 순차 회로의 회로 그래프를 입력으로 하여 모든 궤환을 없앨 수 있도록 하는 플립플롭들의 집합을 만들어 낸다.

그림 2의 1단계에서 회로의 연결선을 분류하여 궤환 경로의 집합을 만들어 낸다. 궤환 경로의 양쪽의 노드가 같으면 이는 self-loop를 나타낸다. 모든 싸이클을 없애려면, 이러한 self-loop는 반드시 없어져야 하므로 self-loop를 갖는 플립플롭을 우선적으로 스캔 고리에 넣도록 한다(2,3단계). 4,5,6,7 단계들이 greedy 알고리즘 부분이다. 5,6 단계에서 궤환 경로의 집합에 있는

```
알고리즘 : Type-S 회로가 되도록 flipflop을 선택

입    력 : finite state machine을 나타내는 directed circuit graph, G = (V, E)
출    력 : scan chain에 들어갈 flipflop의 집합, F
방    법 :
begin
      1.  CLASSIFY ;
      2.  Identify self-loops from a list of fronds ;
      3.  Add all the nodes with self-loops in F ;
      4.  while ( the list of fronds is non-empty ) {
      5.      Choose a node, v, which occurs most frequently
                  in the list of fronds ;
      6.      Add v in F ;
      7.      Remove all the fronds with v at their heads or tails
                  from the list of fronds ;
      8.  }
end.
```

그림 2 Type-S 회로로 변환하는 알고리즘

Fig. 2 An Algorithm to convert to a Type-S circuit

플립플롭 중에서 가장 자주 나타나는 플립플롭 하나를 선택하여 스캔 고리에 넣고, 이 플립플롭을 스캔 고리에 넣었을 때 없어지는 모든 궤환 경로를 궤환 경로의 집합에서 제거한다(단계 7). 이러한 과정을 궤환 경로의 집합에서 모든 궤환 경로가 사라질 때까지 반복하여 최종적으로는 스캔 고리에 넣을 플립플롭들의 집합을 F에서 얻게 된다.

앞서 언급한 바와 같이, 이 방법을 사용할 때, 싸이클을 없앨 수 있는 최소 갯수의 플립플롭이 집합 F에 들어간다는 보장은 없다. 그러나 그에 매우 유사한 결과(minimal solution)를 얻을 수 있다.

3.3 변환 알고리즘의 구현

그림 2에서 기술된 변환 알고리즘을 SUN 워크스테이션의 SunOS 하에서 C 언어를 사용하여 구현하였다. 구현된 알고리즘을 ISCAS89 회로[14]에 적용한 결과를 표1에 보였다. 그 결과 크기가 작은 회로에서는 거의 모든 플립플롭들을 사용해야 Type-S회로로의 변환이 가능하지만, 크기가 큰 회로(예를 들어 s953 이상)에서는 경우에 따라 아주 작은 갯수의 플립플롭만 가지고도 Type-S 회로로 변환 가능함을 알 수 있었다.

4. Type-S 회로에 대한 테스트 패턴 생성

앞에서 부분 스캔 기법을 소개하였고, 그러한 Type-S 회로를 구하는 기법을 알고리즘의 형태로 기술하였다. 이제 여기서는 그러한 알고리즘을 적용하여 얻은 Type-S 회로에 대하여 테스트

표 1 ISCAS89 회로에 대한 그림2의 알고리즘 적용 결과

Table. 1 Partial Scan Result for ISCAS89 Circuits

ISCAS89 회로	회로내 flipflop수 (A)	scan chain 내의 flipflop수 (B)	비 율 B/A (%)
s27	3	3	100.0
s208	8	8	100.0
s298	14	14	100.0
s344	15	15	100.0
s349	15	15	100.0
s382	21	15	71.4
s386	6	6	100.0
s400	21	15	71.4
s420	16	16	100.0
s444	21	15	71.4
s510	6	6	100.0
s526	21	21	100.0
s641	19	15	78.9
s713	19	15	78.9
s820	5	5	100.0
s832	5	5	100.0
s838	32	32	100.0
s953	29	6	20.7
s1196	18	0	0.0
s1238	18	0	0.0
s1423	74	71	95.9
s1488	6	6	100.0
s1494	6	6	100.0
s5378	179	34	19.0
s9234	228	160	70.2
s13207	669	313	46.8
s15850	597	446	74.7
s35932	1728	576	33.3
s38417	1636	1087	66.4
s38584	1452	1131	77.9

벡터를 자동 생성하는 방법을 기술하기로 한다.

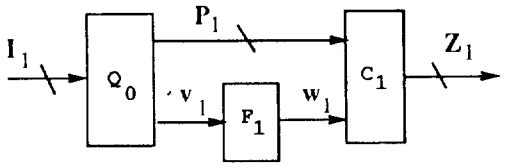
Type-S 회로(유사 조합 회로)에 대한 자동 테스트 패턴 생성법에 대해서는 Miczo[15]에 의하여 주어진 바 있다. 그 알고리즘은 순차 회로에 대한 고전적 ATPG 알고리즘인 반복 배열 모델(iterative array model)에 기반을 두고 있다. 그러나 본 연구의 목적은 이러한 순차 회로에 대한 ATPG 알고리즘보다는 조합 회로에 대한 ATPG 알고리즘을 이용함으로써, ATPG를 효율적으로 함에 있다. 이를 위해서는 다음의 정의에 기반을 둔 사실에 유의할 필요가 있다.

[정의 3] T^n, T_o^n

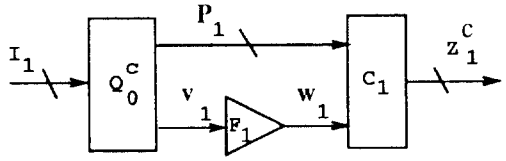
$T^n=\{T_1, T_2, T_3, \cdots, T_n\}$은 n개의 테스트벡터의 집합이라 하자. 이때,

T_o^n는 $T_1=T_2=T_3=\cdots=T_n$인 경우의 T^n를 나타과 $t=0$에서 Q_1^c의 Z_1^c값은 같다. 따라서 $m=1$에 대하여 정리는 성립한다.

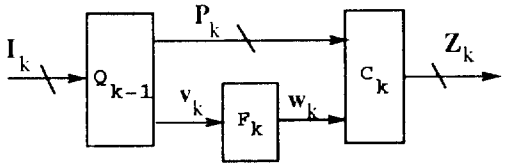
$m=k-1$: $m=k-1$일 때 정리 1이 성립한다고 가정하자 이는 T를 $Q_{k-1}{}^c$의 결함에 대한 테스트 벡터라고 할 때 $T_0{}^k$는 $Q_{k-1}{}^c$의 테스트 벡터임을



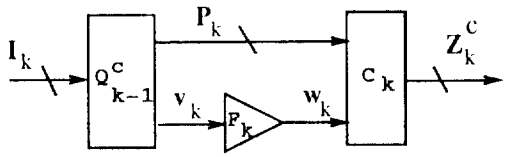
(a) Type-S Circuit Q_1



(b) Type-C Circuit Q_1^c Derived from Q_1



(c) Type-S Circuit Q_k



(d) Type-C Circuit Q_k^c Derived from Q_k

그림 3 정리 1의 증명을 위한 일반적인 Type-C 및 Type-S

Fig. 3 Generalized Circuits for Proof of Theorem 1

의미한다. 즉 t=k−1에서는 Q_{k-1}회로의 출력 Z_{k-1}와 $Q_{k-1}{}^c$회로의 출력 $Z_{k-1}{}^c$ 값이 같다는 것을 의미한다.

m=k : 그림 3(c)는 플립플롭이 k개 들어 있는 Type-S회로 모델이다. 이에 해당하는 Type-C회로는 그림 3(d)에 보였다. 그림 3(d)의 Q_{k-1} 는 그 플립플롭 대신에 버퍼 게이트가 k−1개 들어 있다. 테스트 벡터 시퀀스, $T_0{}^k$가 Q_k에 인가되고 T가 $Q_0{}^k$에 인가되었다고 하자. 이때 그림 3(c)와 그림 3(d)의 P_k 및 V_k는 시간 t=k−1에서 같은 값을 갖는다. 이는 m=k−1에서 그렇게 가정했기 때문이다. 이제 클락이 하나 주어져서 시간 t=k가 되었다고 하자 그러면 V_k 값은 W_k 로 넘어가 t=k에서의 W_k는 t=k−1에서의 V_k 과 같아진다. 따라서 t=k에서는 출력 Z_k와 $Z_k{}^c$ 는 같다. 따라서 정리 1은 m=k−1에서 성립한다면 m=k에서도 성립한다.

<증명 끝>

정리 1에서 플립플롭을 m개 포함하고 있는 Type-S 회로의 테스트 벡터는 m+1개 필요하다고 하였다. 그러나, 이는 사실상 필요한 테스트 벡터 갯수의 최대치(upper bound)라고 할 수 있으며 실제로 대부분의 회로에서는 훨씬 작은 갯수로 충분하다. 반복 배열 모델에 기반을 두고 개발된 Miczo의 알고리즘에서는 이 갯수는 회로의 입력에서 출력 사이의 경로에 들어 있는 플립플롭 갯수의 최대치가 필요한 벡터 갯수의 최대치가 됨을 보이고 있다[12]. 이 최대치는 Type-C 회로의 결함에 대한 CAPTG에 근거하여 경로 분석(path analysis)을 하면 더욱 줄일 수 있다. 즉 회로의 모든 입력으로 부터 모든 출력으로의 경로를 분석할 필요가 없고 CATPG에 참여한 경로, 그 중에서도 임계 경로(critical path)[16]에 놓여 있는 플립플롭에 대해서만 분석하면 된다. 그 이유는 임계 경로가 아닌 네트의 논리값을 다른 값으로 변환하여도 CATPG에 전혀 영향을 미치지 못하기 때문이다. 여기서 임계 경로란 그 경로의 논리값이 변화할 때 회로의 출력 값을 변화 시킬 수 있는 네트의 모임을 의미하며 그 예를 그림 4에 보였다. 그림 4는 게이트 C의 위쪽 입력에 stuct-at-1 결함이 있을 때는 것으로 정의한다.

이 정의에 따라 그림 1의 회로를 다시 보자.

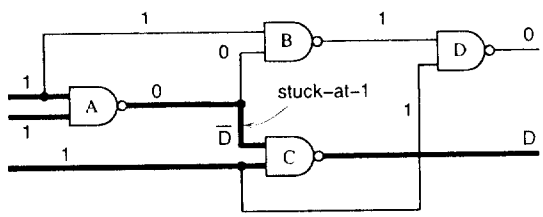


그림 4 테스트 벡터 생성후의 Critical Path

Fig. 4 A Critical Path after Test Vector Generation

그림 1(b)는 테스트하려는 Type-S 회로이다. 그에 해당하는 Type-C 회로가 그림 1(c)에 있다. 그림 1(b)에 표시된 stuck-at-1 결함 (f_s) 와 그림 1(c)에 표시된 stuck-at-1결함 (f_c) 는 같은 결함들이다. 이 결함에 대한 두 회로에서의 테스트 패턴을 생각해 보자. 그림 1(c)의 회로에서 결함 (f_c)에 대한 테스트 패턴은 111이다. 또 그림 6(b)의 회로에서 결함 (f_s) 에 대한 테스트 패턴은 {11X, XX1}이다. 여기서 X는 don't care를 나타내며, 테스트 패턴이 두개인 것은 11X를 먼저 인가한 후에 XX1을 인가해야 함을 나타낸다. 여기서 주목해야 할 것은 (f_s)에 대한 테스트 패턴이 {111, 111}이 될 수 있으며, 이는 (f_c)에 대한 테스트 패턴 111과 같다는 점이다. 이는 결코 우연한 것이 아니며, 이러한 관찰로부터 다음의 정리를 얻을 수 있다.

[정리 1] T를 Type-C회로의 stuck-at 결함, (f)에 대한 테스트 패턴이라 하자. 이때, 길이가 m+1인 테스트 패턴, $T_0^{m+1}=\{T, T, T, \cdots, T\}$는 f와 같은 위치에 있는 (Type-C 회로를 생성시킨) Type-S 회로의 결함에 대한 테스트 패턴이며, 여기서 m은 Type-S 회로의 플립플롭의 갯수이다.

<증 명> 이 정리는 수학적 귀납법에 의하여 다음과 같이 증명할 수 있다.

m=1 : 이 경우는 Type-S 회로의 플립플롭의 갯수가 1개인 경우이다. 이런 회로는 다음 그림 3(a)와 같은 모델로 표시할 수 있다. 이와 같은 회로를 Q_1이라 하자. 여기서 첨자 1은 회로에 들어 있는 플립플롭의 숫자를 나타낸다. 따라서 그림 3(a)의 회로 블럭Q_0와 Q_1 은 조합 회로이다. 그림 3(b)는Q_1 에서 플립플롭을 버퍼 게이트로 대치한 Type-C 회로이다. 이 경우 회로 블럭 Q_0와 Q_0^c 는 같은 회로이다. T를Q_1^c 내의 결함 하나에 대한 테스트 벡터라고 하자. 이때 T가 그림 3(a)의Q_1과 그림 3(b)의 Q_1^c 에 동시(t=0)에 가해졌다고 하자. 이때 P_1 과 V_1값은 동일하다. ($\because$ $Q_0^c=Q_0$). 이제 Q_1에 클락이 인가되어 t=1이 되었을 때 다시 Q_1의 입력에 T가 인가 되었다고 하자, 그러면 t=1에서 Q_1의 W_1값은 V_1값과 같으므로 t=1에서 Q_1의 W_1 값과 t=0에서 Q_1^c의 W_1^c값은 같다. 따라서 t=1에서 Q_1의 Z_1 값

```
Algorithm PATH_ANALYSIS
입  력 :  State of a Circuit after Test Generation for a Target Fault
출  력 :  N = Upper Bound of the Length of a Test Sequence
방  법 :
begin
    Creat a Pseudo output node;
    Connect the primary outputs where faulty value have been propagated to
    the pseudo output node;
    N = flipflop_levelize( the pseudo output node );
end.


flipflop_levelize( v )      /* V is a node */
begin
   mark v visited;
   level <- 1;
   if (v is a primary input)
     return level;
   for every input node w that feeds v {
      if (w is not visited and a critical input) {
         temporary = flipflop_levelize( w );
         if (temporary > level)
            level <- tempory;
      }
   }
   if ( v is a flipflop )
      level <- level + 1;
   return level;
end.
```

그림 5 테스트 벡터 갯수를 구하는 알고리즘
Fig. 5 An Algorithm to Compute Length of Test Sequence

때의 테스트 벡터를 구한 다음, 이에 근거하여 임계 경로를 굵은 선으로 표시하였다. 임계 경로는 결함 값(faulty value)을 갖는 회로의 출력으로 부터 깊이 우선 탐색을 함으로써 구할 수 있다. 깊이 우선 탐색은 각 게이트의 입력의 논리값이 임계값(critical value)일 때에만 계속된다. 예를 들어 2 입력 AND 게이트의 논리값이 0,1이고 출력이 0이라면 2개의 입력중 논리값 0을 갖는 입력만이 임계 경로에 들어간다. 왜냐하면 1은 0으로 바뀌어도 출력이 0으로 그대로 유지되나, 입력이 0인 것은 그 입력이 1로 바뀌면 출력도 1로 바뀌기 때문이다. 임계 경로를 구하는 알고리즘을 다음 그림 5에 도시하였고 그 알고리즘의 이름을 PATH-ANALYSIS라고 하였다.

정리 1과 PATH-ANALYSIS 알고리즘을 적용하면, Type-S 회로의 결함에 대한 테스트 벡터열, $T_0^n=\{T, T, \cdots, T\}$를 구할 수 있다. 여기서 T는 CATPG를 이용하여 구한 테스트 벡터이고 n은 PATH-ANALYSIS 알고리즘으로 부터 구한 것이다. 그러나, 정리 1은 CATPG에 의하여 테

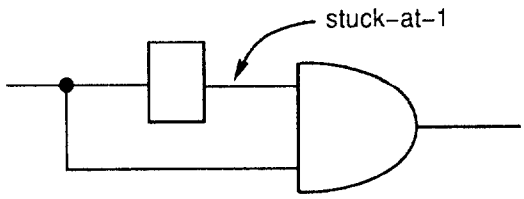


그림 6 Type-C에서는 Redundant Fault이고, Type-S에서는 테스트 가능한 결함의 예
Fig. 6 The Fault is Redundant in Type-C Circuit, while it is Testable in Type-S Circuit

스트 벡터가 구해질 경우에 대해서만 언급하고 있음에 유의 하여야 한다. 즉 CATPG에 의하여 발생한 시험 불가능 결함(redundant fault)의 경우에는 정리 1이 해법을 주지 못하고 있다. 다시 말하면 Type-C회로에서 시험 불가능 결함일 경우에는 Type-S 회로에서는 시험 불가능 결함이거나 테스트 벡터로 구성될 수 없다는 것을 나타낸다. 예를 들어보자. 다음 그림 6의 회로에서 사각형은 D 플립플롭을 표시한다. 그림 6에서 주어진 위치에 stuck-at-1결함이 존재할 때 이 결함에 대한 테스트 벡터는 Type-C 회로에서는 존재하지 않는다. 그러나 Type-S 회로(그림 5의 회로)에서는 $T^2=\{0,1\}$로 시험 가능 하다. 즉 Type-C 회로에서 시험 불가능 결함일지라도 Type-S 회로에서는 테스트 벡터가 존재할 수 있다는 의미가 된다. 이를 결합의 집합 관계로 표시하면 다음과 같다.S, S_t^s, S_r^s, S_t^c 및 S_r^c 를 각각 모든 stuct-at 결함의 집합, Type-S 회로에서 시험 가능한 결함의 집합, Type-S 회로에 시험 불가능한 결함의 집합, Type-C 회로에서 시험가능한 결함의 집합, Type-C 회로의 시험 불가능 결함의 집합이라 하자.

[정리 2]

$$S_t^s \cup S_r^s = S_t^c \cup S_r^c = S, \quad (1)$$

$$S_t^c \leq S_t^s \quad (2)$$

$$S_r^s \leq S_r^c \quad (3)$$

<증 명>

테스트 가능한 결함이 아니면 모두 시험 불가능 결함으로 정의되므로 이는 명백하다.

(2) [정리 1]에서 Type-C 회로의 결함에 대한 테스트 벡터 T일 때의 회로의 테스트 벡터는 T_0^n이므로 (2)가 성립한다.

(3) (1)과 (2)를 표시하는 벤 다이아그램을 이용하여 쉽게 증명 가능하다.

<증명 끝>

위 정리 2의 (3)식은 CATPG를 이용하여 Type-S의 테스트 벡터열을 구할 때의 CATPG의 결과가 시험 불가능 결함으로 나타났다고 하여 모두 Type-S 회로에서도 시험 불가능 결함으로 판단할 수 없음을 나타낸다.

이 문제를 완전히 해결하는 방법은 Miczo[15]가 제시한 바와 같은 순차 회로에 대한 ATPG를 사용하는 것이다. 그러나 회로 내의 대부분의 시험 불가능 결함은 그 결함이 존재하는 부근의 회로들을 조사함으로써(local redundancy checking)알아낼 수 있다. 이와 같은 시험 불가능 결함을 찾아내는 알고리즘을 개발하여 그 이름을 LOCALITY라고 지었으며 다음 그림 7에 그 알고리즘을 보였다.

LOCALITY 알고리즘에서 Type-C 회로에 대한 CATPG의 결과 주어진 결함이 시험 불가능 결함일 때 그것이 그 결함 부근의 회로를 검사하여 시험 불가능 결함임을 알아낼 수 있는지를 검사한다. 즉 플립플롭을 넘어가지 않는 범위 내에서 그 결함이 나타내는 논리값(implication)을 회로에 할당하여 이것이 서로 모순되지 않는지를 검사한다. 이 방법은 PODEM[1] 알고리즘에서 쓰이는 방법을 플립플롭을 고려하여 고친 것이다. 따라서 CATPG의 결과 시험 불가능 결함으로 밝혀진 것들에 LOCALITY 알고리즘을 적

```
Algorithm LOCALITY
입 력 : A Redundant Fault in Type-C Circuit
출 력 : A Flag Indicating either REDUNDANT in Type-S circuit or UNKNOWN.
방 법 :
begin
    Assign values implied by the fault without passing flipflops;
    if ( there were conflicts )
        return REDUNDANT;
    Check x paths from D-frontiers to primary outputs;
    if ( there is no x-paths )
        return UNKNOWN;
    else
        return UNKNOWN;
end.
```

그림 7 Local Redundant Fault를 찾는 알고리즘
Fig. 7 An Algorithm to Find Local Redundant Faults

용하여 Type-S 회로 내에서의 시험 불가능 결함과 포기된 결함(aborted fault)로 분류해 낸다. 여기서 생겨난 포기된 결함들은 SATPG를 적용해야만 테스트 벡터 생성이 가능한 것이다.

새로운 부분 스캔 기법에 의하여 생성된 Type-S 회로에 대한 자동 테스트 패턴 생성은 CATPG와 SATPG의 두 단계로 가능함을 보였다. 즉 Type-S 회로가 순차 회로라고 해서 SATPG를 적용해야 하는 것은 아니다. SATPG가 일반적으로 CATPG에 비하여 많은 시간을 소모하므로 대부분의 결함에 대하여 CATPG를 적용하고 소수의 결함에 대하여 SATPG를 적용하면 ATPG의 성능을 개선할 수 있다. 또한 제안된 부분 스캔 기법을 적용한 회로에는 플립플롭 내부를 제외하면 궤환이 존재하지 않으므로 SATPG가 매우 쉽게 이루어질 수 있다. 이를 정리하여 하나의 알고리즘으로 나타내면 그림 8과 같다.

```
Algorithm Proposed_ATPG
입 력 : A Circuit under Test and Fault set
출 력 : Test pattern set
방 법 :
begin
    Random Test Generation with Fault Simulation;
    for every remaining target fault {
        CATPG;
        if ( redundant fault ) {
            flag = LOCALITY;
            if ( flag == REDUNDANT )
                return REDUNDANT_FAULT;
            else
                SATPG;
        }
        else
            PATH_ANALYSIS;
        if ( test generated )
            do fault simulation;
    }
end.
```

그림 8 부분 스캔 기법을 적용한 회로에 대한 자동 테스트 패턴 생성 알고리즘

Fig. 8 ATPG Algorithm for the Proposed Partially Scanned Circuits.

이 알고리즘에서 CATPG는 PODEM[1], FAN[2]과 같은 알고리즘들을 이용할 수 있으며 SATPG는 ESSENTIAL[9], Marlett[8], Miczo의 알고리즘[12]들을 이용할 수 있을 것이다. LOCALITY와 PATHANALYSIS 알고리즘을 이미 기술한 바 있다.

5. 제시된 자동 테스트 패턴 알고리즘 시험

이제까지 부분 스킨 기법을 제안하였고 그에 따른 ATPG 기법을 제시하였다. 이 기법들을 SUNOS가 사용되는 Sparcstation IPC에서 C언어를 사용하여 구현하였다. CATPG에는 FAN을 SATPG에는 Miczo의 알고리즘을 사용하였다. 실험에는 ISCAS89 benchmark회로를 사용하였다. 이 회로는 순차 회로에 대한 실험에 사용될 수 있도록 고안되어 있으므로 본 논문의 목적상 적합하다고 할 수 있다.

구현된 ATPG 시스템은 그림 8에 따라 구성하였다. 이 방법은 자동 테스트 패턴 생성 시스템을 구현할 때 일반적으로 쓰이는 방법이다. 즉 처음에 주어진 목표 결함(target faults) 중에서 랜덤 벡터에 의하여 테스트하기 쉬운 결함은 제거시키고 남은 테스트하기 어려운 결함에 대해서만 알고리즘을 이용하는 ATPG에 적용한다. 그 결과 시험 불가능 결함이거나 포기된 결함(aborted fault)은 테스트 결과 화일에 기록하고, 테스트 벡터를 생성하였을 경우에는 이를 결함 시뮬레이터에 넣어서 테스트 가능한 결함을 제거하고, 나머지 목표 결함 중의 하나를 골라 다시 ATPG를 수행한다. ATPG에서 backtracking 100번을 넘으면 포기된 결함으로 간주하였다. 처음의 목표 결함들은 stuck-at 결함에 대해서 그림 9의 실험을 수행하였다. 실험 결과를 표 2에 도시하였다. 여기서 F_A 는 CATPG의 결과 포기된 결함, F_R^C은 CATPG의 결과 시험 불가능 결함, F_R^S은 SATPG까지 적용했을 때의 시험 불가능 결함이다. FC는 CATPG 만을 적용했을 경우 ATPG 결과 얻은 시험 가능결함 비율(fault coverage)을 나타낸다. 여기서 시험 가능 결함 비율이란 테스트 벡터를 발생시킨 결함의 수와 시험 불가능 결함으로 밝혀진 결함의 수의 합의 목표 결함의 갯수에 대한 비율을 말한다. FC 값만으로도 매우 높은 시험 가능 결함 비율을 나타내고 있으며, 여기에 CATPG에 실패한 결함에 대하여 SATPG를 적용한 결과 FC1로 표시된 시험

표 2 자동 테스트 패턴 생성 실험 결과
Table 2 ATPG Results : Fault Coverage

Circuits	F_A(%)	F_R^C(%)	F_R^*(%)	FC(%)	FC_i
s382	0.00	0.00	0.00	100.00	100.00
s400	0.00	,1.42	1.18	99.76	100.00
s444	0.00	2.95	2.74	99.79	100.00
s641	0.00	0.00	0.00	100.00	100.00
s713	0.00	6.54	6.54	100.00	100.00
s953	0.00	0.00	0.00	100.00	100.00
s1196	0.00	3.38	0.00	96.62	96.86
s1238	0.00	9.23	1.48	92.25	96.09
s1423	0.00	0.92	0.92	100.00	100.00
s5378	0.17	7.58	0.80	93.05	98.11
s9234	0.97	7.23	3.20	95.01	98.64
s13207	0.10	9.27	0.86	91.48	93.49
s15850	0.07	4.23	2.79	98.49	99.04
s35932	0.00	11.42	3.72	92.30	98.77
s38417	0.00	1.74	0.47	98.73	98.80
s38584	0.00	4.94	3.38	98.41	99.36

F_A = Aborted Faults in Type-C Circuits
F_R^C= Redundant Faults in Type-C Circuits
F_R^*= Redundant Faults in Type-S Circuits
FC = Fault Coverage after CATPG = 100 - (F_R^C- F_R^* + F_A)
FC_i = FC + additional fault coverage from additional SATPG

가능 결함 비율을 얻을 수 있었다.

6. 결 론

본 연구에서는 새로운 부분 스캔 기법과 그에 따른 자동 테스트 패턴 생성 알고리즘을 제시하였다. 새로운 부분 스캔 기법을 사용하면 일반적인 순차 회로를 Type-S 회로로 변환하며, 이는 플립플롭 내부의 궤환을 제외한 모든 궤환을 없애기 때문에 ATPG의 효율이 매우 높아진다. 조합회로를 위한 FAN 알고리즘을 변형한 ATPG를 적용하여 ISCAS89 회로들에서 대부분 95% 이상의 시험 가능 결함 비율을 얻었으며 일부 회로에 대해서는 100%도 가능하였다. 테스트 할 수 없는 결함이 남을 경우 그에 대해서만 Miczo의 알고리즘을 적용하면 될 것이며 Miczo의 알고리즘은 사실상 조합 회로에 대한 ATPG의 변형이다.

본 연구에서 제시된 결과가 최선이라고는 볼 수 없을 것이다. 즉 모든 회로에 대하여 본 연구에서 제시된 부분 스캔 기법이 가장 우수한 테스트 결과를 준다고는 할 수는 없으나, 완전 스캔(full scan)기법이나 종래의 부분 스캔 기법과 비교할 때 여기서 제시한 부분 스캔 기법만으로도 높은 fault coverage를 얻을 수 있는 회로들이 존재함을 실험을 통하여 보였으며, 이 경우 CATPG를 사용할 수 있었다. 따라서, 회로 시험을 위하여 스캔 기법을 이용하고자 할 때 선택할 수 있는 우수한 설계 기법 중의 하나라고 말할 수 있다.

본 연구는 1992년도 한국과학재단의 연구비 지원(KOSEF 923-0800-006-1)에 의해 연구되었으며 이에 감사드립니다.

참 고 문 헌

[1] P.Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," IEEE Transactions on Computers, vol c-30, No.3, pp.215-222, March 1981.

[2] H.Fujiwara, and T.Shimono, "On the Acceleration of Test Generation Algorithms," IEEE Transactions on Computers, vol c-32, No.12, pp.1137-1144, Dec 1983.

[3] M.H.Schulz, and E. Auth, "Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques," Proceedings of the 18th Symposium on Fault Tolerant Computing, pp. 30-35, June 1988.

[4] E.B.Eichelberger, and T.W.Williams, "A Logic Design Struture for LSI Testing," Proceedings of the 14th Design Automation Conference, pp.462-468, June 1977.

[5] E.Trischler, "Testability Analysis and Incomplete Scan Path," Proceedings of the International Conference on Computer~Aided Design, pp.38-39, Oct 1983.

[6] V.D.Agrawal, K.-T.Cheng D.D. Johnson, and T. Lin. "A Complete Solution th the Partial Scan Problem," Proceedings of the International Test Conference, pp.44-51, 1987.

[7] K-.T.Cheng, and V.D.Agrawal, "A Partial Scan Method for sequential Circuits with Feedback," IEEE Transactions on Computers, vol.c-39, No.4, p.544-548, April 1990.

[8] R.A.Marlett, "An Effective Test Generation Systems for Sequential Circuits," Proceedings of the 23rd Design Automation Confer-

ence, pp.250-256, June 1986.

[9] M.H.Schulz, and E. Auth, "ESSENTIAL : An Efficient Self Learning Test Pattern Generation Algorithm for Sequential Circuits," Proceeding of the International Test Conference, pp.28-37, August 1989.

[10] A.Miczo, "The Sequential ATPG : A Theoretical Limit," Proceeding of the International Test Conference, pp. 143-147, 1983.

[11] R.Gupta, R.Gupta, and M.A.Breuer, "The BALLAST Methodology for Structured Partial Scan Design," IEEE Transactions on Computers, vol. c-39, No.4, pp.538-544, April 1990.

[12] A.V.Aho, et.al, "The Design and Analysis of Computer Algorithms," Addison-Wesley Publishing Company, Reading Massachusetts, pp.378-392, 1974.

[13] F.Brglez, D.Bryan, and C.Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," Special Session on Benchmarks for sequential Test Pattern Generation, Proceedings of the International Symposium on Circuits and Systems, 1989.

[14] R.Tarjan, "Finding Dominators in Directed Graph," SIAM Journal on Computing, vol.3, pp. 62-89, 1974.

[15] A.Miczo, Digital Logic Testing and Simulation, Harper & Row, Publishers, New York, 1986.

[16] M.Abramovici, P.R.Menon, and T. Miller, "Critical Path Tracing : An Alternative to Fault Simulation," IEEE Design and Test of Computers, pp.83-93, Feb. 1984.

저 자 소 개

민형복(閔炯福) 1958년 2월 22일생. 1980년 서울대 공대 전자공학과 졸업. 1982년 한국과학기술원 전기 및 전자 공학과 졸업(석사). 1990년 미국 University of Texas at Austin 졸업(박사). 1982년~1985년 금성통신 연구소 주임연구원. 1985년~1986년 미국 Columbia 대학교 연구원. 1991년~현재 성균관대학교 전기공학과 조교수.