논문 2007-44SD-11-12

고장 대상 후보를 줄이기 위한 패턴 비교 알고리즘

(A Pattern Comparison Algorithm for Pruning Fault Candidates)

조 형 준*, 강 성 호**

(Hyungjun Cho and Sungho Kang)

요 약

본 논문에서는 패턴 비교를 통해 고장 대상 후보를 줄이는 알고리즘을 제안한다. 고장 대상 후보의 개수는 고장 진단을 위한 고장 시뮬레이션 시간을 결정한다. 그렇기 때문에 전체 고장 진단 시간을 줄이기 위해서는 고장 대상 수를 줄이는 것이 필수적이다. 임계 경로 추적은 회로의 최종 출력에서부터 시작해 후방 추적을 통해 고장 대상 후보를 결정한다. 제안하는 알고리즘은 이러한 임계 경로 추적을 하는 동안에 고장이 발견되지 않은 패턴에서와 고장이 발견된 패턴에서의 논리 값을 비교하여 고장 대상 후보를 줄이는 알고리즘이다. 고장 진단을 하는데 있어서 고장 대상 후보를 줄이는 것은 전체 고장 진단 시간에 있어 가장 큰 부분을 차지한다. 때문에 제안하는 알고리즘은 기존의 후방 추적을 이용한 방식보다 고장 진단 시 매우 빠른 성능을 보인다. 또한 조합회로와 순차회로의 모든 경우에 적용이 가능하다. 본 논문에서는 ISCAS'85회로와 ISCAS'89의 회로들을 가지고 실험을 하여 기존의 고장 진단 방식의 경우보다 얼마나 성능이 좋아졌는지 보이도록 하겠다.

Abstract

In this paper, we present a pattern comparison algorithm for reducing fault candidate lists. The number of fault candidates determines the total fault simulation time. To decrease the total fault diagnosis time, the reduction of the number of fault candidates is essential. Critical path tracing determines fault candidate lists detected by a set of tests using a backtracing algorithm starting at the primary outputs of a circuit. The proposed algorithm reduces fault candidates comparing failing patterns with good patterns during critical path tracing process. As we reduce all fault candidates of the circuit to more accurately suspected fault candidates, we can greatly reduce fault simulation time. The proposed algorithm greatly increases simulation speed than that of a conventional backtracing method. The proposed algorithm is applicable to both combinational and sequential circuits. Experimental results on ISCAS'85 and ISCAS'89 benchmark circuits showed fault candidates are pruned and fault diagnosis time is also decreased in proportion to fault candidate decrease.

Keywords: Diagnosis, Critical path tracing, Fault simulation, Pattern comparison, Candidate

I. 서 론

반도체 설계 및 공정 기술의 발달로 인하여 회로의 복잡도가 나날이 증가하고 있다. 회로가 복잡하게 됨에 따라 칩 내부의 결함과 고장의 종류와 수 또한 증가하게 되었고, 이로 인하여 고장진단의 필요성 또한 증가하고 있다. 고장 진단이란 매우 집적화 되어 있는 칩 내부에 고장이 있다면 어느 종류의 고장이 어디에 위치하는지 찾아내는 과정을 말한다. 정확한 고장 진단은 설계와 공정상의 오류를 보완하는 역할을 한다. 이와 같이 효율적인 고장 진단을 하는 것은 칩의 질을 높이고 칩 생산 가격을 줄이는데 큰 역할을 한다.

고장 진단[1~5]의 방법으로는 고장 진단에 필요한 정보를 미리 생성하여 고장 사전의 형태로 저장해 두었다가 고장 진단이 실행될 때에 그 정보를 이용하는 방식[1, 6]이 있고, 고장 시뮬레이션[5, 7]이나 후방 추적[3]을 이용하여 테스트를 통해 생성되는 고장이 검출된 패턴과 고장이 전파된 출력 단자의 위치와 같은 고장 응답을 가지고 있는 상태에서 고장 진단을 하는 방식이 있다.

고장 시뮬레이션이나 후방 추적을 이용하는 방법을 동적 고장 진단 방식[5]이라고 부른다. 동적 고장 진단 방식을 사용하기 위해서는 실험적으로 생성된 결과 값

* 정회원, ** 평생회원, 연세대학교 전기전자공학과
(Department of Electrical Electronic Engineering, Yonsei University)
접수일자: 2007년7월20일, 수정완료일: 2007년10월24일

을 실제 결과인 고장 응답과 비교하고 분석하여 고장의 종류와 위치를 알아내야 한다. 최근에는 많은 저장 공간을 필요로 하는 정적인 방법보다 동적 고장 진단 방식을 주로 이용한다. 임계 경로 추적[3]이란 이러한 동적 고장 진단 방식 중 고장 시뮬레이션을 대체하기 위해서 제안된 방식이다. 임계 경로 추적은 고장이 발생하여 전달된 최종 출력 단에서 후방 추적을 하는 방식이다. 최종 출력 단에서 임계 경로를 따라 후방 추적을 하기 때문에 고장 진단 시 고장 대상을 대폭 줄여 고장 시뮬레이션 시간을 훨씬 단축시키는 효과를 가져 올 수 있다. 모든 고장 대상에서 정확한 고장 대상으로 축소되기 때문에 고장 진단의 결과도 좋아지게 된다.

하지만 이러한 임계 경로 추적을 통한 방법은 임계 경로에 있는 모든 대상을 고장 대상 후보로 생각하기 때문에 고장 진단의 시간이 많이 소요 되게 된다. 그리하여 본 논문에서는 고장 진단 시간을 더욱 줄이기 위하여 고장이 발견된 패턴과 고장이 발견되지 않은 패턴에서의 논리 값을 비교하여 고장 대상 후보를 줄이는 방법을 제안한다.

Ⅱ. 고장 진단 방식

고장진단을 하기 위해서는 패턴을 통해 고장이 발현된 패턴을 알아야 하고, 그 패턴에서 고장이 나타난 최종 출력이 어딘지 알아야 한다. 처음 이러한 과정을 거친 후에 고장 대상 후보를 정하여 그 고장 대상 후보들에 대하여 고장 시뮬레이션을 하여 고장의 위치와 종류를 찾아내는 것이다. 이렇게 고장 대상 후보를 정확하고 빠르게 찾아내는 것이 고장 진단 전체 과정에서 중요한 부분을 차지하게 된다. 이렇게 고장 대상 후보를 찾는 방법 중에 임계 경로 추적 알고리즘[3]이 가장 많이 이용되는 방식이다.

1. 임계 경로 추적 알고리즘

임계 경로 추적은 고장이 발생하여 전달된 최종 출력 단에서 후방 추적을 하는 방식이다. 임계 경로 추적 기법은 무고장 시뮬레이션을 수행한 후에 주출력단에서부터 주입력단까지 활성화 경로(sensitized path)를 후방 추적하게 된다. 후방 추적 과정에서, 논리 소자의 입력 신호가 활성화 되려면 입력 신호의 논리값을 반대로 취할 때 출력단의 논리 값도 반대가 되어야 한다. 이처럼 최종 출력 단에서부터 시작하여 어떤 특정한 하나의 입력 값을 변화 시켰을 때 출력 값이 변화했다면 고장이 그 입력 값을 통해 출력으로 전달이 될 수 있는 것을 의미하기 때문에 그 입력 값들을 활성화된 입력이라고 칭한다. 후방 추적 과정은 더 이상 활성화되는 신호가 없거나 주입력단에 도달할 때까지 계속된다. 그리고 활성화된 경로상에 존재하는 모든 고장은 검출이 가능하게 된다. 임계 추적 방식은 동시 고장 시뮬레이션 방시에 비하여 스캔 설계된 회로에서 약 60% 속도 향상을 얻을 수 있다[3].

2. 병렬 고장 시뮬레이션

병렬 고장 시뮬레이션은 직렬 고장 시뮬레이션의 확장 기법으로 여러 개의 고장 회로를 동시에 하나의 무고장 회로와 비교하는 방식이다. 하나의 고장 회로를 무고장 회로 하나와 비교하는 방식보다는 동시에 여러 개의 고장 회로를 무고장 회로와 비교하는 방식이 고장 진단 시간을 고려해 볼 때 많은 시간을 단축시킬 수 있다. 본 논문에서는 이러한 병렬 고장 시뮬레이션에 임계 경로 추적을 적용하였다.

만약 고장회로의 수가 무한정으로 늘어나게 된다면 거의 모든 입력들이 활성화 되게 되므로 임계 경로 추적 방식이 쓸모가 없게 되겠지만, 실제로 고장회로의 수를 무한정으로 테스트 해보기에는 많은 시간과 비용이 많이 소요되고, 또 무한정이 아니라면 활성화된 입력들만을 찾아 후방 추적을 하기 때문에 고장 대상이 되는 대상자들의 수는 상당 수 줄어들게 된다. 고장 대상이 줄게 되면 고장 진단이 정확해지고, 고장 시뮬레이션 시간도 많이 단축되게 된다[3]. 본 논문에서는 이와 같이 병렬 고장 시뮬레이션 방법을 이용하여 좀 더 효율적인 고장진단 방식을 하였고, 이와 병행하여 고장의 대상이 되는 후보군에 대하여 고장 시뮬레이션을 할 때 매칭알고리즘을 통한 점수 계산 방식을 이용하였다. 점수 계산을 통하여 최종적인 결과 값에서 점수가 가장 높은 고장 대상 후보를 실제 고장이 있는 후보라고 최종 판단하게 되는 것이다.

3. 매칭 알고리즘(matching algorithm)

매칭 알고리즘(matching algorithm)을 통한 점수 계산 방식[8~10]은 테스트 결과 값과 시뮬레이션 결과 값이 실제 결함(defect) 장소에 고장(fault) 장소가 가까울수록 더 많이 일치한다는 가정에 기초한다. 테스트 결과 값과 시뮬레이션 결과 값이 실제 결함 장소에 고장 장소가 가까울수록 더 많이 일치한다는 매칭 알고리즘의 기본 가정에 따라 기존에 여러 가지 매칭 알고리즘이

제안되었다. 하지만, 기존 방법에 따른 고장 진단은 사용자의 요구를 충족시킬 만큼 정확하지 못했다.

기존의 매칭 알고리즘[8, 9]과 같이 단순히 실제 고장 응답과 시뮬레이션 출력 결과 값에 따라서 일정한 점수를 증가시켜 주기만 한다거나 부분적으로 일정한 점수를 감소시켜 주는 경우를 추가한다고 해서 정확한 고장의 위치를 찾아내기는 힘들다. 이와는 달리 완전 일치 공통부분이라는 새로운 개념을 사용하여 고장 진단의 정확도를 크게 향상시킨 매칭 알고리즘 방식[10]도 있다. 이 방법은 단순히 완전 일치 공통부분의 개수만을 고려하여서 다양한 모든 회로에 적용하기에는 어려움이 따른다. 각각의 회로는 일반적으로 상이한 개수의 출력 단자를 가지며, 출력 단자의 수가 다르다면 완전 일치 공통부분이 나타날 수 있는 가능성 역시 다를 수밖에 없게 되기 때문이다. 하지만, 이러한 기존의 매칭 알고리즘[10]에서는 이러한 회로 각각의 특성에 따른 가능성을 무시한 채 일괄적으로 완전 일치 공통부분이 몇 번 나타나는가를 가지고 고장 진단을 우선적으로 실시한다. 따라서 이러한 방법으로는 다양한 회로에 대해 탄력적으로 정확한 고장 진단을 수행하기가 어렵다.

위와 같은 기존의 매칭 알고리즘의 문제점을 해결하기 위해서 출력 단자 수를 고려한 매칭 알고리즘[2]을 이용하였다. 본 논문에서 사용된 매칭 알고리즘[2]은 완전 일치 공통부분이 생길 경우에 출력 단자의 수를 가중치의 값으로 이용하여 점수를 계산해주고 기존에 제안된 점수를 빼주는 방법을 보다 폭넓게 적용하여 고장 후보의 점수를 계산하도록 하였다. 기존의 매칭 알고리즘에서의 완전 일치 공통부분과 이용한 매칭 알고리즘에서의 완전 일치 공통부분은 해당되는 경우가 약간 다르다. 즉, 사용된 매칭 알고리즘에서는 모든 출력 단에서 고장 응답이 전혀 생성되지 않은 경우에도 완전 일치 공통부분으로 간주하여 점수를 계산해 주게 된다. 즉, 실제 결과 값과 시뮬레이션의 결과 값이 모두 0의 시그니쳐를 가지는 경우도 완전 일치 공통부분으로 적용시켜 주는 것이다. 이 방법은 기존의 방법보다 더욱 정확하게 고장의 위치를 찾아내고 고장의 형태를 결정지어 주었다.

Ⅲ. 제안하는 패턴 비교 알고리즘

본 장에서는 고장 진단 시간을 줄이기 위하여 패턴 비교를 통한 방식을 소개하도록 하겠다. 패턴 비교를 통한 방식은 패턴을 비교하기 위한 무고장 패턴에서의 시뮬레이션 시간이 소요되기는 하지만 알고리즘으로 인하여 줄어드는 고장 대상 후보가 굉장히 크기 때문에 고장 대상 후보를 대상으로 고장 시뮬레이션을 시간을 고려했을 때 전체 고장 진단 시간은 매우 많이 줄어들게 된다. 이와 같이 본 논문에서는 기존의 임계 경로 추적 방식보다 고장 대상 수를 많이 줄여서, 전체 고장 진단 시간을 줄일 수 있는 방식을 소개하도록 하겠다.

1. 패턴 비교 알고리즘

아래 그림 1은 고장 진단을 하는 순서에서 패턴 비교 알고리즘에 언제 쓰이는 가를 나타낸 그림이다. 앞서 이야기 한 것과 같이 본 논문에서는 정확한 고장진단을 위해서 효율적인 매칭 알고리즘을 도입하였다. 그림 1에서 보는 바와 같이 고장 진단은 패턴을 통한 테스트를 수행하여 회로에 고장이 존재한다고 판단하면 우선 무고장 시뮬레이션을 한 후에 임계 경로 추적을 통해 고장 대상 후보를 완성한다.

이렇게 만들어진 고장 대상 후보를 가지고 고장 시뮬레이션을 하여 실제 고장이 어디에 위치하며 종류는 무엇인지 판단할 수 있다. 이와 같이 고장 대상 후보는 곧, 고장 시뮬레이션의 속도와 정확도를 좌우하기 때문에 고장 대상 후보를 최대한 줄이는 것은 고장 진단 전체의 시간을 줄일 수 있고 고장 진단의 정확도 또한 좋아지게 된다. 고장 진단 전체 시간에 가장 많은 비중을 차지하는 고장 시뮬레이션 시간을 줄이기 위하여 고장 대상 후보를 줄이는 알고리즘으로서 고장이 발현된 패턴과 고장이 발현되지 않은 패턴의 논리 값을 비교하여 고장 대상 수를 줄이는 방식을 소개한다.

패턴 비교 알고리즘은 그림 1과 같이 고장 시뮬레이

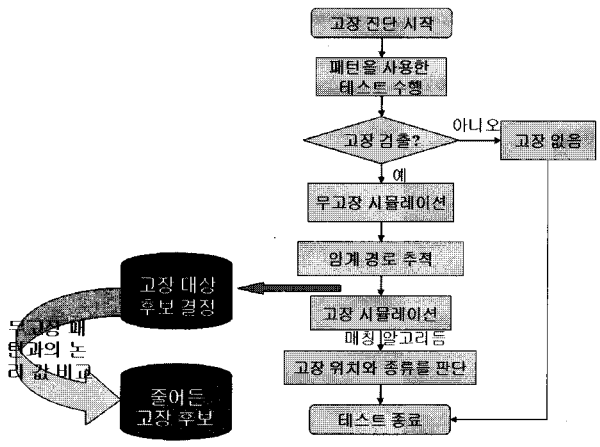


그림 1. 패턴 비교 알고리즘을 이용한 고장 진단 순서
Fig. 1. Order of fault diagnosis using pattern comparison algorithm.

션을 하기 이전에 임계 경로 추적을 할 때, 만들어진 고장 대상 후보에서 고장이 발현된 패턴과 고장이 발현되지 않은 패턴의 논리 값을 비교하여 고장 대상 수 줄이는 방식이다.

다시 말해서, 고장 대상 수를 줄이는 것은 고장 시뮬레이션을 통해 고장 진단을 할 때, 고장 시뮬레이션을 할 대상을 줄이는 것과 같고, 시뮬레이션의 대부분을 차지하는 고장 시뮬레이션 시간을 대폭 줄일 수 있게 되는 것이다.

만약 실제 고장이 있는 고장 후보라면 고장이 발현되지 않은 패턴에서 임계 경로 추적을 한 경로의 논리 값과 고장이 발현된 패턴에서 임계 경로 추적을 한 경로의 논리 값이 같지 않아야 한다. 논리 값이 같다면 고장이 발현되지 않은 무고장 패턴에서 논리 값이 같은 경로에 있는 고장은 고장이 최종 출력까지 전달되었어야 하고 그렇게 되면 시뮬레이션 결과 고장이 발견된 패턴

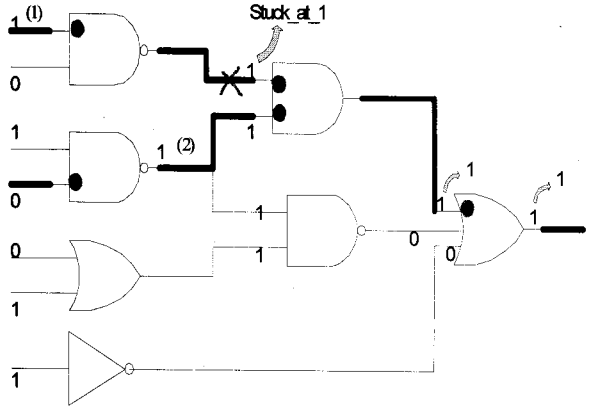


그림 2. 예제(무고장 패턴의 임계 경로추적)
Fig. 2. Example (ciritical path tracing from a good pattern).

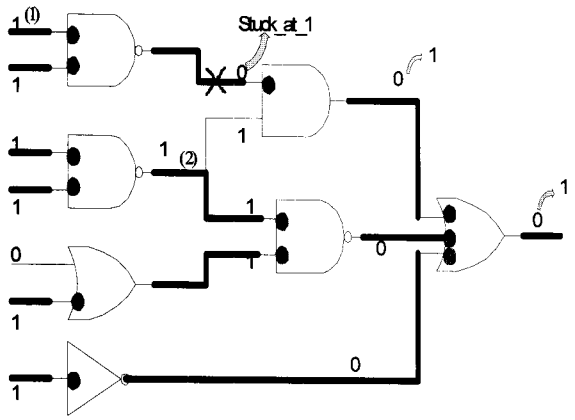


그림 3. 예제(고장 패턴의 임계 경로추적)
Fig. 3. Example(critical path tracing from a failed pattern).

으로 판명이 났어야 한다.

그림 2와 그림 3은 이와 같은 패턴 비교 알고리즘에 대한 설명을 하기 위한 예제이다. 그림 2는 고장이 발견되지 않은 무고장 패턴에서의 임계 경로를 나타낸 것이고, 그림 3은 고장이 발견된 고장 패턴에서 임계 경로를 나타낸 것이다. 그림에서 굵은 선으로 표시된 부분이 최종 출력에서 후방 추적을 하여 결정된 임계 경로가 되겠다. 즉 굵은 색으로 표시된 부분에 고장이 있다면 최종 출력까지 전달이 되어야만 한다. 결론부터 이야기하면 그림 2와 그림 3에서 (1)과 (2)로 표시된 부분은 패턴 비교 알고리즘을 통해 제거 할 수 있는 고장 대상 후보이다. 그림에서 보는 바와 같이 (1)과 (2)는 우선 모두 임계 경로에 존재한다. 또한 논리 값 또한 '1'로 같다. 이러한 이유로 만약에 그림 2에서 (1)이나 (2)에 고장이 존재 한다면 그 고장은 최종 출력까지 전달이 되었어야 한다. 그렇기 때문에 우리는 이러한 고장 대상 후보는 실제 고장이 있을 수 없는 곳으로 판명하고 제거할 수 있게 된다.

2. 패턴 비교 알고리즘의 pseudo 코드

그림 4는 패턴비교 알고리즘을 pseudo 코드로 표현한 것이다. 패턴비교 알고리즘은 고장이 발견된 패턴을 통해 얻은 고장 대상 후보 중에서 고장이 발견되지 않은 패턴을 진행 할 때의 값과의 비교 분석을 통해 불필요한 고장 대상 후보를 제거하여 감소된 고장 대상 후

```
Pattern Comparison Algorithm{
    CPT(failed outputs(FPOs)) in Good patterns;
    CPT(failed outputs(FPOs)) in Failed pattern;
    for every failed_sensitive inputs{
        if (value[i] = value[j] & good_sensitivity)
            eliminate this inputs;
    }
}
CPT(FPOs){
    While (good_sensitivity input ≠ ∅)
            Critical path tracing(FPO) ;
    for every good_sensitivity inputs
            value[i] = logic_value ;
}
CPT= Critical Path Tracing, FPO= Failed Primary Ouput
```

그림 4. 패턴 비교 알고리즘의 pseudo code
Fig. 4. Critical path tracing using the good patterns.

보를 얻는 것이다. 정말 고장이 전달된 후보로 남으려면 고장이 나지 않은 패턴에서 임계 경로를 통해 지나친 임계 입력과는 그 논리 값이 같아선 안 된다. 패턴비교 알고리즘은 이러한 점을 이용하여 불필요한 고장 대상 후보를 제거하는 알고리즘이다. 알고리즘을 진행하기에 앞서 고장이 최종 출력까지 전달된 패턴을 알고 있어야하고, 또한 몇 번째 사이클에서 고장이 발견 되었는지 와, 고장이 난 출력을 미리 알고 있어야 한다. 고장이 발견된 패턴에 따라서 고장이 발견된 출력이 다를 것이므로, 정해진 고장 패턴에 대한 무고장 패턴을 선별하여야 한다.

(1) 우선 첫 번째 고장이 발견된 패턴에서의 고장이 발견된 출력을 저장해 두고, 고장 발견되지 않은 패턴들을 상대로 저장해둔 출력에서 임계 경로 추적을 통하여 임계 경로와 그 논리 값들을 저장해 둔다.

(2) (1)의 경우와 같이 여러 개의 고장이 발견되지 않은 패턴들을 상대로 임계 경로 추적을 통해 임계 경로와 그 논리 값들을 저장한다.

(3) 고장이 발견된 패턴에서 고장이 발견된 출력에서 시작하여 임계 경로 추적을 통해 각각의 입력의 논리 값을 저장해둔다.

(4) 저장해둔 값들을 비교하여 고장이 발견되지 않은 패턴에서의 입력 값에 임계성이 있고, 또한 고장이 발견된 패턴에서의 입력 값에 임계성이 있으며, 그 논리 값이 같은 경우에 그 고장 대상 후보를 후보에서 제거한다.

Ⅳ. 성능평가

본 논문에서는 ISCAS'85와 ISCAS'89의 회로들에 대하여 실험을 하였다. 각각의 회로 하나당 30개의 고장 회로를 대상으로 실험을 하였고 각 고장 회로마다 하나의 고장을 삽입하였다. 실험 결과는 고장 패턴을 제외한 모든 무고장 패턴을 대상으로 비교하여 실험하였다. 정확한 고장 진단을 위해서 MAID라는 매칭알고리즘을 이용한 고장 진단 프로그램을 기존의 방식과 제안하는 방식에 모두 똑같이 이용하였다. 즉, 고장 대상 후보를 정하는 방법에만 차이를 두고 비교 실험하였다. 모든 실험은 매칭 알고리즘을 통하여 1순위에 고장을 정확하게 찾아내었다.

표 1은 실험한 모든 회로에서 패턴 비교 알고리즘을 이용하였을 때의 고장 대상 수와 전체 고장 진단 시간을 비교한 표이다. 제안하는 방식의 고장 대상 수가 기존의 임계 경로 추적 방식보다 훨씬 적은 것을 확인 할 수 있다. 기존의 임계 경로 추적 방식은 임계 경로에 있는 모든 고장 대상 후보를 고장 진단하였고, 제안하는 방식은 그 수를 대폭 줄였기 때문이다. 표 1의 다섯 번째 열의 조합회로에서의 평균을 보면 대략 30%정도 고장 대상 후보가 줄어든 것을 확인 할 수 있다. 이와 같이 구해진 고장 대상 후보를 가지고 같은 고장 진단 프로그램으로 돌렸을 때 전체 고장 진단 시간을 비교한 것이 6~8 열이다. 같은 프로그램으로 실험을 하였기 때문에 그 비율은 정확하다고 할 수 있다. 결론적으로 마지막 열의 비율을 보면 10~30% 정도 줄어든 것을

표 1. 조합 및 순차회로에 대한 평균 고장 후보와 고장진단 시간의 비교
Table 1. Results of candidates and total diagnosis times for combinational & sequential circuits.

회로	전체 고장후보	(a)고장후보 (기존의CPT)	(b)고장후보 (제안방식)	(b)/(a)	(c)전체 진단시간 (기존의 CPT)	(d)전체 진단시간 (제안 방식)	(d) / (c)
c1355	1219	534	396	0.74	0.76	0.61	0.80
c1908	949	315	203	0.65	0.51	0.49	0.95
c2670	1742	213	155	0.73	1.29	1.05	0.82
c3540	2790	671	467	0.70	2.70	2.49	0.92
c5315	3978	259	165	0.64	2.17	1.74	0.80
c6288	8875	3819	2558	0.67	14.61	11.20	0.77
c7552	5054	545	362	0.66	4.58	3.62	0.79
평균				0.68			0.80
s5378	4898	356	211	0.59	14.23	12.70	0.89
s9234	6534	454	255	0.56	27.15	23.03	0.85
s13207	12422	506	340	0.67	51.50	45.82	0.89
s15850	13881	428	206	0.48	46.21	34.60	0.75
s35932	39399	224	118	0.53	21.71	17.54	0.81
평균				0.55			0.84

알 수 있고 평균을 보면 20%정도 전체 고장 진단 시간이 줄어든 것을 확인 할 수 있다.

표 1의 s회로들은 순차회로를 의미한다. 순차 회로 중 큰 회로 6개를 비교하였고 기존의 임계 경로 추적을 통한 방식과 비교했을 때, 조합 회로보다 훨씬 더 크게 고장 대상 후보가 줄어든 것을 확인 할 수 있다. 표에서 보면 알 수 있듯이 평균적으로 45% 정도 고장 대상 후보가 줄어들었다. 순차회로에서도 마찬가지로 구해진 고장 대상 후보를 가지고 고장 진단을 하였을 때 전체 고장 진단의 시간을 구한 것이 6~8열에 나타나있다. 고장대상 수는 45%로 크게 줄어든 반면에 전체 고장진단 시간은 15%정도 줄어든 것을 볼 수 있다. 조합 회로의 실험 결과 는 줄어든 고장 대상 수와 줄어든 전체 고장 진단 시간이 거의 비례한 것을 볼 수 있는 반면에 순차회로의 결과는 시간이 크게 안 줄어든 것을 확인 할 수 있다. 이것은 고장 대상 후보를 줄이기 위하여 무고장 패턴에서의 시뮬레이션 시간이 그만큼 많이 들어갔기 때문이다. 앞서 이야기했듯이 본 논문의 실험에서는 고장 패턴 하나당 고장이 발생하지 않은 나머지 무고장 패턴을 모두 비교하였기 무고장 패턴에서의 임계 경로 추적 시간이 그만큼 길었던 것이다. 의 본 실험에서는 비교한 무고장 패턴의 수를 변화시켜서 실험한 결과를 추가하지는 않았지만, 비교할 무고장 패턴의 수를 정해서 실험을 해서 적절한 비교 수를 정할 수 만 있다면, 순차회로에서의 결과가 훨씬 좋아질 것이다. 예를 들면 고장 패턴 하나당 무고장 패턴 수를 두 개씩 비교한다면 비교하는 시간이 훨씬 줄어들기 때문에 전체 고장 진단 시간을 봤을 때 훨씬 이득이 될 것이다.

이와 같이 패턴비교 알고리즘은 고장이 발견된 패턴 하나당 고장이 발견되지 않은 패턴을 몇 개를 사용하여 비교하는 가에 따라서 성능이 결정되게 된다. 모든 무고장 패턴을 비교하면 고장 대상 후보를 많이 줄일 수 있게 되지만 그만큼 비교하기 위한 시간이 많이 소요되게 되므로 이런 점들을 절충하기 위하여 비교 개수를 제한할 필요가 있다. 또한 순차 회로의 경우에는 무고장 패턴에서의 시뮬레이션 시간도 만만치 않게 소요되기 때문에 비교 개수를 좀 더 줄일 필요가 있다. 그러므로 상황에 따라 적절하게 유동적으로 패턴 비교 알고리즘을 적용할 때에 그만큼 효과를 볼 수 있게 된다.

V. 결 론

본 논문에서는 고장 시뮬레이션과 후방 추적 방식을 이용하여 매칭 알고리듬을 통해 고장 진단하였다. 후방 추적 방식인 임계 경로 추적을 통해 고장 대상 후보를 결정하여 고장 시뮬레이션을 진행하여 최종적인 고장 진단 결과를 볼 수 있다.

실제로 고장 진단 과정에서 고장 시뮬레이션 시간이 전체 고장 진단 시간에 가장 크게 영향을 미치기 때문에 이러한 고장 시뮬레이션 시간을 줄이기 위해서는 고장 대상 후보를 줄이는 작업이 필수적이다. 때문에 본 논문에서는 무고장 패턴과 고장 패턴에서의 임계 입력들의 논리 값을 비교하여 불필요한 고장 대상을 후보에서 제거하여, 최종적으로 고장 진단의 시간을 줄일 수 있다는 것을 보여준다.

본 논문에서의 실험은 고장이 발현된 패턴 하나당 나머지 무고장 패턴 모두를 비교하여 고장 대상 수를 최대한 줄였지만 적절한 수의 무고장 패턴과 비교를 할 수 있다면 더욱 빠른 고장 진단 결과를 얻을 수 있을 것이다. 결론적으로 패턴 비교 알고리즘은 조합회로와 순차회로 모두에 적용 가능하고 비교할 패턴 수를 적절히 조절한다면 최상의 결과를 기대할 수 있다.

참 고 문 헌

[1] S. D. Millman, E. J. McCluskey and J. M. Acken, "Diagnosing CMOS Bridging Faults with Stuck-At-Fault Dictionaries", Proc. of International Test Conference, pp. 860-870, 1990.

[2] Joohwan Lee, Yoseop Lim, Hyungjun Cho, Sungho Kang, "An Efficient matching Algorithm using the Number of Primary Outputs for Fault Diagnosis", Proc. of 2005 SOC Design Conference, pp. 295-298, 2005.

[3] Hyungjun Cho, Joohwan Lee, Yoseop Lim, Sungho Kang, "An Effective fault diagnosis using critical path tracing", Proc. of Korea Test Conference, pp. 8-12, 2005.

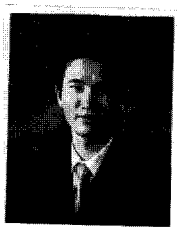
[4] Abramovici, "A Maximal Resolution Guided-Probe Testing Algorithm", Proc. of Design Automation Conference, pp. 189-195, 1989.

[5] Marzouki, J. Laurent and B. Courtois, "Coupling Electron-Beam Probing with Knowledge-Based Fault Localization", Proc. of International Test Conference, pp. 238-247, 1991.

[6] Pomeranz and S. M. Reddy, "On Dictionary-Based Fault Location in Digital Logic Circuits", IEEE Transactions on Computers, pp. 48-59, 1997.

[7] S. Venkataraman, Ismed Hartanto and W.K. Fuchs, "Dynamic Diagnosis of Sequential Circuits Based on Stuck-at Faults", Proc. of VLSI Test Symposium, pp. 198-203, 1996.
[8] Jiang Brandon Liu, "Incremental Fault Diagnosis", IEEE Transactions on Computers, pp. 240-251, 2005.
[9] Boppana V., Fujita M., "Modeling the unknown! Towards model-independent fault and error diagnosis", Proc. of International Test Conference, pp. 1094-1101, 1998.
[10] S. Venkataraman and S. Drummonds, "Poirot : Applications of a Logic Fault Diagnosis Tool", IEEE Design & Test of Computers, pp. 19-30, 2001.

저 자 소 개



조 형 준(정회원)
2005년 연세대학교 전기전자 공학과 학사 졸업.
2007년 연세대학교 전기전자 공학과 석사 졸업.
2007년 현재 연세대학교 전기전자 공학과 박사 과정.
<주관심분야 : Diagnosis, DFT, SoC 테스트>

강 성 호(정회원)
1986년 2월 서울대학교 제어계측 공학과 학사 졸업.
1988년 6월 The University of Texas, Austin 전기 및 컴퓨터공학과 석사 졸업.
1992년 6월 The University of Texas, Austin 전기 및 컴퓨터공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원
1994년 Motorola Inc. 선임 연구원
2007년 현재 연세대학교 전기전자공학과 교수
<주관심분야 : SoC 설계 및 SoC 테스트>