논문 2009-46SD-3-5

고속 고장 진단을 위해 고장 후보 정렬과 테스트 패턴 정렬을 이용한 고장 탈락 방법

(A Fault Dropping Technique with Fault Candidate Ordering and Test Pattern Ordering for Fast Fault Diagnosis)

이 주 환*, 임 요 섭*, 김 홍 식**, 강 성 호***

(Joohwan Lee, Yoseop Lim, Hongsik Kim, and Sungho Kang)

요 약

한 제품을 만들어 시장에 내놓는 데 걸리는 시간이 짧아짐에 따라 고속 고장 진단의 필요성이 커지고 있다. 본 논문에서는 고속 고장 진단을 위하여 정렬된 고장 후보 목록과 정렬된 테스트 패턴을 사용하여 고장 점수를 기준으로 고장 탈락을 시키는 방법을 제안한다. 제안하는 고장 탈락 방법은 고장 시뮬레이션과 매칭 알고리듬을 기반으로 하는 모든 고장 진단에 적용할 수 있다. 완전 주사 ISCAS 89 벤치마크 회로를 이용한 실험 결과는 정렬된 고장 후보 목록 및 정렬된 테스트 패턴을 적용한 고장 탈락 방법의 효율성을 보여준다.

Abstract

In order to reduce time-to-market, the demand for fast fault diagnosis has been increased. In this paper, a fault dropping technique with fault candidate ordering and test pattern ordering for fast fault diagnosis is proposed. Experimental results using the full-scanned ISCAS 89 benchmark circuits show the efficiency of the fault dropping technique with fault candidate ordering and test pattern ordering.

Keywords: Fault dropping, Fault candidate ordering, Test pattern ordering, Fault diagnosis, Fault simulation

I. 서 론

VLSI (very large scale integration) 회로와 디지털 시스템의 복잡도가 나날이 증가하고 크기가 줄어듦에 따라서, 고집적 회로의 고장 분석 (fault analysis)이 점점 더 어려워지고 있다. 따라서 고장 진단 (fault diagnosis)의 중요성이 꾸준히 늘어나고 있으며, 빠르게 변화하는 시장의 요구를 만족시키기 위해서는 짧은 시간 안에 고장 진단이 이루어져야 한다.

고장 진단을 위해 고장 시뮬레이션 (fault simulation)을 기반으로 한 방법이 널리 사용되고 있으며, 이러한 고장 진단 시간의 대부분은 고장 시뮬레이션의 수행을 위해 사용된다. 그러므로 고장 시뮬레이션의 수행 시간을 줄이는 것은 고장을 빠르게 진단할 수 있도록 해준다. 같은 회로를 대상으로 했을 경우에, 고장 시뮬레이션의 수행 시간은 테스트 패턴 (test pattern) 수와 고장 후보 (fault candidate) 수의 곱에 비례하여 결정된다.

고속 고장 진단을 통해 짧은 시간 안에 고장 진단이 수행될 때에도 일정 수준 이상의 정확한 고장 진단 결과가 보장되어야 한다. 고장 시뮬레이션을 이용하여 정확한 고장 진단을 수행하기 위해서는 적절한 매칭 알고리듬 (matching algorithm)[1]을 이용하여 각 고장 후보들에 대해 동일한 횟수의 비교가 이루어져야 한다. 왜냐하면 해당 고장에 대해 매칭 알고리듬을 적용하여 고

* 학생회원, ** 정회원, *** 평생회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)
접수일자: 2008년12월16일, 수정완료일:2009년2월20일

장 점수를 결정하고 이를 바탕으로 실제 고장일 확률이 높은 고장 후보 순위를 결정하기 때문이다. 그리고 무엇보다도 동일한 횟수의 비교를 통해 올바른 비교 결과를 위한 공정성을 확보할 수 있다. 또한 동 순위 고장 후보의 개수를 줄이기 위해서는 고장을 전파하는 테스트 패턴뿐만이 아니라 고장을 전파하지 않는 나머지 테스트 패턴의 정보를 이용하여 고장 진단의 고장 분해도 (fault resolution)를 향상 시키는 것이 필요하다. 따라서 테스트 패턴의 수를 임의로 변경하는 것은 고장 진단 결과에 좋지 않은 영향을 미치기 때문에 고장 시뮬레이션의 실행 시간을 변화시키기 위한 시도에서 제외한다.

고장 후보 수를 줄이기 위해서 다양한[2-3] 방법이 제안되었다. 임계 경로 추적 방법[2]은 고장이 전파된 주출력단과 고장을 전파한 테스트 패턴을 이용하여 고장 시뮬레이션을 수행하기 전에 고장 후보의 수를 줄였으며, 다른 방법[3]은 고장 시뮬레이션을 수행하는 과정 중에 고장 후보의 수를 줄였다. 하지만, 이[3]와 같은 방법은 테스트 대상 회로 내의 고장 유무를 판별할 수는 있지만, 고장의 위치와 종류를 밝혀내는 고장 진단을 수행할 수는 없다. 따라서 고장 진단이 가능하도록 고장 후보 수를 줄여주는 방법으로 고장 탈락 (fault dropping) 방법[4]이 제안되었다.

고장을 정렬하여 사용하는 것을 통해 고장 시뮬레이션의 성능을 향상시킨 방법[5]이 연구되었다. 본 논문에서는 원활한 고장 진단 성능의 비교를 위해서 병렬 고장 시뮬레이션을 이용한다. 이에 따라 적은 시간 부담을 가지며 큰 효과를 나타내는 깊이 우선 탐색 방식을 응용한 고장 후보 정렬 방법을 사용한다.

사용하는 고장 탈락 방법은 고장 시뮬레이션과 매칭 알고리듬을 수행하는 과정 중에 일어난다. 테스트 대상 회로의 고장 시뮬레이션을 수행하기 위해서는 테스트 패턴이 필요하며, 테스트 패턴의 특성을 이용하여 적절한 형태로 정렬한 후에 사용[6]하면 보다 향상된 고장 진단 결과를 얻을 수 있다. 따라서 고장 탈락 방법에 적합한 방식으로 정렬된 최대의 정보를 가지는 테스트 패턴을 우선적으로 사용한다.

고장 시뮬레이션을 사용하는 다수의 고장 진단 방법[7-9]이 제안되었다. 고장 후보 정렬 및 테스트 패턴 정렬을 적용한 고장 탈락 방법은 고장 시뮬레이션과 매칭 알고리듬에 기반을 둔 어떠한 고장 진단에도 사용 가능하다. 그러므로 고장 후보 정렬 및 테스트 패턴 정렬 방법은 기존 고장 진단의 성능을 쉽게 향상시킬 수 있다.

본 논문에서는 고속 고장 진단을 위해 고장 후보 정렬과 테스트 패턴 정렬을 이용한 고장 탈락 방법을 제안한다. 제안하는 고장 탈락 방법은 매칭 알고리듬을 사용하여 고장 시뮬레이션 수행 중에 고장을 탈락시켜 고장 진단의 수행 시간을 줄인다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 고장 시뮬레이션에 기반을 둔 일반적인 고장 진단 방법과 고장 점수 결정을 위한 매칭 알고리듬 및 고장 점수를 이용한 고장 탈락 방법에 대해 기술한다. 또한, 고속 고장 진단을 위해 고장 후보 정렬과 테스트 패턴 정렬을 이용하는 방법을 제안한다. Ⅲ장에서는 제안하는 방법의 효율성을 실험을 통해 제시한다. 그리고 Ⅳ장에서 결론을 맺는다.

Ⅱ. 본 론

1. 고장 시뮬레이션 기반의 고장 진단

고장 진단의 목적은 회로에 존재하는 고장의 위치와 종류를 밝혀내는 것이다. 고장 시뮬레이션을 사용하는 고장 진단의 일반적인 진행 과정은 다음과 같다.

고장 진단은 대상 회로, 테스트 패턴 , 고장 기록을 필수 요소로 준비하고 가능하다면 고장 리스트도 준비하여 시작한다. 고장 시뮬레이션할 고장 후보 수를 줄이기 위해 회로 자체의 구조 정보를 이용한 등가 고장 중첩 (equivalent fault collapsing) 및 고장 기록과 고장을 전파하는 테스트 패턴을 이용한 임계 경로 추적 (critical path tracing) 방법을 사용한다. 추려진 고장 후보 중에서 고장 목표 (target-fault)를 선택하여 고장 시뮬레이션을 수행하며, 적절한 매칭 알고리듬을 이용해 해당 고장에 대한 고장 점수를 결정한다. 일련의 과정을 거친 고장은 고장 리스트에서 삭제되며, 고장 리스트에 고장 후보가 더 이상 존재하지 않을 때까지 고장 시뮬레이션과 매칭 알고리듬 적용의 과정을 반복한다. 모든 고장에 대해 고장 점수가 결정되면 고장 점수를 기준으로 내림차순 정렬을 실시한다. 정렬된 결과는 고장일 확률이 높은 순서와 같으며, 이것이 고장 진단의 출력 결과 값이 된다.

고장 시뮬레이션을 사용하는 고장 진단 과정의 수행 시간은 고장 시뮬레이션의 계산량에 의해 결정된다. 한 번에 하나의 고장을 시뮬레이션할 때의 계산량은 테스트 대상 회로의 크기, 테스트 패턴의 수 그리고 고장 후보의 수에 대략적으로 비례한다. 고장 후보의 수는 개

략적으로 테스트 대상 회로의 크기에 비례하기 때문에, p 개의 테스트 패턴과 n 개의 게이트로 이루어진 회로의 고장 시뮬레이션에 대한 전체 시간 복잡도는 $O(pn^2)$로 나타낼 수 있다. 그러므로 테스트 대상 회로의 크기가 커질수록 고장 진단을 수행하는 시간은 비약적으로 길어지게 되며, 이를 줄일 수 있는 방법이 필요하다.

2. 고장 점수 결정을 위한 매칭 알고리듬

매칭 알고리듬은 고장 진단의 결과에 직접적인 영향을 주기 때문에 적절한 매칭 알고리듬의 사용은 매우 중요하다. 고장 진단을 위한 최적의 매칭 알고리듬[1]을 찾기 위해 가산 계산 (positive calculation), 감산 계산 (negative calculation), 잠재적으로 검출되는 출력 (potentially detected output), 완전 일치 출력 (perfect match), 비교 기준 선정 (criterion output) 그리고 가중치 적용 (weight change)과 같은 필수적인 요소들을 고려하였다. 이러한 결과를 바탕으로 최적의 매칭 알고리듬을 결정하였으며, 그림 1에 고장 진단에 사용하는 점수 계산을 위한 매칭 알고리듬을 나타내었다.

3. 고장 점수를 이용한 고장 탈락 방법

테스트는 고장의 유무만을 판단하기에, 고장이 검출되면 그 즉시 고장 탈락이 일어난다. 하지만, 고장 진단은 고장의 유무만을 판별하는 것이 아니라 고장의 위치와 종류를 밝혀내야 하기 때문에, 고장을 한 번 검출했다고 해서 테스트 과정에서와 같이 고장을 탈락시킬 수 없다.

고속 고장 진단을 위해 제안된 고장 탈락 방법[4]은 고장 시뮬레이션과 매칭 알고리듬을 교대로 이용하며 누적되는 고장 점수의 특징을 이용하여 고장을 탈락시킨다. 즉, 고장 점수가 현저히 낮은 고장 후보를 실제 고장일 확률이 낮다고 가정하고 고장 진단 과정 중에 탈락시켜 고장 진단을 빠르게 수행한다.

고속 고장 진단을 위해 제안된 고장 탈락 방법[4]은 병렬 고장 시뮬레이션을 사용하였다. 고장 탈락 방법을 적용하기 위해서 특별히 병렬 고장 시뮬레이션을 사용해야 하는 것은 아니며, 고장 시뮬레이션 및 매칭 알고리듬을 기반으로 하는 고장 진단이라면 고장 시뮬레이션의 방법에 상관없이 고장 탈락 방법을 적용할 수 있다. 비교 실험 및 개념 정립의 필요성 때문에 본 논문에서도 병렬 고장 시뮬레이션을 사용하도록 하겠다. 고장 탈락 기법을 통해 고속 고장 진단을 실시할 수 있으며,

```
if (the observed signature is a faulty signature)
  if (the candidate signature is a faulty signature)
    if (the observed outputs are equal to the candidate outputs)
      /* perfect match */
      score[current fault] += number of outputs
    else if (the observed outputs are not equal to the candidate outputs)
      /* positive & negative match */
      score[current fault] += number of simultaneously erroneous outputs
      score[current fault] -= number of separately erroneous outputs
  else if (the candidate signature is not a faulty signature)
    /* negative 1 match */
    score[current fault] -= number of observed erroneous outputs
else if (the observed signature is not a faulty signature)
  if (the candidate signature is a faulty signature)
    /* negative 2 match */
    score[current fault] -= number of candidate erroneous outputs
```

그림 1. 고장 점수 계산을 위한 매칭 알고리듬의 의사 코드

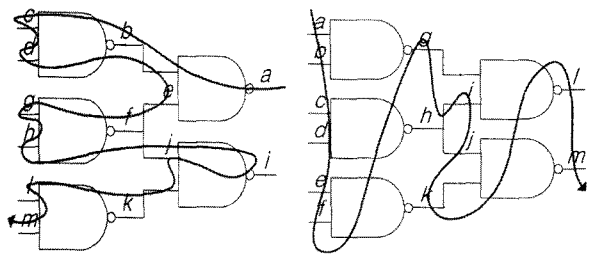
Fig. 1. The pseudo code for fault score calculation of matching algorithm.

제안하는 정렬된 고장 후보 목록의 사용 및 정렬된 테스트 패턴을 사용함으로써 보다 고속의 고장 진단을 구현할 수 있다.

4. 고속 고장 진단을 위해 제안하는 고장 탈락 방법

가. 정렬된 고장 후보 목록의 사용

고속 고장 진단을 위한 고장 탈락 방법은 동시에 수행하는 모든 고장 후보에 대해서 고장 탈락이 일어나야 고장 진단 과정이 중단된다. 하나의 고장 후보라도 고장 탈락이 되지 않고 남아있다면, 고장 시뮬레이션 및 매칭 알고리듬의 적용이 계속되어야 한다. 따라서 동시에 수행되는 모든 고장 후보가 빠르게 고장 탈락될수록 고장 진단 시간을 단축시킬 수 있다.



(a) (b)

그림 2. 개략적인 고장 후보 선택 순서 (a) 깊이 우선 탐색 응용, (b) 넓이 우선 탐색 응용

Fig. 2. The order of selecting fault candidate (a) The application of depth-first search, (b) The application of breadth-first search.

하지만, 각 고장 후보 별 고장 탈락 시기는 각기 다르다. 고장 점수가 매우 낮은 음의 값을 가져야만 고장 탈락 조건을 만족시키는데, 이 때 필요한 테스트 패턴의 수는 같지 않다. 따라서 고속 고장 진단을 위해서는 되도록 비슷한 성향을 보이는 고장 후보들을 함께 묶어서 고장 시뮬레이션 및 매칭 알고리듬을 수행하는 것이 효과적이며, 고장 진단의 조기 중단이 가능할 것이다.

즉, 고장 후보를 적절한 규칙으로 정렬시켜 사용하는 것이 고장 진단을 더욱 빠르게 할 수 있다.

실제 결함과 고장 시뮬레이션에서 고장 영향은 주입력단에서 주출력단으로 전파된다. 또한, 하나의 고장 영향의 결과와 해당 고장이 전파되는 경로 상에 존재하는 다른 고장 후보들에 의한 고장 영향 결과는 유사한 경향을 보인다. 그러므로 고속 고장 진단을 위한 고장 후보 묶음의 선택은 임계 경로 추적 방식에서처럼 깊이 우선 탐색 방법을 따르는 것이 효과적일 것이다.

그러므로 모든 고장 후보가 가능한 같은 시기에 고장 탈락을 일으켜 빠른 고장 진단을 수행할 수 있도록 하기 위해서 그림 2의 (a)와 같이 깊이 우선 탐색 방법을 응용하여 정렬한 고장 후보 목록을 적용한다. 그리고 깊이 우선 탐색 방법의 효율성을 증명하기 위해 그림 2의 (b)와 같이 같은 레벨 내의 고장을 우선적으로 선택하는 넓이 우선 탐색 방법을 응용하여 대조군으로 사용한다. 깊이 우선 탐색 방법과 넓이 우선 탐색 방법의 효율성은 실험 결과에서 확인해 보도록 한다.

나. 정렬된 테스트 패턴의 사용

고장 탈락은 주어진 테스트 패턴을 순서대로 이용하여 고장 시뮬레이션 및 매칭 알고리듬을 수행하며 일어난다. 각각의 테스트 패턴은 검출할 수 있는 고장 후보가 다르기 때문에, 고장 탈락을 적용한 고장 진단의 성능은 적절한 방식으로 정렬된 테스트 패턴을 사용하는 것에 의해서 향상되어질 수 있다.

테스트 패턴의 정렬을 위해 다음의 세 가지 측정 기준을 고려하였다.

- 테스트 패턴 검출력 (Pattern ability : PA)
  PA는 테스트 패턴에 의해서 검출할 수 있는 후보 고장의 수를 나타낸다.
- 고장 빈도 (Fault frequency : FF)
  FF는 해당 고장 후보를 검출하는 테스트 패턴의 수를 나타낸다.
- 고장 빈도의 합 (FF-sum)
  FF-sum은 FF 값의 합을 나타낸다. 단, 테스트 패턴이 검출할 수 있는 고장 후보의 FF 값들의 합으로만 FF-sum 값을 계산한다.

고속 고장 진단을 위해서 중간값 (mid-point)의 PA 값을 가지는 테스트 패턴을 우선적으로 사용하는 테스트 패턴 정렬 알고리듬을 제안한다. 고장의 검출 유무가 고장 진단을 위한 정보의 한 종류로 고려되고 그 정보의 양이 충분하다면, 고장 진단의 성능은 더욱 향상될 것이다. 높은 PA 값을 가지는 테스트 패턴은 많은 수의 고장 후보를 검출할 수 있고, 낮은 PA 값을 가지는 테스트 패턴은 적은 수의 고장 후보를 검출할 수 있다. 그러나 높은 PA 값을 가지고 있으며, 다수의 고장을 검출하는 테스트 패턴이 가장 많은 정보를 가지고 있는 것은 아니다. 왜냐하면, 모든 고장을 검출하는 테스트 패턴을 사용하였을 경우엔 실제 고장이 아닐 확률

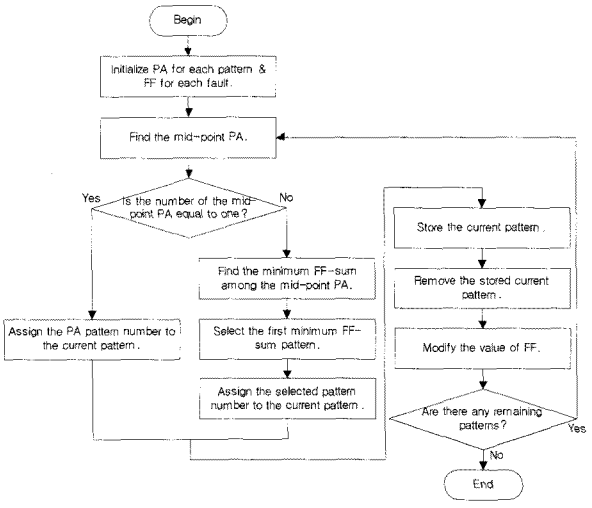


그림 3. 테스트 패턴 정렬 알고리듬

Fig. 3. The algorithm for ordering test pattern.

이 높은 고장 후보를 확실하게 구분해 낼 수 없기 때문이다. 반면에 중간값의 PA 값을 가지는 테스트 패턴을 이용하면 절반의 고장 후보에 대해서는 계속 고장 진단을 수행하고 나머지 절반의 고장 후보에 대해서는 고장 탈락을 일으키도록 결정을 내리기가 용이하다. 그러므로 고장 탈락을 신속히 수행하기 위해서는, 가장 높은 PA 값을 가지는 테스트 패턴 보다는 중간값의 PA 값을 가지는 테스트 패턴을 우선적으로 사용해야 한다.

동일한 PA 값을 가지는 테스트 패턴이 있다면, 어떤 테스트 패턴을 우선적으로 선택해 사용해야 할지를 결정해야 한다. 이런 경우에, 최소의 FF-sum 값을 가지는 테스트 패턴을 사용하는 방법을 제안한다. 낮은 FF-sum 값을 가지는 테스트 패턴은 비교적 드물게 검출되는 고장 후보를 검출하는 것을 의미하기 때문에, 다른 경우보다 높은 고장 검출율을 짧은 시간 안에 얻을 수 있다.

테스트 패턴 정렬 알고리듬을 그림 3에 블록 다이어그램으로 표현하였다. 알고리듬을 시작하기에 앞서 각 테스트 패턴에 대한 PA 값과 각 고장 후보에 대한 FF 값을 계산하여 초기화한다. 그 다음에 계산된 PA 값 중에서, 중간값의 PA 값을 찾는다. 만약 중간값의 PA 값을 가지는 테스트 패턴의 수가 하나라면, 해당 테스트 패턴을 선택한다. 반면에, 중간값의 PA 값을 가지는 테스트 패턴의 수가 하나보다 많다면, 최소의 FF-sum 값을 가지는 테스트 패턴을 우선적으로 선택한다. 선택된 테스트 패턴은 삭제되기 전에 저장되고, PA 값과 FF 값은 새롭게 갱신된다. 이와 같은 과정을 남은 테스트

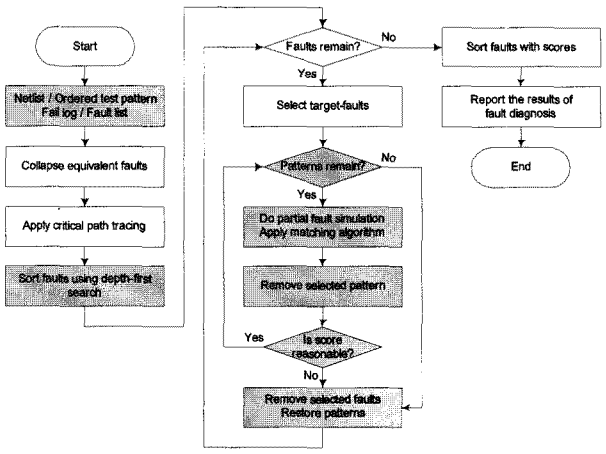


그림 4. 제안하는 고장 탈락 방법을 이용한 고장 진단 과정

Fig. 4. The procedure of fault diagnosis based on fault simulation.

패턴이 없을 때까지 반복한다. 고장 탈락을 이용한 고장 진단은 더 나은 성능을 위하여 정렬된 테스트 패턴을 적용하여 수행될 수 있다.

고장 탈락 방법에 제안하는 정렬된 고장 후보 목록 및 정렬된 테스트 패턴을 적용한 고속 고장 진단 과정을 그림 4에 나타내었다. 그림 4의 음영으로 나타낸 곳이 기존 고장 진단 과정과 다른 차이점이 있는 부분이다. 테스트 패턴은 중간값을 이용한 방법으로 미리 정렬한 후 준비하고 고장 후보 목록은 고장 시뮬레이션을 수행하기 전에 정렬시켜야 한다. 그리고 고장 탈락 방법을 사용하여 고장 시뮬레이션 및 매칭 알고리듬을 적용한다.

III. 실 험

본 논문에서 제안한 고속 고장 진단을 위해 고장 후보 정렬과 테스트 패턴 정렬을 적용한 고장 탈락 방법의 성능을 검증하기 위해 여러 가지 경우에 대한 고장 진단 결과를 비교하였다. 완전 주사 (full scan) 형태로 합성한 ISCAS 89 벤치마크를 실험 회로로 사용하였고, 테스트 패턴은 TetraMax[10]를 이용하여 생성했다. 결함이 있는 회로를 설정하기 위하여, 각각 1개에서 3개의 고장을 무작위로 선정한 후 실험 회로에 삽입하여 각 고장의 개수 별 대상 회로 당 30개씩의 고장 회로를 생성했다. 고장 기록은 주어진 테스트 패턴으로 고장 회로를 고장 시뮬레이션한 고장 응답과 정상 회로의 결과를 비교하여 생성했다. 제안된 기법의 성능 평가를 위해 기존 논문[1]과 비교하는 방식을 채택하였다. 기존 논문[1]은 고장 진단을 위해 병렬 고장 진단 시뮬레이션을 사용하였으며, 본 논문에서도 병렬 고장 시뮬레이션과 매칭알고리듬을 이용하여 고장 진단을 수행했다. 모든 실험은 SUN Fire 880 서버에서 수행했다.

고장 진단의 결과는 FHR (first hit rank)과 CPU (CPU time) 값으로 표 1과 표 2에 정리하였다. FHR은 실제 고장을 찾을 때까지 조사해야 하는 고장 후보 수의 평균값을 그리고 CPU는 전체 고장 진단을 수행할 때 걸린 총 시간의 평균값을 나타낸다. 표 1과 표 2에서 1개에서 3개까지의 서로 다른 고장 개수가 삽입된 세 가지 경우를 각각 10개의 회로에 대하여 고려하였다. 표 1에서는 고장의 개수와 회로별로 기존 논문[1]의 결과 값과 넓이 우선 정렬 방식을 응용하여 고장 탈락을 수행한 방법 및 깊이 우선 정렬 방식을 응용하여 고장

표 1. 정렬된 고장 후보 목록을 사용하여 고장 탈락 방법을 적용한 고장 진단 결과
Table 1. The result of fault diagnosis using fault dropping technique with fault candidate ordering.

Circuit	1 fault						2 faults						3 faults					
	[1]		BF		DF		[1]		BF		DF		[1]		BF		DF	
	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)
s1196	1.00	0.98	1.00	0.88	1.00	0.73	1.03	1.57	1.03	1.13	1.03	0.96	1.03	2.05	1.27	1.23	1.20	1.05
s1238	1.00	0.94	1.00	0.88	1.00	0.76	1.00	1.58	1.03	1.25	1.00	1.02	1.00	2.07	1.30	1.42	3.23	1.19
s1488	1.00	1.32	1.00	0.99	1.00	0.82	1.03	1.92	2.23	1.30	1.07	1.05	1.03	2.36	5.77	1.27	5.33	1.06
s1494	1.00	1.48	1.00	1.13	1.00	0.91	1.00	2.12	2.43	1.36	1.37	1.08	1.00	2.68	3.80	1.46	2.63	1.19
s5378	1.00	5.54	1.00	3.87	1.00	3.44	1.00	7.89	1.53	5.30	1.17	4.63	1.00	11.94	1.43	6.95	1.23	5.83
s9234	1.00	8.55	1.00	5.94	1.00	5.51	1.00	18.39	1.07	8.40	1.03	7.55	1.23	26.85	1.77	9.73	1.87	8.70
s13207	1.00	22.35	1.00	13.18	1.00	12.46	1.17	31.95	1.37	15.19	2.93	13.36	1.00	47.59	1.53	20.58	3.43	18.14
s15850	1.00	14.84	1.00	10.78	1.00	10.13	1.30	44.41	1.33	20.07	1.40	19.06	1.30	53.76	1.40	22.53	1.37	20.45
s35932	1.00	10.36	1.00	9.45	1.00	9.03	1.00	12.89	1.17	12.07	1.00	11.00	1.00	16.39	2.37	13.24	4.97	12.09
s38584	1.00	81.01	1.00	60.97	1.00	54.46	1.00	168.10	3.10	75.93	2.30	68.72	1.00	227.05	2.27	85.36	2.40	78.31

표 2. 세 가지 테스트 패턴 정렬 방식을 사용하여 고장 탈락 방법을 적용한 고장 진단 결과
Table 2. The result of fault diagnosis using fault dropping technique with test pattern ordering.

Circuit	1 fault						2 faults						3 faults					
	max		mid		min		max		mid		min		max		mid		min	
	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)	FHR	CPU(s)
s1196	1.00	0.75	1.00	0.73	1.00	0.73	1.27	1.00	1.03	0.92	2.37	0.91	1.50	1.15	1.43	1.07	2.63	1.04
s1238	1.00	0.76	1.00	0.76	1.00	0.80	1.00	1.00	1.00	1.07	2.87	1.09	1.17	1.18	1.10	1.27	3.53	1.22
s1488	1.00	0.82	1.00	0.85	1.00	0.83	1.07	1.04	1.17	1.10	1.03	0.98	1.30	1.13	1.30	1.16	2.43	0.99
s1494	1.00	0.87	1.00	0.91	1.00	0.89	1.37	1.12	1.40	1.15	1.00	1.11	6.57	1.24	2.70	1.23	1.23	1.24
s5378	1.00	3.56	1.00	3.65	1.00	3.57	1.23	4.42	1.20	4.68	3.63	4.44	2.90	5.72	1.13	6.04	3.67	5.53
s9234	1.00	5.54	1.00	5.51	1.00	5.50	1.00	7.53	1.03	7.50	1.03	7.65	1.27	9.30	1.87	8.67	1.87	9.06
s13207	1.00	12.45	1.00	12.59	1.00	12.85	1.03	15.13	1.03	14.32	3.10	13.24	1.03	19.80	1.87	17.49	3.40	17.07
s15850	1.00	9.78	1.00	9.36	1.00	9.50	1.33	18.67	1.40	18.03	1.33	20.90	1.37	19.49	1.33	18.48	1.40	19.30
s35932	1.00	9.19	1.00	9.06	1.00	8.87	1.13	11.09	1.17	11.00	1.33	10.74	1.17	12.84	2.67	12.50	3.13	11.98
s38584	1.00	54.46	1.00	53.18	1.00	54.96	1.33	67.21	2.43	65.38	2.33	69.26	2.07	80.70	2.37	76.67	1.63	78.63

탈락을 수행한 방법을 비교하였다. 각각의 방법은 [1], BF 및 DF로 나타내었다. 표 2에서는 깊이 우선 탐색 방법을 응용한 고장 탈락 방법을 기반으로 테스트 패턴의 정렬 방식을 다르게 적용하여 고장 진단한 결과를 보였다. PA 값이 가장 큰 테스트 패턴을 우선적으로 사용한 방법을 max로 나타내었으며, PA 값이 가장 작은 테스트 패턴을 먼저 적용한 방법을 min으로 표시하였다. 그리고 중간값의 PA를 사용하는 제안하는 테스트 패턴 정렬 방법을 mid로 나타내었다.

표 1과 표 2를 바탕으로 고장 탈락 방법, 정렬된 고장 후보 목록 및 정렬된 테스트 패턴의 사용의 성능을 평가하겠다.

1. 고장 탈락 방법의 성능 평가

고장 탈락 방법 자체의 성능을 평가하기 위해서 고장 탈락 방법을 적용하지 않은 고장 진단 시간 결과와 고장 탈락 방법만을 적용한 고장 진단 시간 결과를 비교하였다. 두 가지 방법 모두 응용된 넓이 우선 탐색 방법을 적용하여 고장 진단을 수행했으며, 테스트 패턴은 TetraMax에서 생성된 패턴을 그대로 사용하였다.

비교 논문 [1]에서와 같이 고장 탈락을 적용하였을 경우에 많게는 64% 그리고 적게는 6%의 시간 절감 효과를 얻을 수 있다. 그리고 고장 진단 시간을 평균 36% 단축시킬 수 있다.

2. 정렬된 고장 후보 목록을 사용한 고장 탈락 방법의 성능 평가

깊이 우선 탐색 방식과 넓이 우선 탐색 방식을 응용하여 적용한 고장 탈락 방법의 비교 결과를 그림 5에 나타내었다. 각각의 정렬된 고장 후보 목록을 고장 탈락 방법에 적용하였으며 테스트 패턴은 TetraMax에서 생성된 패턴을 수정 없이 사용하였다.

그림 5에서 비교 기준 값은 넓이 우선 탐색 방식이며, 고장 진단의 시간 결과 비교 값이 1보다 작을수록 깊이 우선 탐색 방법이 고장 진단 시간 단축에 효과적인 것을 의미한다. 깊이 우선 탐색 방식을 응용하여 고장 탈락을 적용한 고장 진단은 넓이 우선 탐색 방식을 응용하여 적용한 경우보다 크게는 24% 적게는 4%의 시간 절감 효과를 얻을 수 있다. 그리고 고장 진단 시간을 평균 12% 단축시킬 수 있다.

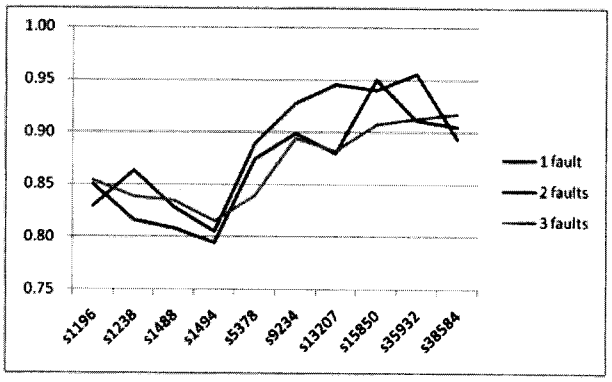


그림 5. 두 가지 고장 후보 정렬 방식을 사용하여 고장 탈락 방법을 적용한 고장 진단 시간 결과 비교

Fig. 5. The CPU time ratio of the case of depth-first search : the case of breadth-first search (DF/BF).

3. 정렬된 테스트 패턴을 사용한 고장 탈락 방법의 성능 평가

표 2에 서로 다른 방식으로 정렬한 테스트 패턴을 사용한 고장 진단 결과를 나타냈다. 각 방식을 비교해 보면, 중간값의 PA 값을 가지는 테스트 패턴을 우선적으로 사용하는 것이 가장 우수한 성능을 보인다.

중간값의 PA 값을 가지는 테스트 패턴이 고장 진단 시간 결과에 주는 영향을 그림 6에 나타내었다. 비교의 기준이 되는 고장 진단 시간 결과 값은 TetraMax로 생성한 테스트 패턴을 이용했으며, 비교 값이 1보다 작을수록 고장 진단 시간이 단축되는 것을 의미한다. 중간값을 이용한 테스트 패턴 정렬 방법은 테스트 패턴 정렬 방법을 사용하지 않았을 경우보다 10%에서 -10%의 고장 진단 시간 단축 효과를 나타내며, 평균적으로 1%

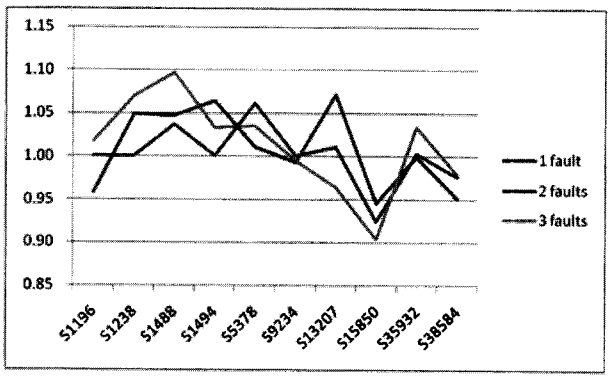


그림 6. 중간값을 이용해 정렬한 테스트 패턴을 사용한 고장 진단 시간 결과 비교

Fig. 6. The CPU time ratio of the case of mid-point : the case of no test pattern ordering (mid/DF).

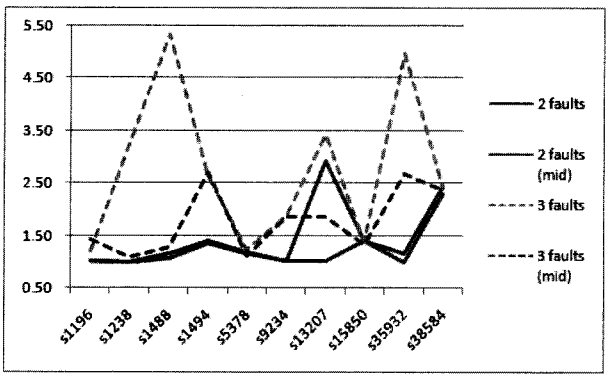


그림 7. 중간값을 이용해 정렬한 테스트 패턴을 사용한 고장 진단 정확도 결과 비교

Fig. 7. The comparison of accuracy for fault diagnosis using test pattern ordering (mid).

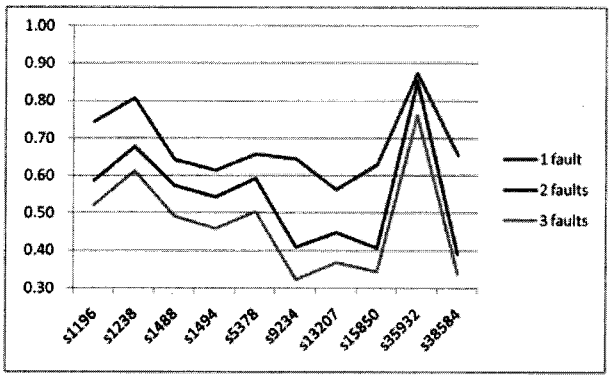


그림 8. 깊이 우선 탐색을 응용하여 정렬한 고장 후보 목록과 중간값을 이용하여 정렬한 테스트 패턴을 사용하여 고장 탈락 방법을 적용한 고장 진단 시간 결과 비교

Fig. 8. The CPU time ratio of the proposed case : the previous case (mid/[1]).

증가하는 결과를 보인다. 하지만, 회로의 크기가 커질수록 시간 단축의 효과가 점점 향상되는 것을 알 수 있다.

또한, 중간값을 이용해 정렬한 테스트 패턴을 이용할 경우에 그림 7과 같이 고장 진단 정확도가 향상되는 것을 확인할 수 있다. 그림 7의 가로축은 실험 회로를 나타내고 세로축은 고장 진단의 정확도인 FHR 값을 의미한다. 하나의 고장이 삽입된 경우는 그림 7에 표시하지 않았다. 왜냐하면 중간값을 이용해서 테스트 패턴을 정렬해서 사용한 경우(mid)와 그렇지 않은 경우 모두 FHR 1인 정확한 고장 진단을 수행하기 때문이다. 하지만 삽입된 고장의 수가 두 개인 경우와 세 개인 경우에는 중간값을 이용하여 테스트 패턴을 정렬하였을 때 그렇지 않았던 때보다 각각 10%, 36%의 고장 진단 성능의 향상을 나타낸다. 따라서 고장 탈락을 위해 정렬된 테스트 패턴을 사용하는 것은 고장 진단 시간의 단축

효과는 물론 고장 진단의 정확도를 높여준다.

제안하는 고장 후보 정렬 방식과 테스트 패턴 정렬 방법을 모두 적용한 고장 탈락을 사용하는 고속 고장 진단 방법의 시간 결과를 그림 8에서 비교하였다. 제안하는 고속 고장 진단은 기존의 방법과 비교해서 최대 68%에서 최소 13%의 시간 절감 효과를 나타내었으며, 평균 43%의 고장 진단 시간을 단축시킨다.

IV. 결 론

본 논문에서는 고속 고장 진단을 위하여 깊이 우선 방식을 이용하여 정렬한 고장 후보 목록과 중간값의 PA 값을 가지는 테스트 패턴을 우선적으로 사용하여 고장 점수를 기준으로 고장 탈락을 시키는 방법을 제안하였다. 고장 탈락 방법은 전체 고장 진단의 수행 시간을 평균 36% 줄여 주며, 정렬된 고장 후보 목록과 정렬된 테스트 패턴을 사용함으로써 고장 진단 성능을 보다 향상시킬 수 있다. 정렬된 고장 후보 목록과 정렬된 테스트 패턴을 추가적으로 사용함에 따라, 수행 시간이 평균 12%정도 더 향상될 뿐만 아니라 고장 진단의 정확도가 평균 23% 증가한다. 테스트 대상 회로의 크기가 증가할수록 고장 탈락에 의한 단축 효과가 크게 나타남에 따라 고장 진단의 수행 시간이 줄어드는 것을 확인할 수 있다. 또한, 회로에 존재하는 고장의 수가 많을수록, 제안하는 고속 고장 진단 방법이 효율적임을 확인할 수 있다. 제안하는 정렬된 고장 후보 목록과 정렬된 테스트 패턴을 사용한 고장 탈락 기법은 기존의 고장 진단 방법과 비교하여 평균 43% 줄어든 고장 진단 수행 시간을 나타내므로 빠르게 변화하는 시장에 대응하는 적절한 수단이 될 수 있다.

참 고 문 헌

[1] J. Lee, Y. Lim, H. Cho and S. Kang, "An accurate matching algorithm with essential factors for fault diagnosis", *in Proceedings of International SoC design Conference*, pp. 301-304, 2006.

[2] P. R. Menon, Y. Levendel and M. Abramovici, "Critical path tracing in sequential circuits", *in Proceedings of International Conference on Computer-Aided Design*, pp. 162-165, 1988.

[3] Y. Karkouri, E. M. Aboulhamid, E. Cerny and A. Verreault, "Use of fault dropping for multiple fault analysis", *in Transactions on Computers*, pp. 98-103, 1994.

[4] J. Lee, Y. Lim and S. Kang, "A fault dropping technique using fault scores for fast fault diagnosis", *in Proceedings of Korea Test Conference*, 2007.

[5] I. Pomeranz and S. M. Reddy, "The accidental detection index as a fault ordering heuristic for full-scan circuits", *in Proceedings of Design, Automation and Test in Europe*, pp. 1008-1013, 2005.

[6] H. D. Schnurmann, E. Lindbloom and R. G. Carpenter, "The weighted random test-pattern generator", *in Transactions on Computers*, Vol. C-24, pp. 695-700, 1975.

[7] K. Shigeta and T. Ishiyama, "An improved fault diagnosis algorithm based on path tracing with dynamic circuit extraction", *in Proceedings of International Test Conference*, pp. 235-244, 2000.

[8] V. C. Vimjam and M. S. Hsiao, "Efficient fault collapsing via generalized dominance relations", *in Proceedings of VLSI Test Symposium*, 2006.

[9] H. K. Lee and D. S. Ha, "HOPE: an efficient parallel fault simulator for synchronous sequential circuits", *in Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, pp. 1048-1058, 1996.

[10] TetraMax Reference Manual. Release 2004. 12, Synopsys Inc., Mountain View, CA, 2001.

저 자 소 개

이 주 환(학생회원)

2003년 연세대학교 전기전자 공학과 학사 졸업.
2005년 연세대학교 전기전자 공학과 석사 졸업.
2009년 현재 연세대학교 전기 전자공학과 박사 과정.

<주관심분야 : SoC 설계, Diagnosis, BIRA, CAD, DFT>

김 홍 식(정회원)

1997년 연세대학교 전기공학과 학사 졸업.
1999년 연세대학교 전기 및 컴퓨터 공학과 석사 졸업.
2004년 연세대학교 전기전자 공학과 박사 졸업.
2004년~2005년 Virginia 공대 박사 후 연구원.
2006년 삼성전자 시스템 LSI 사업부 책임연구원.
2009년 현재 연세대학교 TMS 사업단 연구교수.

<주관심분야 : SoC 설계, 테스트>

임 요 섭(학생회원)

2004년 연세대학교 전기전자 공학과 학사 졸업.
2006년 연세대학교 전기전자 공학과 석사 졸업.
2009년 현재 연세대학교 전기전자 공학과 박사 과정.

<주관심분야 : Diagnosis, CAD, DFT>

강 성 호(평생회원)

1986년 서울대학교 제어계측 공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터 공학과 석사 졸업.
1992년 The University of Texas, Austin 전기 및 컴퓨터 공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원.
1994년 Motorola Inc. 선임 연구원.
2009년 현재 연세대학교 전기전자공학과 교수.

<주관심분야 : SoC 설계, SoC 테스트>