논문 00-25-12A-19 한국통신학회논문지 '00-12 Vol.25 No.12A

합선고장을 검출하기 위한 IDDQ 테스트 패턴 생성에 관한 연구

정회원 배 성 환*, 김 대 익**, 전 병 실***

A Study on IDDQ Test Pattern Generation for Bridging Fault Detection

Sung-hwan Bae*, Dae-ik Kim**, Byoung-sil Chon*** *Regular Members*

요 약

IDDQ 테스팅은 CMOS에서 발생 빈도가 가장 높은 합선고장을 효과적으로 검출할 수 있는 기법이다. 본 논문에서는 테스트 대상 회로의 게이트간에 발생 가능한 모든 단락을 고려하여, 이러한 결함을 효과적으로 검출하기 위한 테스트 패턴 생성기와 고장 시뮬레이터를 구현하였다. 구현된 테스트 패턴 생성기와 고장 시뮬레이터는 O(n2)의 복잡도를 가지는 합선고장을 효과적으로 표현하기 위한 기법과 제안된 테스트 패턴 생성 알고리즘 및 고장 collapsing 알고리즘을 이용하여 빠른 고장 시뮬레이션 수행시간과 높은 고장 검출률을 유지하면서 적은 수의 테스트 패턴의 생성이 가능하다. ISCAS 벤치마크 회로에 대한 실험을 통하여 기존의 다른 방식보다 성능이 우수함을 보여주었다.

ABSTRACT

IDDQ Testing is a very effective testing method to detect bridging faults occurred in CMOS circuits. In this paper, we consider shorts between gates within circuit under test and implement IDDQ(quiescent power supply current) test pattern generator and fault simulator. Implemented test pattern generator and fault simulator use a new efficient test pattern generation algorithm and fault collapsing schemes to achieve fast run time, high fault coverage and short test sets. Experimental results for ISCAS benchmark circuits demonstrate its efficiency in comparison with results of previous methods.

Ⅰ. 서 론

최근에는 VLSI 회로의 집적도가 증가함에 따라 기존의 단일 고착고장은 CMOS 회로에서 발생 가능한 많은 수의 물리적 결함 및 고장을 효과적으로 모델링 할 수 없음이 밝혀졌다[1]. 합선고장은 두선 혹은 그 이상의 선이 서로 의도하지 않게 연결되는 고장형태이며 CMOS 회로에서 발생 빈도가 가장 높은 고장이다. 또한, 단일 고착고장으로는 모델링 되지 못하는 많은 결함을 포함하고 있으며, 실제 CMOS 결함의 40~50% 정도가 합선고장을 이용하여 효과적으로 모델링 할 수 있다.

따라서 CMOS에서 발생 빈도가 가장 높은 합선고장을 효과적으로 검출할 수 있는 테스트 기법이 필요하다[2]. 최근 IDDQ 테스트는 CMOS에서 발생 가능한 여러 종류의 물리적 결함을 효율적으로 검출 할 수 있는 테스트 방식으로 소개되었다[3]. CMOS 게이트는 nMOS 풀다운(pulldown) 블록과

* 한려대학교 멀티미디어정보통신공학과 (shbae@hlu.hanlyo.ac.kr),
** 전북대학교 전기전자회로합성연구소 (dikim@idec.chonbuk.ac.kr)
*** 전북대학교 전자정보공학부 (bschon@moak.chonbuk.ac.kr)
논문번호 : 00246-0704, 접수일자 : 2000년 7월 4일

pMOS 풀업(pullup) 블록으로 이루어져 있어 고장이 없는 회로에서는 VDD와 GND 사이에 전류의 도통 경로가 형성되지 않는다. 따라서 이 경우 정적 상태에서는 수 nA정도의 무시할 만한 누설 전류(leakage current)만이 흐르게 된다. 그러나, 게이트 옥사이드 단락, 합선 결함, 기생 트랜지스터 누설, 누설 PN 결합, 개방 결함, 그리고 전송 게이트의 개방 등과 같은 물리적 결함은 CMOS로 이루어진 회로의 정적상태 전류를 증가시켜 많은 전력을 소비하도록 야기 시킨다.

두 선 l_1과 l_2 사이에 발생한 합선고장을 검출하기 위해서 IDDQ 테스트를 수행하는 경우, l_1은 "1" 논리 값을 그리고 l_2는 "0" 논리 값을 갖도록(혹은 그 반대로)해주면 합선된 두 선을 통과하는 VDD와 GND 사이의 전류경로가 형성되어 회로의 정적상태에서도 많은 양의 전류가 흐르게 되어 가정한 고장을 검출할 수 있다[4]. IDDQ 테스트 방식을 이용하여 합선고장을 검출할 경우에는 두 가지에 문제점이 존재하게 된다. 첫째, 합선고장은 $O(n^2)$의 복잡도를 가지게 되어 테스트 패턴 생성기와 고장 시뮬레이터를 이용하여 고장을 검출하기 위해서는 전체 고장을 표현하는 기법과 무해고장(redundant fault)을 효율적으로 찾아내는 알고리즘이 필요하다. 둘째, IDDQ 테스트 방식은 높은 고장 및 결함 검출률을 갖지만 상대적으로 전압 테스트 방식에 비해 느린 테스트 시간을 가진다. 따라서 높은 고장을 유지하면서 가능한 적은 수의 테스트 패턴을 얻어야 한다.

본 논문에서는 $O(n^2)$의 복잡도를 가지는 합선고장을 효과적으로 표현하기 위한 기법과 새롭게 제안한 고장 collapsing 알고리즘을 이용한다. 또한 전압 테스트 방식에 비해 상대적으로 느린 IDDQ 테스트를 위해서 새로운 테스트 패턴 생성 알고리즘을 제안한다. 제안된 알고리즘은 빠른 고장 시뮬레이션 수행시간을 가지게 되어 짧은 시간에 많은 테스트 패턴의 적용이 가능하여 높은 고장 검출률을 유지하면서 적은 수의 테스트 패턴의 생성이 가능하다. 본 논문은 2장에서 고장 모델과 제안된 테스트 패턴 생성 알고리즘에 관해서 논의하고, 3장에서는 제안된 고장 collapsing 기술에 관해서 설명한다. 또한, 4장에는 구현된 테스트 패턴 생성기와 고장 시뮬레이터에 대한 모의 실험결과를 검토하고 5장에서 결론을 맺는다.

Ⅲ. 고장 모델 및 테스트 패턴 생성 알고리즘

1. 고장 모델

가정한 고장 모델은 테스트 회로의 게이트간에 발생 가능한 모든 단락으로 인한 합선고장이다[5-7]. 테스트 회로에서 합선고장이 발생한 경우, IDDQ 테스트는 합선이 발생한 두 선 l_1과 l_2 사이에 l_1은 "1" 논리 값을 그리고 l_2는 "0" 논리 값을 갖도록(혹은 그 반대로) 해 준다면 회로의 정적 상태에서도 VDD와 GND 사이에 전류의 경로가 생성된다. 그림 1에는 노드 a와 노드 g사이에 합선고장이 발생한 예로서 IDDQ 테스트를 이용할 경우, 입력 노드 a, b, c에 "1 0 1"의 테스트 벡터를 인가하면 노드 d, e, g는 "1 0 0"이 되어 VDD와 GND 사이에 경로가 형성되어 높은 IDD가 흐르게 됨으로 고장을 검출 할 수 있다.

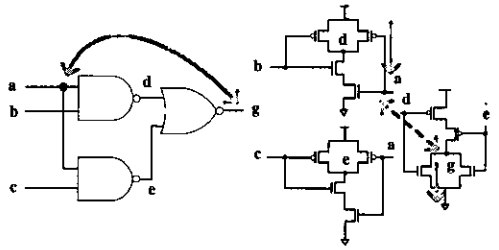

그림 1. 합선고장의 예

이러한 기본적인 IDDQ 테스트 원리를 응용하여 임의의 테스트 패턴에 같은 값을 갖는 노드별로 등가그룹을 할당하게 되면 다음의 정의와 식을 이용하여 고장 검출률의 계산이 가능하다.

- 전체 합선고장 수(A)

$$= {}_nC_2 = n(n-1)/2 \quad (n : \text{노드 수}) \qquad (1)$$

- 검출하지 못한 합선고장 수(B)

$$= \sum_i \frac{|G_i| * (|G_i|-1)}{2} \qquad (2)$$

($|G_i|$: i번째 등가그룹의 수)

$$\text{- 고장 검출률(\%)} = \frac{A-B}{A} \times 100 \qquad (3)$$

예를들어 임의의 테스트 패턴 T_1 = "0 0 0"을 그림 1의 테스트 회로의 입력 노드에 인가되면 다음과 같은 G_1 = { a, b, c, g }, G_2 = { d, e }의 두 등가그룹(i = 2)을 얻게된다. 수식 (1), (2), (3)를 이용하여 테스트 패턴 T_1의 고장 검출률을 계산할 수 있다.

- 전체 합선고장 수(A) = n(n - 1)/2 = (6×5)/2 = 15

- 검출하지 못한 합선고장 수(B)
  = $G_1(G_1 - 1)/2 + G_2(G_2 - 1)/2 = 6 + 1 = 7$

- 고장 검출률(%) = $\frac{A - B}{A} \times 100 = 53.3\%$

만약, 그림 1에 또 다른 임의의 테스트 패턴 P_1 = "0 0 1"을 테스트 회로의 입력 노드에 인가할 경우 새로운 G_1 = { a, b, g }, G_2 = { c, d, e } 의 등가그룹을 얻게되어 수식 (1), (2), (3)를 이용하여 테스트 패턴 P_1의 고장 검출률을 계산하면 60%가 됨을 알 수 있다. 따라서 위의 예를 통해서 임의의 테스트 패턴 T_1과 P_1을 입력 노드에 인가할 경우에는 각각 53.3%와 60%의 다른 고장 검출률을 얻었다. 이러한 결과를 통해서, 효율적인 테스트 패턴 생성 알고리즘을 이용하여 단계별로 가장 많은 수의 합선고장을 검출할 수 있는 최적의 테스트 패턴을 찾아내면 최소의 테스트 패턴 수로 높은 고장 검출률을 얻을 수 있다.

2. 제안된 테스트 패턴 생성 알고리즘

IDDQ 테스트 방식은 기존의 전압 테스트 방식에 비해 높은 고장 및 결함 검출 능력을 가진다. 그러나 결함이나 고장의 유·무에 관계없이 과도기간에는 높은 IDD를 가지게 되어 전류 테스팅 기법을 이용할 경우, 전류는 과도 상태를 지난 후, 정적상태에서 IDD를 측정해야 된다. IDDQ 테스트를 이용하여 합선고장을 효과적으로 검출하기 위해서는 다음과 같은 조건을 만족시키는 효율적인 테스트 패턴 생성 알고리즘이 요구된다.

- 높은 고장 검출률
- 최소의 테스트 패턴 수
- 빠른 테스트 패턴 생성 시간

일반적인 시뮬레이션의 경우 주 입력단에 인가되는 값에 따라 전체소자의 2.5%만이 값이 변화한다고 알려져있다. 디지털 회로의 시뮬레이션은 회로가 커질수록 연산량이 많아져서 시뮬레이션 시간이 길어지므로, 본 논문에서는 빠른 테스트 패턴 생성을 위해서 비트반전 테스트 패턴 생성 알고리즘을 제안한다. 제안된 테스트 생성 알고리즘은 고장 시뮬레이션을 위해 입력단에 인가된 테스트 패턴은 무작위로 선택된 임의의 패턴에 대해서 한 비트만 반전하여 연속적으로 적용함으로서, 테스트 대상회로의 1개 입력 노드만이 값이 변하게 되어 시뮬레이션 속도가 빨라지고 무작위 패턴의 이용으로 최적의 테스트 패턴을 생성하기 위해 많은 테스트 패턴의 적용이 가능하다. 또한 입력 탐색방식을 적용하여 회로의 모든 소자가 아닌 입력단자의 값이 변하는 소자만 연산하는 기법을 적용하여 빠른 테스트 패턴 생성이 가능하다.

기존의 무작위 테스트 패턴 생성 알고리즘은 임의로 생성된 테스트 패턴을 회로의 입력노드에 차례로 인가하여 단계별로 가장 많은 고장을 검출하는 테스트 패턴을 선택하는 단순한 알고리즘이고[8], Genetic algorithm을 이용한 테스트 패턴 생성 알고리즘은 population 크기, generation 수, mutation 확률, fitness 함수 등 다양한 파라메터 값을 결정과 계산에 많은 시간이 요구된다[7]. 제안된 비트반전 알고리즘은 그림 2에 보인바와 같이 입력노드에 인가되는 테스트 패턴이 단지 1-비트 변화가 발생하여 값이 변하는 소자의 수를 급격히 줄여 빠른 시뮬레이션 수행이 가능하다. 따라서 짧은 시간에 많은 입력 테스트 패턴적용이 가능하게 되어, 높은 고장 검출률을 유지하면서 적은 테스트 패턴의 생성이 가능하다. 제안한 알고리즘은 다음의 3단계를 통하여 하나의 테스트 패턴을 생성하게 된다.

단계 1 :

무작위로 생성된 테스트 패턴 tp0을 테스트 대상 회로에 인가하고 고장 시뮬레이션을 통해 얻어진 검출 고장 수는 detectFault에 보관하고, 테스트 패턴 tp0는 tp에 보관한다. 테스트 패턴 tp0의 임의의 j번째 비트를 반전하여 새롭게 생성된 테스트 패턴 tp0의 고장 검출 수는 NewDetF에 보관한다.

만약, NewDetF의 값이 detectFault 보다 큰 경우에 테스트 패턴 tp0은 tp에 새롭게 보관하고 detectFault의 값은 NewDetF 값으로 대치된다. detectFault의 값이 큰 경우에는 Unimp의 값은 1 증가한다. 다음, tp0의 j+1 비트가 반전되어 새로운 테스트 패턴을 생성하게 되어 다시 고장 시뮬레이션을 수행한다. 이러한 반복된 테스트 패턴 생성과 고장 시뮬레이션을 통해서 가장 높은 고장 검출 수를 가지는 패턴이 생성된다. 만약, Unimp가 테스트 대상 회로의 전체 입력노드의 수 NPI보다 큰 경우나 LoopCounter의 수가 2×NPI인 경우에 단계 1의 고장 시뮬레이션은 단계 2로 진행된다.

단계 2 :

단계 1의 진행 후 변수 j의 값은 1 증가하게되고,

새롭게 무작위로 생성된 임의의 테스트 패턴 tp0에 대해서 단계 1의 과정을 반복 수행하게 된다. i의 값은 초기에 설정된 nSeed 만큼 지속적으로 증가하여 단계 1을 반복 수행한다.

단계 3 :

단계 1과 2를 통해서 얻어진 테스트 패턴 tp를 이용하여 고장 시뮬레이션을 수행하여 현재 등가노드 그룹을 새로운 다음 단계의 등가노드 그룹을 생성시킨다.

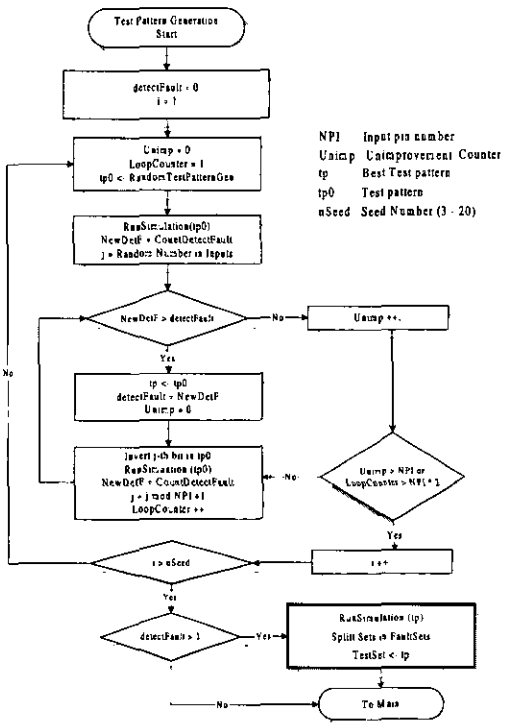

그림 2. 제안한 테스트 패턴 생성 알고리즘

Ⅲ. 제안된 고장 collapsing 알고리즘

두 개의 합선된 노드가 항상 같은 논리 값을 갖는 경우, 두 단락된 노드에 "1"과 "0" 값의 적용이 불가능하게 된다. 이러한 합선고장은 모든 테스트 패턴으로도 검출이 불가능하여 무해고장으로 부르며, 무해고장을 효과적으로 검출하는 기술을 고장 collapsing이라 부른다[5]. 효율적인 고장 collapsing 알고리즘은 IDDQ 테스트로 검출하지 못하는 노드를 삭제함으로써, 고장 시뮬레이션과 테스트 패턴 생성에 요구되는 시간을 줄여준다. 기존의 고장 collapsing 알고리즘[5-7]에서는 무해고장을 효과적으로 검출하기 위해 다음과 같은 3가지 경우를 고려하였다. 따라서 1, 2, 3의 경우에 해당되는 노드사이에 합선고장이 발생할 경우 검출이 불가능하게 된다.

- 경우 1 : a = BUFF(b)
- 경우 2 : a = INV (INV (b))
- 경우 3 : 기본 게이트의 종류가 같고 공통의 입력을 받는 경우

본 논문에서 제안한 알고리즘의 기본 collapsing 방법은 기존의 고장 collapsing 방법과 비슷하다. 그러나 노드 값에 의한 경로탐색 방식을 이용하기 때문에 기존의 방식에서 검출하지 못한 등가노드를 효율적으로 검출할 수 있다(예 : 그림 4). 또한 기존에 경우 1, 2, 3의 단순한 방식에 비해서 확대된 고장 collapsing 방식을 이용하기 때문에 테스트 대상회로에서 발생 가능한 등가노드를 더 찾을 수 있다. 이는 netlist 정보를 제공하는 벤치마크 회로의 시뮬레이션 결과에서 증명된다. 제안된 고장 collapsing 방법은 다음과 같다.

1. 버퍼와 인버터

버퍼와 인버터의 경우는 입력이 하나이기 때문에 여러 개의 인버터와 버퍼가 연결되어 있을 때 [5-7]과 비슷한 방법으로 경로를 탐색하여 항상 같은 값을 갖는 노드들을 collapsing 한다. 본 논문에서의 경로탐색 알고리즘은 노드값에 의한 방식을 이용하여 최대의 연결 경로를 찾는다. 예를 들어 그림 3과 같은 연결 구조에서 인버터 G5부터 입력 게이트를 찾아가면 OR게이트 G1에서 끝난다. 연결경로에 있는 노드 G2, G3, G4, G5중 G2, G3, G5는 게이트 G1의 반대 값인 "-G1"을 갖게되어 등가노드가 형성되며 이를 그룹 1로 설정하고, G1, G4는 "G1"값을 가져서 그룹 2로 지정된다. 따라서 {G2, G3, G5}와 {G1, G4}의 등가노드 그룹을 찾게 된다. 본 논문에서 적용한 노드 값에 의한 등가관계를 탐색하는 알고리즘을 이용할 경우에는 기존의 알고리즘에서 구한 등가노드 외에 항상 상수 값을 가지는 노드와 이로 인한 추가적인 등가노드를 검출할 수 있는 장점이 존재한다.

예를들어 그림 4의 회로는 ISCAS '85 벤치마크 c2670의 일부분 예이다. 노드 2671은 입력이 노드 2291과 2531을 갖지만, 노드 2531은 인버터로 출력 값이 "-2291"을 가진다. 따라서 노드 2671은 "2291"과 "-2291"의 노드 값을 가지게되어 항상 0

인 출력 값을 가지게 된다. 또한 노드 2784는 노드 2671과 2531을 입력으로 한다. 여기서 노드 2671이 항상 0의 노드 값을 가지게 되므로 OR 게이트의 출력 값은 "-2291"이 되어 노드 2531과 등가노드가 된다. 값에 의한 등가노드를 탐색하는 알고리즘은 추가적인 등가노드 {2531, 2784}와 상수노드 {2671}을 검출 할 수 있다.

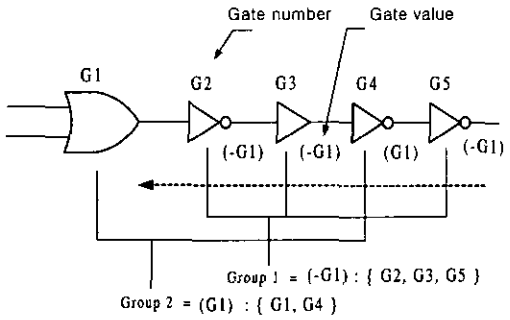

그림 3. 버퍼와 인버터의 고장 collapsing

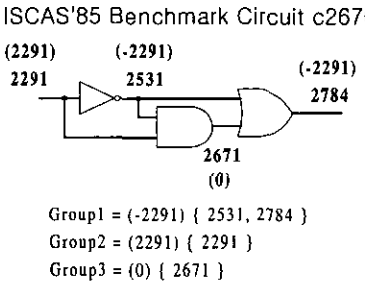

그림 4. 노드 값을 이용하는 고장 collapsing 방법

2. 기본게이트

테스트 대상 회로에서 버퍼와 인버터를 제외한 기본 게이트(AND, OR, NAND, NOR, XOR)에 대하여 Thadikaran[6]의 고장 collapsing 원리를 확대하여 게이트의 종류는 다르지만 입력 노드가 같은 경우, 입력 노드의 그룹 번호가 같은 경우, 출력이 항상 상수 값을 가지는 경우에 적용하였다. 그림 5에 확대된 collapsing 예를 보인다. 그림에서 ISCAS '85 벤치마크 c3540의 노드 2745와 2746은 종류는 다르지만 같은 입력 값을 받는다. 이러한 경우 노드 2962는 노드 2745와 등가노드가 된다. 또한 게이트 3195는 같은 입력을 받게되어 인버터와 같은 기능을 하게되어 노드 2746과 등가노드가 된다. 본 논문에서 제안한 값에 의한 경로탐색 알고리즘과 확대된 collapsing 알고리즘을 이용하여 표 1에는 ISCAS '85 벤치마크 회로에 적용한 결과를 보인다.

기존의 고장 collapsing 방법에 비해서 제안된 알고리즘이 효과적으로 고장 collapsing을 수행하여 더 많은 수의 무해고장을 검출함을 알 수 있다.

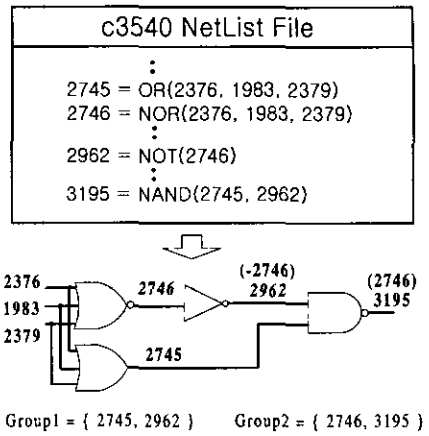

그림 5. 확대된 고장 collapsing 방법

표 1. 합선고장의 collapsing 결과 비교

회로	합선고장수 (노드 수)	Reddy[5]	Thadikaran [6]	Shinogi [7]	제안한 방 법
c432	19,110 (196)	18,145 (191)	18,145 (191)	-	18,145 (191)
c499	29,403 (243)	22,155 (211)	22,155 (211)	-	22,155 (211)
c880	97,903 (443)	78,210 (396)	75,466 (389)	-	75,466 (389)
c1355	171,991 (587)	136,503 (523)	136,503 (523)	136,503 (523)	136,503 (523)
c1908	416,328 (913)	270,480 (736)	191,271 (619)	270,480 (736)	115,440 (481)
c2670	1,016,025 (1,426)	746,031 (1,222)	627,760 (1,121)	746,031 (1,222)	440,391 (939)
c3540	1,476,621 (1,719)	1,010,331 (1,422)	668,746 (1,157)	823,686 (1,284)	517,653 (1,018)
c5315	3,086,370 (2,485)	2,316,628 (2,153)	1,867,278 (1,933)	2,316,628 (2,153)	1,540,890 (1,756)
c6288	2,995,128 (2,448)	2,956,096 (2,432)	2,956,096 (2,432)	2,956,096 (2,432)	2,842,920 (2,385)
c7552	6,913,621 (3,719)	5,022,865 (3,170)	3,924,201 (2,802)	5.019,696 (3,169)	3,081,403 (2,483)

Ⅳ. 테스트 패턴 생성기와 고장 시뮬레이터의 모의 실험 및 검토

개발한 테스트 패턴 생성기와 고장 시뮬레이터는 PC의 Windows 기반에서 Visual Basic 소프트웨어를 이용하여 사용자의 편의를 위해 GUI 방식으로 구현하였다. 그림 6에 개발한 테스트 패턴 생성기와 고장 시뮬레이터를 화면을 보여 주고 있다.

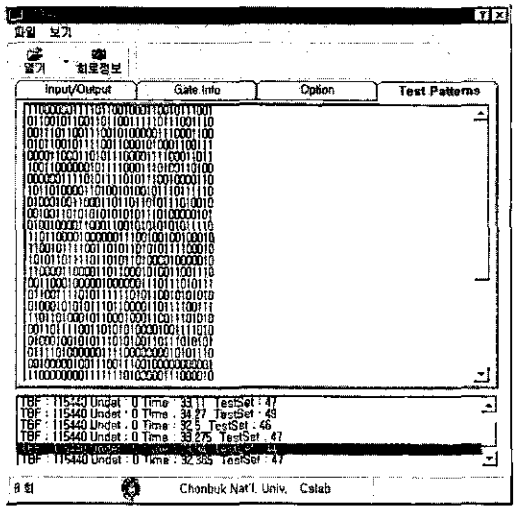

그림 6. 구현된 테스트 패턴 생성기와 고장 시뮬레이터

그림은 ISCAS '85 벤치마크 회로 c1908을 20회 시뮬레이션 한 결과를 보인다. 고장 시뮬레이션을 수행할 경우, 제안된 비트반전 테스트 생성 알고리즘이 지역 해로 수렴되는 것을 막기 위해 초기 Seed의 개수는 3~20개로 선택한다. 조합회로로 구성되어 있는 ISCAS '85 벤치마크 회로에 대해, 제안된 알고리즘을 이용하여 구현된 테스트 패턴 생성기와 고장 시뮬레이터의 성능 평가를 표 2에 보인다.

표 2. ISCAS '85 벤치마크 회로에 대한 시뮬레이션 결과

회로	테스트 패턴 수	미검출 합선고장	고장 검출률(%)	CPU time (min)
c432	14	0	100.00	0.03
c499	32	0	100.00	0.07
c880	16	0	100.00	0.12
c1355	64	0	100.00	0.21
c1908	44	0	100.00	0.30
c2670	24	0	100.00	1.35
c3540	51	2	99.999	0.7
c5315	28	3	99.999	2.6
c6288	23	0	100.00	0.6
c7552	37	104	99.997	4.7

효율성을 평가하기 위해 고장 검출률, 생성 테스트 패턴의 개수, 테스트 패턴 생성 시간을 기존 방식과 비교하였다. 구현한 테스트 패턴 생성기와 고장 시뮬레이터의 고장 검출률을 기존에 발표된 방식과 비교하여 표 3과 4에 보였다. 모든 ISCAS '85 벤치마크 회로에서 기존에 구현한 방식보다 높은 고장 검출률을 보이고 있다. 특히 c3540, c5315, c7552를 제외한 모든 회로에서 100%의 고장 검출률을 얻을 수 있다.

표 3. 고장 검출률 비교

(단위 : %)

회 로	Reddy[5]	Thadi-karan[6]	Shinogi[7]	제안한 방 법
c432	100.00	99.96	-	100.00
c499	100.00	99.72	-	100.00
c880	99.98	-	-	100.00
c1355	100.00	100.00	100.00	100.00
c1908	99.86	99.77	99.86	100.00
c2670	99.95	99.90	99.95	100.00
c3540	99.85	99.86	99.95	99.999
c5315	99.97	99.95	99.97	99.999
c6288	99.99	99.99	99.99	100.00
c7552	99.97	99.95	99.97	99.997

표 4. 생성된 테스트 패턴 개수의 비교

회 로	Reddy[5]	Thadi-karan[6]	Shinogi[7]	제안한 방 법
c432	15	19	-	14
c499	32	25	-	32
c880	17	-	-	16
c1355	66	68	63	64
c1908	44	44	43	44
c2670	23	29	20	24
c3540	52	61	52	51
c5315	27	36	25	28
c6288	23	28	21	23
c7552	36	51	27	37

표 3과 4를 통해서, 제안한 알고리즘을 이용한 테스트 패턴 생성기와 고장 시뮬레이터는 합선고장을 기존에 제안된 방법과 비교해서 비슷한 수의 테스트 패턴을 생성하여 최소 99.99% 이상의 높은 고장 검출률을 얻을 수 있음을 알 수 있다. 테스트 패턴을 생성하는데 소요되는 시간을 표 5에 보여주었다. Reddy의 경우는 HP Model 705 워크스테이션을 사용하였고, Thadikaran은 SUN SPARC 2 워크스테이션을 Shinogi는 Pentium-pro 200MHz를 사용하였다. 본 논문에서는 Pentium-pro 400MHz를 이용하여 서로 사용한 하드웨어 플랫폼이 다르기 때문에 정확한 성능 비교는 어렵지만 모든 벤치마크 회로에서 짧은 시간 내에 테스트 패턴을 생성해 줌을 확인할 수 있다. 특히 본 논문의 테스트 패턴

생성시간은 가장 큰 c7552에서도 5분 이내로 기존의 테스트 생성시간에 비해 크게 향상되었음을 알 수 있다.

표 5. 테스트 패턴 생성 시간의 비교

(단위 : min)

회로	Reddy[5] (HP Model 705)	Thadi-karan[6] (SUN-SPARC 2)	Shinogi[7] (Pentium-pro-200)	제안한 방법 (Pentium-pro-400)
c432	2.6	1	-	0.03
c499	7.9	1	-	0.07
c880	7.3	-	-	0.12
c1355	87	1	0.2	0.21
c1908	37	6	0.2	0.49
c2670	33	8	1.2	1.35
c3540	78	12	0.8	0.7
c5315	52	14	2.4	2.6
c6288	56	13	0.5	0.6
c7552	92	25	4.6	4.7

앞의 성능 비교 평가를 통해 본 논문에서 구현한 테스트 패턴 발생기와 고장 시뮬레이터의 효율성이 기존의 다른 방식들에 비해 증대되었음을 알 수 있다. 또한 효율적인 고장 collapsing의 결과로 검출하지 못한 합선고장을 쉽게 구할 수 있다. 예를 들어 본 논문의 고장 시뮬레이션을 통해서 벤치마크 회로 c3540의 경우에 노드 {4493, 4545}와 {4186, 4331}의 합선 고장을 검출하지 못했고, c5315의 경우 노드 {6566, 6955}, {7723, 7782}, {7769, 7771}의 합선고장을 검출하지 못했다. 검출하지 못한 합선고장은 결정론적 방법(deterministic method)을 이용하여 검출이 가능한 테스트 패턴이 존재하는지 여부를 조사할 수 있다.

V. 결 론

CMOS VLSI 회로에서 발생 빈도가 가장 높은 합선고장을 효과적으로 검출하기 위한 IDDQ 테스트용 테스트 패턴 생성기와 고장 시뮬레이터를 개발하였다. 구현한 테스트 패턴 생성기와 고장 시뮬레이터는 PC의 Windows 기반에서 Visual Basic 소프트웨어를 이용하여 사용자의 편의를 위해 GUI 방식으로 구현하였으며, ISCAS 벤치마크 회로를 이용한 시뮬레이션 결과로 고장 검출률, 생성된 테스트 패턴의 개수, 테스트 패턴 생성 시간이 기존의 제안된 방식에 비해 효율성이 증가되었음을 확인하였다. 고장 검출률에서는 최소 99.99% 이상을 보였고(표 3참조), 테스트 생성시간이 가장 큰 c7552의 경우에서도 5분 이내의 빠른 패턴 생성시간을 보였다(표 4와 표5 참조). 또한 높은 고장 검출률의 결과로 검출하지 못한 적은 수의 합선고장은 결정론적 방법을 이용하여 검출이 가능한 테스트 패턴의 존재 여부를 조사할 수 있는 장점을 가진다.

이러한 결과는 제안한 비트반전 테스트 패턴 생성 알고리즘을 이용하여 빠른 고장 시뮬레이션을 통해 많은 테스트 패턴의 적용과 새롭게 제안한 고장 collapsing 알고리즘을 이용하여 합선고장 수를 효과적으로 감소시킨 결과이다. 따라서 구현된 IDDQ용 테스트 패턴 생성기와 고장 시뮬레이터를 이용하여 적은 수의 테스트 패턴을 생성할 경우, 상대적으로 느린 전류 테스트의 단점을 보완할 수 있으며 기존의 전압 테스트 방식에 비해서 신뢰성 있는 테스트가 가능하다. 생성된 테스트 패턴과 내장형 전류 감지기(Built-in Current Sensor)를 이용하여 내장형 자체 테스트(*Built-in Self Test*) 기법에 응용할 경우, 더 빠르고 효율적인 테스트 수행이 가능하다. 향후 순차회로에 적용시키는 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] J. A. Abraham, "Challenges in fault detection," *International Symposium on Fault-Tolerant Computing*, pp. 96-114, 1995.

[2] 전병실 외, "합선고장을 위한 IDDQ 테스트 패턴 발생기의 구현," *한국통신학회논문지, vol. 24*, no. 12-A, pp. 2008-2014, 1999.

[3] R. Rajsuman, *IDDQ Testing for CMOS VLSI*, Artech House, 1994.

[4] 전병실 외, "기능테스트와 IDDQ 테스트를 위한 자체 점검 BIST 회로의 설계," 서울대학교 반도체공동연구소 연구보고서, 1998.

[5] R. S. Reddy, I. Pomerantz, S. M. Reddy, S. Kajihara, "Compact test generation for bridging faults under IDDQ testing," *IEEE VLSI Test Symposium*, pp. 310-315, 1995.

[6] P. J. Thadikaran, "Evaluation, selection and generation of IDDQ tests," PHD. Thesis, Department of Computer Science, State University of New York, 1996.

[7] T. Shinogi and T. Hayashi, "An iterative

improvement method for generating compact tests for IDDQ testing of bridging faults," *IEICE Trans. INF & SYST.*, vol. E81-D. no. 7, July 1998.

[8] S. Chakravarty and P. J. Thadikaran, "Simulation and generation of IDDQ tests for bridging faults in combinational circuits," *IEEE Trans. Computers,* vol. 45, no. 10, pp. 1131-1140, Oct. 1996.

배 성 환(Sung-hwan Bae) 정회원

1993년 2월 : 전북대학교 전자공학과 학사
1995년 2월 : 전북대학교 대학원 전자공학과 석사
2000년 2월 : 전북대학교 대학원 전자공학과 박사
2000년~현재 : 한려대학교 멀티미디어정보통신공학과 전임강사
<주관심 분야> Iddq 테스팅, VLSI 설계, Built-In self Test

김 대 익(Dae-ik Kim) 정회원

1991년 2월 : 전북대학교 전자공학과 학사
1993년 2월 : 전북대학교 대학원 전자공학과 석사
1996년 8월 : 전북대학교 대학원 전자공학과 박사
2000년~현재 : 전북대학교 전기전자회로합성연구소 Post Doctor
<주관심 분야> 저전력 디지털 시스템 설계, ASIC 테스팅

전 병 실(Byoung-sil Chon) 정회원

제25권 제11호 참조
현재 : 전북대학교 공과대학 전자정보공학부 교수 전북대학교 정보통신연구소 연구원
<주관심 분야> ATM switching, Design for testability