CMOS 조합회로의 IDDQ 테스트패턴 생성

김강철*, 송근호**, 한석붕**

IDDQ Test Pattern Generation in CMOS Circuits

Kang-Chul Kim*, Geun-Ho Song**, Seok-Bung Han**

본 연구는 경상대학교부설 정보통신연구센터의 부분적인 지원에 의한 것임

요 약

본 논문에서는 새로운 동적 컴팩션(dynamic compaction) 알고리즘을 제안하고 이용하여 CMOS 디지털 회로의 IDDQ 테스트패턴 생성한다. 제안된 알고리즘은 프리미티브 게이트 내부에서 발생하는 GOS, 브리징 고장을 검출할 수 있는 프리미티브 고장패턴을 이용하여 초기 테스트패턴을 구하고, 초기 테스트패턴에 있을 수 있는 don't care(X)의 수를 줄여 테스트 패턴의 수를 감소시킨다. 그리고 난수와 4 가지 제어도(controllability)를 사용하여 백트레이스를 수행시키는 방법을 제안한다. ISCAS-85 벤치마크 회로를 사용하여 모의 실험한 결과 큰 회로에서 기존의 정적 컴팩션 알고리즘에 비하여 45% 이상 테스트패턴 수가 감소함을 확인하였다.

Abstract

This paper proposes a new compaction algorithm for IDDQ testing in CMOS Circuits. A primary test pattern is generated by the primitive fault pattern which is able to detect GOS(gate-oxide short) and the bridging faults in an internal primitive gate. The new algorithm can reduce the number of the test vectors by decreasing the don't care(X) in the primary test pattern. The controllability of random number is used on processing of the backtrace together four ones of heuristics. The simulation results for the ISCAS-85 benchmark circuits show that the test vector reduction is more than 45% for the large circuits on the average compared to static compaction algorithms.

* 여수대학교 컴퓨터공학과

** 경상대학교 전자공학과

접수일자 : 1999년 2월 13일

Ⅰ. 서 론

반도체 공정 및 회로 설계 기술의 발전으로 수백만 개의 트랜지스터들을 단일 칩 상에 집적한 VLSI의 실용화가 이루어지고 있다. 이러한 IC들의 정확성과 신뢰도는 기계제품에 비하여 높지만 아직도 100%의 신뢰성을 보장하지는 못하는 실정이다. 통신, 의학, 우주항공 등의 응용에 있어서 IC의 결함은 치명적인 손상을 가져올 수도 있으므로 고장이 없는 IC의 제작은 제품의 신뢰도를 향상시키고 생산비를 절감하는 필수요인이 될 것이다. VLSI의 테스트 기술은 불량 반도체 소자를 검출하는 것으로 회로의 집적도가 증가될수록 더욱 철저한 테스트 과정이 요구된다. 특히 통신용 ASIC(Application Specific Integrated Circuits)에는 동작 시에 전력 소모가 거의 없고 스위칭 속도가 빠르며, 집적도가 높은 CMOS 회로가 많이 사용되므로 다층 금속 도선과 서브 마이크론 설계 규칙이 적용되어 설계와 제조과정에서 많은 물리적인 결함(defect)들이 발생하고 있다. 이러한 고장들은 대부분 논리고장은 발생하지 않으면서 신호지연이 발생하고, 시간이 지남에 따라서 그 상태가 더욱 악화되어 현장에서 사용하는 도중에 결국 논리 고장을 일으켜 시스템 신뢰도를 현저하게 감소시킨다[1]. 따라서 이에 대한 테스트는 더욱 큰 비중을 차지하게 되어 칩 내의 고장들을 검출하기 위한 연구가 활발히 진행되고 있다[2,3,4].

대부분의 테스트패턴 생성 알고리즘은 테스트 세트(test set)의 크기에는 상관없이 주어진 회로에 대한 완벽한 테스트 세트의 생성에 주력하고 있다. 그러나 테스트 세트의 크기는 테스트 적용 시간과 비용에 직접적으로 관련된다. 작은 테스트 세트는 테스트 적용 시간, 테스트 장비의 기억장치 용량 등 테스팅 비용을 줄이고, 칩의 제작 시에 테스팅을 위해 필요한 테스트 저장영역을 줄임으로써 IC의 하드웨어 비용을 줄인다. 특히 컴팩션(compaction)은 스캔 설계(scan design)에 매우 중요한데, 이는 테스트 적용 시간이 테스트 세트의 수와 스캔 체인(scan chain) 내의 flip-flop 수의 곱에 비례하여 증가하기 때문이다. 컴팩션의 이상적인 목표는 최소의 테스트 세트를 얻는 것이지만, 중복회로가 없는 조합 회로 내에 존재하는 모든 고장을 검출할 수 있는 최소의 테스트패턴 수를 계산하는 것은 NP-hard로 밝혀졌다[5].

컴팩션은 고장 검출율에는 아무런 영향을 미치지 않으면서 생성된 패턴을 합하거나 제거하여 패턴의 수를 줄이는 것이다. 테스트 세트 컴팩션을 위한 다양한 방법들이 제안되고 있는데, 이러한 방법들은 알고리즘의 적용 시기에 따라 정적 컴팩션(static compaction)과 동적 컴팩션(dynamic compaction)으로 분류된다. 정적 컴팩션은 전체 고장에 대하여 테스트패턴을 생성 후 컴팩션 알고리즘을 적용하여 패턴을 결합하거나 제거하여 테스트패턴 수를 줄이는 방법으로, reverse order fault simulation[6], ROTCO[7], essential fault pruning method[8], double detection method[9] 등의 방법들이 제안되었다. 이러한 방법들은 생성된 패턴의 형태와 패턴들의 컴팩션 되는 순서에 따라 컴팩션의 결과가 다르게 나타나고, 또한 처음에 생성된 많은 패턴들을 처리해야 하며 시뮬레이션에 관한 정보를 가지고 있어야 함으로 많은 메모리를 필요로 하는 단점이 있다. 동적 컴팩션은 테스트패턴을 생성하는 동안에 컴팩션 알고리즘을 같이 병행하여 테스트패턴을 생성하는 방법으로, dynamic compaction[10], maximal compaction[11], independent fault sets[12] 등의 방법들이 제안되었다. 이러한 방법들은 하나의 고장에 대하여 패턴을 생성 후, 정해지지 않고 don't care로 남아있는 입력들에 값을 할당하여 추가로 여러 개의 고장을 검출한다. 이렇게 하여 더 이상 결합될 수 없는 하나의 테스트패턴이 생성되면 시뮬레이션을 수행하여 대상 고장 이외에 우연히 검출된 고장을 고장 집합에서 제거하므로 각각의 고장에 대하여 패턴을 생성할 필요가 없다. 지금까지 제안된 컴팩션 알고리즘들은 대부분이 stuck-at 고장모델을 사용한 전압 테스팅 방식에 적용되고 있다.

본 논문에서는 CMOS를 사용하는 디지털 논리회로의 IDDQ 테스팅 방식에 적용할 수 있는 새로운 동적 컴팩션 알고리즘을 제안한다. 전압 테스팅 방식에서는 고장을 활성화시키고 주 출력에서 고장신호를 전파하여 정상상태와 고장상태의 출력값이 달라야 검출이 가능하지만, IDDQ 테스팅 방식에서는 고장 게이트의 입력에서 고장을 활성화시키기만

하면 고장을 검출할 수 있다. 이러한 장점을 이용하여 게이트 내부에 존재하는 GOS와 브리징 고장을 검출할 수 있는 입력 조합을 프리미티브 고장패턴으로 모델링하고, 이 모델링을 이용한 x_gate 개념을 도입하여 고장패턴 그레이딩과 동적 컴팩션 알고리즘을 효과적으로 수행할 수 있다. 그리고 컴팩션 알고리즘의 성능을 높이기 위하여 난수에 의한 백트레이스를 추가하였고, 기존의 네 가지 제어도와 난수에 의한 백트레이스를 순차적으로 순환시켜 패턴수를 더욱 감소시킨다.

본 논문의 Ⅱ장에서는 고장모델과 본 논문에서 제안한 컴팩션 알고리즘 및 순환 백트레이스에 대하여 기술한다. Ⅲ장에서는 벤치마크 회로들에 대한 모의 실험 결과를 보여주며, Ⅳ장에서 결론을 기술한다.

Ⅱ. 고장모델과 컴팩션 알고리즘

1. 고장패턴

본 논문에서는 NAND, AND, NOR, OR, XOR, INV(inverter), BUF(buffer) 등의 프리미티브 게이트를 대상으로 하며, 기존 고장모델의 한계를 극복할 수 있도록 스위치 레벨에서 고장을 모델링하여 게이트 내의 트랜지스터에서 발생하는 GOS와 게이트 내부의 모든 노드에서 발생하는 브리징 고장을 고려한다. 이 때, 브리징 고장은 저항 성분이 0인 완전한 단락을 의미하며, GOS는 게이트와 소오스(source), 게이트와 드레인(drain), 게이트와 벌크(bulk)간의 단락을 모델링한다. 만일 트랜지스터에서 단락이 발생하였다면 정적 상태에서도 VDD에서 GND로의 전류 통로가 생기게 된다. 따라서 이런 전류 통로를 형성시킬 수 있는 즉, 내부 노드의 단락이나 GOS 등을 활성화시킬 수 있는 입력 조합을 먼저 구한다.

게이트 내의 스위치 레벨에서 GOS와 브리징 고장 등 모든 고장들을 검출할 수 있는 입력 조합을 조사하여 이들 중 서로 중복되거나 불필요한 것들을 제거하여 최소의 게이트 입력 조합을 구하고, 이를 프리미티브 고장패턴이라고 정의한다[13]. 이러한 프리미티브 고장패턴을 각 게이트의 2-입력과 3-입력에 대해 구하여 표 3에 나타내었다. 여기서 알 수 있듯이 AND(NAND)와 OR(NOR)의 입력 조합 수는 입력 개수가 N일 때 (N + 1)개로 나타난다. 그러나 XOR는 입력 수에 관계없이 항상 3개로 나타나며 INV와 BUF는 1과 0으로 나타난다. 이러한 프리미티브 고장패턴만 있으면 게이트 내부에 존재하는 모든 브리징 고장과 GOS 고장을 검출할 수 있으며 stuck-at 및 stuck-on 고장까지도 검출할 수 있다. 예를 들어, 3입력 NOR 게이트의 경우 첫 프리미티브 고장패턴으로서 (0 0 0)에 대하여 테스트패턴을 생성한다. 다음으로 (0 0 1), (0 1 0) 그리고 (1 0 0)에 대해서 차례대로 테스트패턴을 생성하게 된다.

표 1. 프리미티브 게이트의 고장패턴
Table 1. Fault patterns of primitive gates

AND(NAND)				OR(NOR)				XOR				INV(BUF)
2 input		3 input		2 input		3 input		2 input		3 input		input
A	B	A	B	C	A	B	A	B	C	A	B	A	B	C	A
0	1	0	1	1	0	0	0	0	0	1	1	1	1	1	0
1	0	1	0	1	0	1	0	0	1	1	0	1	0	0	1
1	1	1	1	0	1	0	0	1	0	0	1	0	1	1	
		1	1	1			1	0	0						

2. 제안된 동적 컴팩션 알고리즘

본 논문에서 제안하는 컴팩션 알고리즘은 IDDQ 테스팅 방식에 적용될 수 있는 동적 컴팩션 알고리즘으로, 고장을 활성화시키고 고장 신호를 주 출력까지 전파해서 정상 상태값과 고장 상태값이 달라야 검출 가능한 전압 테스트와는 달리 고장을 활성화시키기만 하면 고장을 검출할 수 있는 IDDQ 테스팅의 장점을 이용한다. 먼저 하나의 대상 고장에 대하여 임시 패턴을 생성한다. 이렇게 생성된 패턴은 많은 don't care를 포함하고 있는데, 이 don't care에 논리값을 할당하여 검출 가능한 다른 고장을 추가로 검출하는 방법이다.

하나의 고장에 대하여 패턴이 생성되면 회로 내에는 입력에 don't care를 포함하는 많은 게이트들이 존재하며 이러한 게이트들을 선택하여 앞에서 구한 프리미티브 고장패턴을 가져온다. 제안된 알고리즘은 IDQ 테스팅 방식에 적용되므로 선택된

게이트의 프리미티브 고장패턴만 만족시키면 고장은 검출되는 것이다. 처음에 생성된 패턴은 변화시키지 않으면서 don't care에 값을 할당하여 패턴을 생성한다. Don't care에 논리값을 할당하여도 더 이상 검출되는 고장이 존재하지 않으면 지금까지 정해진 입력조합을 테스트패턴으로 저장한다. 즉 하나의 패턴으로 검출 가능한 모든 고장을 검출함으로써 적은 수의 패턴을 얻을 수 있다.

(1) X_gate

하나의 고장에 대하여 생성된 패턴을 논리 시뮬레이션을 수행 후 회로내 각 게이트의 입력을 조사하여 보면 입력에 don't care를 포함하는 게이트들이 존재하게 된다. 이러한 게이트들 중 다음의 두 조건을 모두 만족하는 게이트들을 x_gate라고 정의한다.

1. 입력에 적어도 하나 이상의 don't care를 가진다.
2. 게이트의 입력값과 프리미티브 고장패턴을 비교하여 같은 입력선에서 서로 다른 논리값을 가지지 않는 프리미티브 고장패턴이 적어도 하나 이상 존재해야 한다.

예를 들어 3 입력 NAND 게이트의 입력이 (1 X X)이라면 don't care를 입력에 포함하고 있으며, 프리미티브 고장패턴 (0 1 1), (1 0 1), (1 1 0), (1 1 1) 중에서 (0 1 1)를 제외한 다른 패턴과는 충돌이 발생하지 않으므로 x_gate가 될 수 있다. 만약 입력이 (0 0 X) 라면 1번 조건은 만족하나 2번 조건을 만족하지 못하기 때문에 x_gate가 될 수 없다.

이러한 x_gate들을 가져와 프리미티브 고장패턴 값을 인가하여 고장을 검출한다. 만약 x_gate의 입력이 (1 X 1)라면 (1 0 1), (1 1 1)의 프리미티브 고장패턴을 검출할 수 있는 가능성이 있다. 먼저 X에 0의 값을 할당하고 이 입력선의 논리값을 만족시킬 수 있는 입력조합을 구하면 x_gate의 (1 0 1) 프리미티브 고장패턴은 검출되는 것이다. 만약 이미 정해진 논리값과 충돌이 발생하면 다른 프리미티브 고장패턴을 가져오게 된다. 현재 정해진 노드값 내에서 (1 0 1)을 만족시키지 못하면 다른 프리미티브 고장인 (1 1 1)을 가져와 X에 0 대신 1을 할당하여 이 입력선의 논리값을 만족시키는 입력조합을 구하면 된다. 만약 이 두 개의 프리미티브 고장패턴에 대한 패턴 생성이 실패하면, 더 이상 프리미티브 고장패턴이 존재하지 않으므로 회로내의 다른 x_gate를 선택하여 수행한다.

회로 내에 x_gate를 살펴보면 모든 입력선이 don't care 일 수도 있지만, 주 출력에서 가까운 게이트부터 선택하기 때문에 대부분의 게이트가 몇 개의 입력선은 정해져 있다. 그러므로 x_gate를 선택하여 컴팩션 알고리즘을 수행할 때 모든 입력을 만족시키는 것이 아니라 몇 개의 선만을 만족시키면 고장이 검출되므로 고장 검출이 용이하다.

그림 1은 x_gate의 예를 보여준다. M 게이트의 프리미티브 고장패턴 (0 1)을 검출하기 의하여 입력이 (0 X X)로 정해졌다. 이 정해진 입력을 논리 시뮬레이션을 수행하여 위에서 정의한 조건을 만족하는 게이트를 찾아보면 D, E, F, H, J, K, L이 x_gate가 된다. 앞에서 설명한 컴팩션 알고리즘을 수행하기 위하여 먼저 K 게이트를 선택하고, 2-입력 AND 게이트의 프리미티브 고장패턴들 중 현재 할당된 값과 충돌이 발생하지 않는 (0 1)을 가져온다. K 게이트의 한 입력은 0을 만족하고 있으므로 J 게이트의 출력 라인에 X값 대신 1을 할당하여 패턴을 생성한다. 이렇게 하나의 입력선만을 만족시키면 K 게이트의 프리미티브 고장패턴 중 하나는 쉽게 검출되는 것이다. 패턴이 생성되면 새로 정해진 주 입력선에 대하여 논리 시뮬레이션을 수행하는데, 이때 다른 x_gate의 don't care로 남아있던 입력선들에 값을 할당함으로 다시 x_gate를 선택하여 컴팩션 알고리즘을 수행할 때 모든 입력선에 대하여 고려하지 않아도 쉽게 고장을 검출할 수 있는 장점을 지닌다. 이렇게 하여 더 이상 x_gate가 존재하지 않으면 고장패턴 그레이딩을 수행한다.

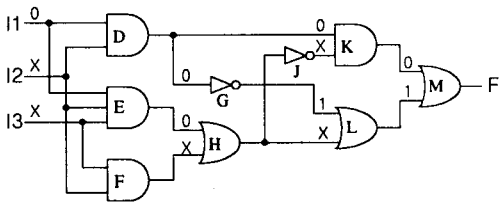

그림 1. x_gate의 예제

Fig. 1. An example of x_gate

(2) 동적 컴팩션 알고리즘

본 논문에서 제안된 동적 컴팩션 알고리즘의 각 단계에서의 수행은 다음과 같다.

단계 1 : 고장 집합에서 하나의 프리미티브 고장패턴을 가져와 초기 패턴을 생성.

단계 2 : 정해진 입력 패턴에 대하여 논리 시뮬레이션을 수행.

단계 3 : 각 노드들의 값을 조사하여 주 출력에 가까운 x_gate를 선택.

단계 4 : 현재 정해진 입력값과 충돌이 발생하지 않는 프리미티브 고장패턴들 중 검출되지 않은 프리미티브 고장패턴을 선택.

단계 5 : 함의조작과 백트레이스 과정 등을 수행하여 테스트패턴을 생성. 이러한 과정 중에 이미 정해진 노드의 논리값과 충돌이 발생하면 4번으로 돌아가 게이트의 다른 프리미티브 고장패턴을 선택하여 다시 수행하고, 만약 모든 프리미티브 고장패턴에 대하여 테스트패턴이 생성되지 못하면 3번으로 돌아가 다른 x_gate를 가져와 수행.

단계 6 : 임시 패턴 생성. 패턴이 생성되면 2번 단계로 돌아감.

단계 7 : 더 이상 x_gate가 존재하지 않으면 고장패턴 그레이딩을 수행.

단계 8 : 현재까지 정해진 주 입력값들을 테스트패턴으로 등록.

단계 1에서 생성된 초기 패턴은 많은 don't care를 포함하고 있는데, 이 초기 패턴을 가지고 단계 2에서 논리 시뮬레이션을 수행하여 회로내 각 노드의 값을 할당한다. 단계 3에서는 더 많은 고장패턴 그레이딩을 수행하기 위하여 주 출력에서부터 게이트의 입력값을 조사하여 don't care를 포함하는 x_gate를 선택한다. 단계 4에서는 x_gate의 프리미티브 고장패턴들 중 검출되지 않은 프리미티브 고장패턴을 가져와 단계 5에서 패턴을 생성한다. 만약 패턴 생성 단계에서 함의 조작 수행 중에 이미 정해진 노드값과 충돌이 발생하면 이 x_gate의 프리미티브 고장패턴은 이미 정해진 입력 조합 내에서는 검출되지 못하므로 패턴 생성을 포기하고 다른 프리미티브 고장패턴을 가져와서 수행한다. 함의 조작 단계 후 발생한 오브젝티브에 대하여 백트레이스를 수행하는데 이미 정해진 회로내의 논리값과 충돌이 발생하면 백트랙을 수행하고, 만약 백트랙 제한 회수 내에 고장을 검출하지 못하면 x_gate의 다른 프리미티브 고장패턴을 가져온다. 만약 x_gate의 모든 프리미티브 고장패턴에 대하여 패턴이 생성되지 않으면, 다른 x_gate를 가져와 수행한다. 이미 정해진 논리값에는 충돌 없이 패턴이 생성되면, 다시 단계 2로 돌아가 정해진 입력에 대하여 논리 시뮬레이션을 수행한다. 컴팩션 알고리즘의 수행 중 단계 5에서 충돌이 발생하여도 프리미티브 고장패턴을 중복고장이나 실패고장으로 저장하지 않는다. 충돌이 발생한 프리미티브 고장패턴들은 나중에 다른 게이트에 대한 컴팩션 알고리즘 수행 중에 검출되거나 대상 고장으로 선택되어 검출되어진다. 이러한 과정을 반복적으로 수행하면서 검출 가능한 고장을 계속 검출한다. 더 이상 x_gate가 존재하지 않으면 단계 7에서는 회로 전체의 각 게이트의 입력을 조사하여 생성된 패턴으로서 검출된 모든 고장을 고장 집합에서 제거한 다음 단계 8에서 테스트패턴으로 등록을 한다. 이렇게 함으로써 하나의 패턴으로 검출 가능한 모든 고장을 검출할 수 있다.

3. 백트레이스의 개선

백트레이스는 테스트패턴 생성 과정 중 함의조작에 의해 발생한 오브젝티브를 만족시키기 위하여 각 게이트의 입력을 선택하면서 주 입력을 찾아가는 단계이다. 백트레이스를 수행할 때 게이트의 여러 입력선 중 하나의 입력을 선택하기 위하여 제어도가 휴리스틱 조건으로 사용되는데, 백트랙 제한 회수 내에 고장을 검출할 수 있는 입력 패턴을 찾도록 도와준다.

(1) 난수에 의한 백트레이스

백트레이스를 수행하면서 게이트의 입력선들 중 하나를 선택할 때 회로 내의 제어가 어려운 입력선은 선택하지 않기 때문에 이 부분의 고장들은 프리

미티브 고장패턴 그레이딩 수행 중에 검출되지 않고 남아 있게 되고, 제어가 용이한 부분은 계속해서 반복적으로 같은 고장이 검출되는 결과를 가져온다. 그리고 할당되는 논리값이 같아짐으로 x_gate로 선택되어도 검출 가능한 프리미티브 고장패턴이 한정되어 컴팩션의 효과가 줄어든다.

예를 들어 그림 2에서 AND 게이트의 b 입력선은 제어가 쉽고 나머지 a, c 입력선은 b 입력선에 비하여 제어가 어렵다. 만약 AND 게이트의 출력선 f의 오브젝티브 값이 1이라면 입력선 a, b, c 모두 1의 값을 할당하면서 백트레이스를 수행하지만, 오브젝티브 값이 0일 경우에는 제어가 쉬운 b 입력선을 선택하여 백트레이스를 수행한다. 결과적으로 a, c 입력선에는 논리값 0이 할당되지 못하고 입력선 b에는 계속해서 0이 할당되므로 b 입력선 앞부분에 연결된 게이트들은 같은 프리미티브 고장패턴이 계속해서 고장패턴 그레이딩으로 검출되고, 입력선 a와 c는 선택되지 않으므로 don't care로 남아있게 된다. 컴팩션 알고리즘의 수행중 이 AND 게이트는 x_gate로 선택되어 입력값 (X 0 X)과 충돌이 발생하지 않는 프리미티브 고장패턴 (1 0 1)만이 선택되어 검출된다. 그러므로 나머지 프리미티브 고장패턴 (0 1 1), (1 1 0)은 모든 입력이 don't care 값을 가지거나 이 게이트가 대상 고장으로 선택되기 전에는 검출되지 못하고 남아있게 된다. 즉 이 AND 게이트를 대상으로 고장을 검출하기 전까지는 a, b, c의 입력선에 연결된 게이트들이 일정한 논리값을 가지게 되어 컴팩션에 나쁜 영향을 미친다. 이러한 점을 개선하기 위하여 백트레이스 수행시에 제어도에 의존하여 입력선을 선택하지 않고, 난수를 발생시켜 임의로 입력선을 선택하여 주 입력을 찾아가게 한다. 제어가 어려워서 선택하지 않는 경로도 난수에 의해 임의로 선택되어 지나면서 패턴을 생성하게 되어, 이러한 부분의 고장들이 우연히 검출되는 확률을 높이고 고장패턴 그레이딩에 의하여 제거할 수 있다. 또한 don't care로 남아있는 부분은 x_gate로 선택되어 고장이 검출될 수 있다.

그림 2에서 출력값 0을 만족시키기 위하여 입력을 선택할 때 난수에 의해서 a가 선택된다면 컴팩션 알고리즘의 수행 중 (0 X X)와 충돌이 없는 (0 1 1) 프리미티브 고장패턴이 검출될 수 있으며,

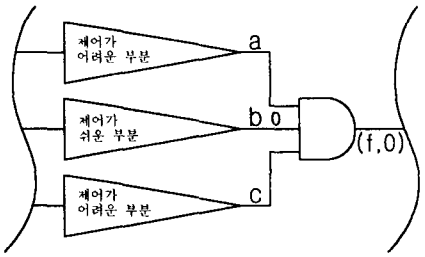

그림 2. 제어도
Fig. 2. Controllability

a에 연결된 다른 게이트들도 다른 논리값이 할당되어 컴팩션의 효과를 높일 수 있게 된다. 이렇게 함으로써 제어가 어려운 부분에 대하여 추가의 패턴을 생성하지 않게 되고, 제어가 용이한 부분에 일정한 노드값이 할당되지 않아 다른 프리미티브 고장패턴에 대하여 컴팩션 알고리즘을 적용할 수 있으므로 적은 수의 패턴을 얻을 수 있고 패턴 생성 시간도 줄일 수 있다.

(2) 제어도의 순환

제어도는 휴리스틱 방법으로 백트레이스를 수행하며 주 입력을 찾아갈 때 백트랙 회수를 줄이기 위하여 사용된다. 그러므로 제어도를 참조하지 않고 난수를 참조하여 백트레이스를 수행하는 것은 본 논문에서 제안한 컴팩션 알고리즘에는 효과적이나 테스트패턴을 생성하는데는 적합하지 못하다. 난수에 의해 임의로 주 입력을 찾아감으로 고장을 검출할 수 있는 입력 조합을 만들기가 어렵고, 큰 회로에서는 탐색 공간이 많으므로 충돌이 많이 발생하여 백트랙 제한 회수 내에 고장을 검출하지 못한다. 이를 개선하기 위하여 난수에 의한 백트레이스가 백트랙 제한 회수 내에 고장을 검출하지 못하면, 제어도를 도입하여 백트레이스를 수행한다.

본 논문에서 제안하는 컴팩션 알고리즘의 효과를 높이기 위하여 먼저 난수를 발생시켜 백트레이스 수행하고, 백트랙 제한 회수 내에 고장을 검출하지 못하면 4가지의 제어도를 사용하게 된다. 이때 한 가지의 제어도만 참조하게 되면 앞에서 설명한 난수에 의한 백트레이스의 장점을 살리지 못하므로, 난수에 의한 백트레이스 방법과 4가지의 제어도를 순차적으로 교대로 참조하게 한다. 각 제어도는 계

산하는 방법이 차이가 있으므로 각 노드에 대하여 다른 값을 가지므로 주 입력을 찾아갈 경우 제어도를 사용하여도 회로내의 여러 가지 경로를 통과하면서 패턴을 생성하게 되어 그레이딩 되는 프리미티브 고장패턴이 많아질 수 있으며, 따라서 컴팩션의 효과도 높일 수 있다.

먼저 하나의 오브젝티브에 대하여 난수에 의한 방법으로 백트레이스를 수행하고 만약 백트랙 제한 회수 내에 오브젝티브를 만족시키지 못하면 FAN[14], COP[15], CAMELOT[16], SCOAP[17] 순으로 제어도를 참조하게 된다. 만약 FAN에서 오브젝티브가 만족되어지면 계속 사용하다가, FAN이 패턴 생성에 실패하면 COP로 전환된다. 이렇게 하여 5가지의 방법을 순차적으로 전환시키면서 백트레이스를 수행하는데, 이 5가지의 방법을 모두 사용하여도 고장을 검출하지 못하면 실패고장으로 처리한다.

Ⅲ. 모의 실험 및 결과

본 논문에서 제안한 동적 컴팩션 알고리즘을 사용하여 ATPG(automatic test pattern generation)를 C 언어로 구현하였고, ISCAS 85 벤치마크 회로에 대하여 실험하였다.

표 2는 백트랙 제한 회수를 100회로 하여 벤치마크 회로에 대한 실험 결과를 나타낸다. 여기서 총 고장패턴수(Nfp)는 회로내 각 게이트의 프리미티브 고장패턴을 합한 것이고, 실패 고장패턴수(Nffp)와 중복 고장패턴수(Nrfp)는 백트랙 제한 회수 내에 검출되지 않은 프리미티브 고장패턴수와 중복 회로내에서 만족될 수 없는 프리미티브 고장패턴수를 각각 나타낸다. 순수 고장 검출율(FCp)은 전체 고장패턴수에 대한 검출된 고장패턴수의 백분율을 나타내며, 고장 검출율(FC)은 전체 고장패턴수에 대한 검출된 고장패턴수와 중복 고장패턴수를 합한 것의 백분율을 나타낸다. 평균 시간(Tav)은 하나의 고장패턴 당 소비되는 컴퓨터 사용시간이며, 평균 백트랙 수(NBav)는 테스트패턴 생성 동안에 발생한 총 백트랙 수에 대한 총 고장패턴으로 계산되어 하나의 고장패턴을 생성할 때에 발생한 백트랙 회수를 나타낸다. 그리고 Ntp는 테스트패턴수를 나타낸다.

표 2. 모의 실험 결과(백트랙 제한=100)
Table 2. Simulation results(Backtrack limit=100)

	Nfp	Nffp	Nrfp	Ntp	FCp	FC	Tav(ms)	NBav
c432	496	0	18	30	96.37	100	3.23	0.22
c499	610	30	0	50	95.08	95.08	16.07	32.03
c880	1112	0	0	18	100	100	3.14	0.01
c1355	1610	0	0	87	100	100	5.34	1.46
c1908	2377	0	0	109	100	100	4.80	0.04
c2670	3268	11	17	22	99.14	99.66	6.98	1.79
c3540	4605	29	17	43	98.91	99.37	17.74	6.71
c5315	6693	0	2	33	99.96	100	6.89	0.54
c6288	7216	0	35	28	99.51	100	4.98	0.39
c7552	9656	0	20	36	99.79	100	5.53	0.10

표 3은 난수와 네 개의 제어도에 의한 백트레이스를 다양하게 바꾸어 실험한 결과이다. R은 난수에 의한 백트레이스를 나타내며, S는 scoap, C는 camalot, F는 fan, CO는 cop를 나타낸다. 순서를 다양하게 백트레이스를 수행하여도 생성된 패턴의 수는 큰 차이가 없이 나타났다. 패턴수의 차이는 난수나 제어도에 따라 회로내의 다른 게이트들을 지나서 패턴이 생성됨으로 고장패턴 그레이딩되는 게이트와 개수가 달라지며 선택되는 x_gate도 바뀌기 때문이다.

표 3. 제어도 순서의 변화에 대한 패턴 수
Table 3. The Numbers of Patterns for the controllability sequences

	R,S,C,F,CO	R,C,F,S,CO	R,CO,F,S,C	R,F,CO,C,S
c432	30	30	30	30
c499	50	51	49	50
c880	20	18	19	18
c1355	87	87	87	87
c1908	114	112	117	109
c2670	22	22	22	22
c3540	38	45	40	43
c5315	31	30	31	33
c6288	28	29	29	28
c7552	37	37	38	36

표 4에서 NO_comp.은 컴팩션을 수행하지 않고 프리미티브 고장패턴 그레이딩만 수행하여 생성된 테스트패턴수를 나타내며, Simple S_comp.은 고장에 대하여 테스트패턴을 생성 후 논리값을 조사하여 결합될 수 있는 테스트패턴을 생성 순서대로 결합한 간단한 정적 컴팩션의 결과이다. D_comp.의 CONT.는 4개의 제어도만을 사용하여 생성된 패턴수이고, RAND.는 난수에 의한 백트레이스를 추가하여 수행한 결과이다. CONT.의 경우에 작은 회로 대해서는 S_comp.에 비하여 약 20~30%정도 줄었지만, 회로가 커질수록 45% 이상 줄어든 것을 알 수 있다. RAND.의 경우에는 작은 회로에 대해서 패턴이 늘어났지만, 큰 회로에서는 평균 15% 이상 패턴이 줄어들었다. c432 회로에 대하여 패턴수가 늘어난 것을 볼 수 있는데, 이것은 난수에 의한 백트레이스의 효과를 나타내기 전에 모든 고장이 검출됨으로서 본 논문에서 제안한 컴팩션 알고리즘의 성능을 더 높이지 못하기 때문이다. 그러나 c880 이상의 회로에서는 테스트패턴 수가 줄어드는 것을 확인할 수 있다.

표 4. NO_comp., S_comp.와 D_comp에 대한 테스트패턴 수

Table 4. The Numbers of Test patterns for NO_comp., S_comp. and D_comp.

	NO_comp.	Simple S_comp.	D_comp.	
			CONT.	RAND.
c432	141	30	21	30
c499	194	48	44	50
c880	501	28	21	18
c1355	380	96	93	87
c1908	512	147	123	109
c2670	1040	43	33	22
c3540	1036	92	53	43
c5315	2137	69	37	33
c6288	313	67	34	28
c7552	2049	114	45	36

Ⅳ. 결 론

본 논문에서는 테스트패턴수를 줄이기 위한 동적 컴팩션 알고리즘을 제안하고, IDDQ 테스팅 방식으로 CMOS VLSI 회로의 게이트 내에 존재하는 모든 브리징 고장과 GOS 고장을 검출할 수 있는 자동 테스트패턴 생성기를 구현하였다. 본 논문에서 제안하는 동적 컴팩션 알고리즘은 하나의 고장에 대하여 패턴을 생성 후 x_gate에 대하여 추가로 고장을 검출하는 방법으로, 더 이상 결합할 수 없는 적은 수의 테스트패턴을 얻을 수 있었다. 그리고 네 가지의 제어도를 참조하여 수행하던 백트레이스 방법에서 난수에 의한 백트레이스를 추가하였고, 이 방법들을 순환시킴으로써 프리미티브 고장패턴 그레이딩을 더 많이 수행할 수 있었으며 컴팩션의 성능도 향상되었다. 테스트패턴을 생성 후 간단한 정적 컴팩션을 수행한 패턴수에 비해 동적 컴팩션을 수행한 패턴수가 작은 회로에서는 c1355를 제외하고는 평균 20% 이상 줄었으며 c3540 보다 큰 회로에서는 평균 45% 이상 줄었다. 또한 네 가지의 제어도만을 사용한 방법보다 난수에 의한 백트레이스와 제어도를 순환시킨 방법이 패턴수를 평균 15% 이상 더 줄일 수 있었다.

참고문헌

[1] W. Maly and P. "Nigh, Built-in Current Testing - Feasibility Study," *Proc. 1988 Int. Conf. on Computer-Aided Design*, pp. 340-343, 1988.

[2] Udo Mahlstedt, Jurgen Alt, and Matthias Heinitz, "CURRENT : A Test Generation System for IDDQ Testing," *IEEE VLSI Test Symposium*, pp. 317-323, 1995.

[3] Marek Syrzycki, "Modeling of Gate Oxide Shorts in MOS Transistors," *IEEE Trans. Computer-Aided Design*, Vol. 8, No. 3, pp. 193-202, Mar. 1990,

[4] Thomas M. Storey and Wojciech Maly, "CMOS Bridging Fault Detection," *Proc. Int. Test Conf.*, pp. 842-851, Sept. 1990.

[5] B. Krishnamurthy and S. B. Akers, "On the complexity of estimating the size of a test set," *IEEE Trans. Comput.*, vol. C-33, no. 8,

pp. 750-753, Aug. 1984.

[6] M. Schulz, E. Trischler, and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. on CAD.*, pp. 126-137, Jan. 1988.

[7] L. N. Reddy, I. Plmeranz, K. Kinoshita, and S. M. Reddy, "ROTCO: A Reverse Order Test Compaction Technique," *1992 IEEE EURO-ASIC Conf.*, pp. 189-194, Sept. 1992.

[8] J. S. Chang and C. S. Lin, "Test Set Compaction for Combinational Circuit," *IEEE Tran. on CAD.*, vol. 14, no. 11, pp. 1370-1378, Nov. 1995.

[9] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," *30th DAC*, pp. 102-106, June 1993.

[10] P. Goel and B. C. Rosales, "Test Generation and Dynamic Compaction of Tests," *Proc. IEEE Test Conf., Cherry Hill, NJ, pp.* 189-192, 1979.

[11] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," *1991 ITC.*, pp. 194-203, Oct. 1991.

[12] Sheldon B. Akers and Christie Joseph, "On the Role of Independent Fault Sets in the Generation of Minimal Test Sets," *IEEE ITC.* pp. 1100-1107, 1987.

[13] 김강철, 류진수, 한석붕, "CMOS VLSI의 IDDQ 테스팅을 위한 ATPG 구현," *대한전자공학회 논문지*, 제 33권 A편 제 3호, pp. 176-186, 1996년 3월,

[14] Fujiwara, H. and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. Comput.*, vol. C-32, : 1137-1144, 1983.

[15] Brglez, F., P. Pownall, and R. Hum, "Application of testability analysis: From ATPG to critical delay path tracing," *Proc. 1984 Int. Test Conf.*, pp. 705-712, Oct. 1984.

[16] Bennetts, R. G., C. M. Maunder, and G. D. Robinson, "CAMELOT: A Computer-Aided Measure for Logic Testability," *Proc. Int. Conf., on Circuits and Computers,* pp. 1162-1165, Oct. 1980.

[17] Goldstein, L. H. and E. L. Thigpen, "SCOAP: Sandia Controllability / Observability Analysis Program," *Proc. 17th Design Automation Conf.*, pp. 190-196, June 1980.

김 강 철(Kang-Chul Kim)
1981년 2월 서강대학교 전자공학과 졸업(공학사)
1983년 2월 서강대학교 대학원 전자공학과 졸업(공학석사)
1996년 8월 경상대학교 대학원 전자공학과 졸업(공학박사)
1983년 3월 - 1989년 6월 한국전자통신연구소
1989년 7월 - 1990년 2월 삼성종합기술원
1990년 3월 - 1997년 8월 진주산업대학교 전자계산학과
1997년 9월 - 현재 여수대학교 컴퓨터공학과
*주관심분야 : VLSI 설계 및 테스팅, 전력전자.

송 근 호(Geun-Ho Song)
1995년 2월 동아대학교 전자공학과 졸업(공학사)
1997년 2월 경상대학교 전자공학과 졸업(공학석사)
1997년 3월 ~ 경상대학교 대학원 전자공학과 박사과정
*주관심 분야 : VLSI Design, VLSI Testing, Analog Testing, Design for Testability, IDDQ Testing, ATPG 등임

한 석 붕(Seok-Bung Han)

1982년 2월 한양대학교 전자공학과 졸업(공학사).

1984년 2월 한양대학교 대학원 전자공학과 졸업(공학석사)

1988년 2월 한양대학교 대학원 전자공학과 졸업(공학박사)

1988년 3월 경상대학교 전자공학과 전임강사

1992년 1월 ~ 1993년 1월 Stanford University department of Electrical Engineering and Computer Science POST DOC.

1993년 4월 ~ 1999년 4월 현재 경상대학교 전자공학과 교수

*주관심 분야 : VLSI Design(CMOS 및 BiCMOS), VLSI Testing, Analog Testing, CMOS 칩의 Reliability Testing, Design for Testability, Fault Tolerance Computing System, VLSI/CAD, ASIC Design, ATPG 등임