IDDQ 테스트를 위한 고장 시뮬레이터

A Fault Simulator for IDDQ Testing

배 성 환*, 김 대 익**, 이 창 기***, 전 병 실****
(Sung Hwan Bae*, Dae Ik Kim**, Chang Ki Lee***, Byoung Sil Chon****)

※본 연구는 서울대학교 반도체공동연구소의 교육부 반도체분야 학술연구조성비(과제번호 : ISRC 97-E-2033)에 의해 수행되었습니다.

요 약

CMOS 기술이 발달됨으로써 고집적화에 따른 합선고장이 상대적으로 증가하고 있다. IDDQ 테스트는 기능테스트로 검출하기 어려운 합선고장을 효율적으로 검출하여 회로의 신뢰성을 향상시키는 기법이다.

본 논문에서는 테스트 대상 논리회로의 각 게이트 내부에서 발생 가능한 합선고장에 대한 시뮬레이션을 수행하기 위한 IDDQ 테스트용 고장 시뮬레이터를 개발하였다.

ABSTRACT

As CMOS technologies have been rapidly developed, bridging faults have been relatively increased. IDDQ testing is a current testing methodology which can enhance reliability of the circuit since it efficiently detects bridging faults that are difficult to detect by functional testing.

In this paper we consider internal bridging faults occurred in each gate of logic circuits under test and finally develop a fault simulator for IDDQ testing to detect assumed bridging faults.

I. 서 론

반도체 칩이 고집적화 될수록 회로 내부에 정의한 고장의 존재를 검사하는데 요구되는 테스트 시간과 비용이 급속도로 증가하게 된다.

TTL과 nMOS 기술을 이용할 때에는 고착고장(Stuck-At Fault : SAF) 모델이 사용되었다. 고착고장은 몇가지 중대한 결함을 표현하는데 문제점이 있었지만 고장모델로서 널리 이용되었다. 그러나 최근 CMOS 기술이 발달됨으로써 고집적화에 따른 합선고장(Bridging Fault : BF)이 상대적으로 증가하게 되었다. 고착고장은 항상 논리 값으로 고장상태를 정의하였으나 합선고장은 많은 경우에 있어서 고장상태를 항상 논리 값으로 결정지을 수 없게 되어 전압 수준을 점검하는 기능테스트로는 검출할 수 없다[1].

이러한 고장모델을 검출할 수 있는 효과적인 테스트로 IDDQ 테스트가 제안되었다[1-3]. IDDQ 테스트는 CMOS 회로에서 VDD와 GND 사이의 전류 흐름을 검사하여 고장 및 결함의 존재를 점검하는 기법이다.

기존의 여러 논문에서 합선고장을 검출하기 위해 IDDQ 테스트의 패턴으로 고착고장을 위한 테스트 패턴의 사용에 대한 연구를 해 왔으나 많은 합선고장의 검출을 보장하지 못하였다[4-6].

본 논문에서는 테스트 대상 논리회로의 각 게이트 내부에서 발생 가능한 합선 고장에 대한 시뮬레이션을 위한 IDDQ 테스트용 고장 시뮬레이터를 개발하였다. 본 논문의 구성은 가정한 합선고장과 이를 테스트 할 수 있는 패턴에 관해서 설명하고 가정한 합선고장을 시뮬레이션 할 수 있는 고장 시뮬레이터의 구현에 대하여 살펴본 후 결론을 맺는다.

II. IDDQ 테스트를 위한 합선고장 모델

2.1 합선고장 모델과 테스트 패턴

VLSI 회로 내에 발생되는 결함은 일정 기간에는 회로의 동작에 전혀 악영향을 주지 않지만 전기적, 환경적인 스트레스로 인하여 사용 중에 고장을 발생시킬 수 있다. 이것은 칩 혹은 사용중인 시스템의 신뢰성을 절하시킬 수 있으므로 조기에 이러한 결함들을 검출해야만 한다.

완벽한 합선고장을 모델링하기 위해서는 테스트 대상이 되는 회로의 스위칭 수준에서 전위가 다른 모든 노드 사이의 모든 가능한 단락을 고려해야 하지만 이를 위한 시뮬레이션은 매우 복잡하고 많은 시간이 필요하게 된다. 그리고 일반적으로 합선고장은 게이트의 외부에서 발생되는 게이트 외부단락(inter-gate short)과 내부에서 발생되는 게이트 내부단락(intra-gate short)으로 분류할 수 있다[6]. 임의의 테스트 대상 회로에 있어서 외부단락이 발생할 수 있는 경우 수는 회로의 게이트 수가 증가할수록 기하

* 한려대학교 정보통신학과
** 전북대학교 전기전자회로합성연구소
*** 서남대학교 전산정보학과
**** 전북대학교 전기전자제어공학부
접수일자 : 1998년 10월 30일

급수적으로 늘어나게 된다.

따라서 본 논문에서는 이러한 문제점을 피하기 위해 하나의 게이트 내에서 발생하는 내부단락을 고려하였다.

회로내의 각 노드는 특성에 따라 다음과 같이 세 종류로 분류된다.

1) 상수노드(constant node)

상수의 논리 값을 갖는 노드(VDD 혹은 GND)

2) 입력노드(input node)

임의의 논리 값(1 또는 0)으로 설정해 놓을 수 있는 게이트의 입력노드

3) 종속노드(dependent node)

논리 값이 게이트의 입력 신호에 종속되는 노드(출력노드, 내부노드)

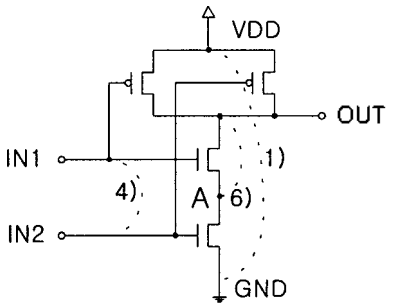

그림 1. 2개의 입력단을 갖는 CMOS NAND 게이트
Fig. 1. 2-Input CMOS NAND gate.

그림 1에서 상수노드는 VDD와 GND로 표시된 곳이며 입력노드는 IN1과 IN2로 표시되었고 종속노드는 출력단인 OUT와 직렬로 연결된 두 개의 nMOS 사이의 A로 표시된 점이다.

본 논문에서는 스위칭 수준의 회로 내에서 분류된 세 종류의 노드들 사이에 발생할 수 있는 모든 합선고장을 고려하였다.

따라서 가정한 단락은 노드들의 모든 가능한 조합으로 이루어져 다음과 같이 여섯 종류의 단락 형태로 나누어진다.

1) 상수전위를 갖는 두 노드 사이의 단락

2) 상수노드와 입력노드 사이의 단락

3) 상수노드와 종속노드 사이의 단락

4) 두 입력노드 사이의 단락

5) 입력노드와 종속노드 사이의 단락

6) 두 종속노드 사이의 단락

위 단락의 종류는 MOS의 드레인, 게이트, 소오스 세 단자 사이에서 발생 가능한 단락 형태를 포함하고 있음을 쉽게 알 수 있다. 그림 1에 단락의 종류 1), 4), 6)을 보여주고 있다.

각 단락에 따라 요구되는 테스트 패턴의 생성문제를 살펴보기로 하자. 종류 1)의 단락에 있어서 상수전위는 오직 VDD와 GND이 존재하므로 VDD-GND의 단락을 위한 별도의 테스트 패턴을 요구하지 않는다. 그러나 다른 종류의 단락을 검출하기 위해서는 두 노드 사이에 VDD-GND 경로를 만들어주기 위해 두 노드의 논리 값이 서로 상반되는 값을 갖도록 입력 패턴을 인가해 주어야 한다.

예를 들어 그림 1에서 4)의 단락을 검출하기 위해서는 IN1의 입력 값을 1(0)로 설정해 놓고 IN2의 입력 값을 0(1)으로 인가시켜 주면 IN1-IN2 사이에 전류 경로가 발생된다. 그리고 6)의 단락을 검출하기 위해서는 노드 OUT와 A 사이에 전류 경로를 생성시켜야하므로 노드 OUT를 1 값으로 만들기 위해 IN1의 입력에 0 값을 인가시켜 준다. 그리고 노드 A의 논리 값을 0으로 설정하기 위해 IN2의 입력에 1 값을 입력한다. 따라서 단락 6)을 검출하는데 요구되는 테스트 패턴은 (IN1, IN2) = (0, 1)가 된다.

테스트 패턴을 발생시켜 주기 위해 다음과 같은 두 단계의 절차를 수행한다.

step 1> 테스트 대상 회로를 구성하고 있는 **원시 게이트를 회로 수준으로 추출**하여 앞에서 정의한 6가지 형태를 갖는 모든 단락을 검출할 수 있도록 게이트의 입력에 인가할 테스트 패턴을 찾아내어 **각 게이트 종류별로 라이브러리를 구축**한다.

step 2> 테스트 대상 회로에 대하여 모든 게이트를 대상으로 하여 각 게이트에 대한 모든 단락을 검출할 수 있는 패턴이 인가될 수 있도록 **전체 회로의 주입력단에 인가할 테스트 패턴을 생성**한다. 또한 이 과정에서 고려한 게이트가 아닌 다른 게이트에 고장 테스트패턴이 저절로 발생되면 추후에 그 게이트에 대한 테스트 패턴을 발생시킬 때 저절로 발생된 고장 테스트 패턴은 제외시킨다.

Step 1에서 게이트에 대한 라이브러리 구축의 예로써 그림 2의 2개의 입력단을 갖는 CMOS NOR 게이트에 대해 살펴보면 다음과 같은 단계를 수행한다.

1) 그림 2의 2개의 입력단을 갖는 NOR 게이트의 노드를 종류별로 다음과 같이 나눈다.

- 상수노드 : 1 (VDD), 6 (GND)
- 입력노드 : 2 (IN1), 4 (IN2)
- 종속노드 : 3 (internal node), 5 (OUT)

2) 각 노드 간에 이루어질 수 있는 가능한 모든 단락을 조사한다.

여기에서는 노드 6개중에 2개씩 짝을 이루는 형태이므로 6C2가 되어 모두 15개의 단락이 존재한다. 그리고 노드 사이의 단락을 표시하기 위해 예로써 노드 1과 2 사이의 단락을 S12로 정의한다.

3) IN1과 IN2에 인가하여 각 단락을 검출할 수 있는 테스트 패턴을 생성시키기 위해 표 2와 같은 결정표를 작성한다. 여기에서 가장 중요한 점은 단락을 IDDQ 테스트로 검출하기 위해서는 가정한 두 노드 사이에 VDD-GND 경로가 형성되어야함으로 두 노드에 상반된 값(1 ↔ 0)을 갖도록 해 주어야 한다.

표 1에서 음영으로 표시된 곳을 모두 검출할 수 있는 입력 패턴을 찾아내어야 한다. 특히 진하게 표시된 곳은 반드시 왼쪽의 입력 값을 인가해야 검출할 수 있는 부분이기 때문에 입력 패턴 (00, 01, 10)은 꼭 선택해야 한다.

이 세 가지의 입력 패턴을 서로 중첩시켜 놓으면 모든 단락을 검출할 수 있음을 알 수 있다. 따라서 2개의 입력단을 갖는 NOR 게이트의 모든 내부단락을 검출하는 패턴은 (IN1, IN2) = (00, 01, 10)이 된다.

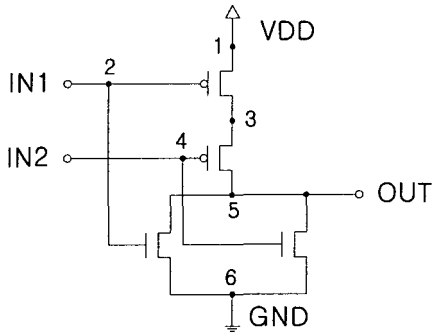

그림 2. 2개의 입력단을 갖는 CMOS NOR 게이트
Fig. 2. 2-Input CMOS NOR gate.

이와 같은 형태로 테스트 대상 회로를 구성하고 있는 모든 원시 게이트에 대해 요구되는 테스트 패턴을 구하여 라이브러리를 구축한다.

표 1. 테스트 패턴 생성을 위한 결정표
Table 1. Decision table for test pattern generation.

IN1IN2	00	01	10	11
S12	1	1		
S13				
S14	1		1	
S15		1	1	1
S16	X	X	X	X
S23	1	1		
S24		1	1	
S25	1		1	1
S26			1	1
S34				
S35				
S36	1	1		
S45	1	1		1
S46		1		1
S56				

다음으로 step 2에 대한 예를 살펴보기로 하자.

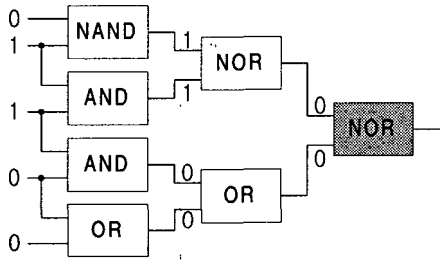

그림 3. 테스트 패턴 생성 예제 회로
Fig. 3. Test pattern generation example circuit.

그림 3에서 음영으로 표시된 게이트의 고장 패턴 입력을 발생시켜 주는 수위 입력단에서 인가되는 테스트 패턴을 생성하는 문제를 논의한다.

그림 3에서 음영으로 표시된 NOR 게이트의 입력에 (0, 0)를 인가하여 내부단락을 검출하고자 한다. 따라서 이런 조건을 만족시키기 위한 테스트 패턴을 발생시키기 위해 NOR 게이트와 연결된 앞단의 게이트들의 상태를 결정해야 한다. 결국에는 수위 입력단까지 거슬러 올라가 그림 3과 같이 테스트 패턴을 생성시켜 준다. 이런 과정에서 다른 게이트의 내부단락을 검출할 수 있는 고장 입력 패턴이 발생되면 현재 구해진 테스트 패턴으로 대신해 준다. 즉, 음영으로 처리된 NOR 게이트의 입력에 연결된 OR 게이트에 대해 (0, 0)로 검출할 수 있는 단락은 현재 생성된 테스트 패턴으로 사용하면 된다.

그리고 추후에 OR 게이트 내부단락을 검출하기 위한 테스트 패턴을 생성할 때 고장 입력 패턴 (0, 0), (0, 1), (1, 0) 중에서 (0, 0)를 제외한 나머지 패턴을 만들어 주는 테스트 패턴을 찾으면 된다.

III. 고장 시뮬레이터

고장 시뮬레이터는 임의의 테스트 패턴을 인가하여 테스트 대상 회로에서 정의한 고장을 얼마만큼 검출할 수 있는지를 시뮬레이션 해주는 툴이다.

본 논문은 테스트 대상회로의 게이트 내부에서 발생하는 합성고장을 효과적으로 검출하는 IDDQ 테스트 방식에 사용하기 위한 고장 시뮬레이터를 개발한다. 그림 4에 고장 시뮬레이터의 전체적인 흐름도를 보인다.

개발한 고장 시뮬레이터의 구성 및 기능을 살펴보면 다음과 같다. PC의 Windows 95 기반에서 Delphi를 이용하여 사용자의 편의를 위해 GUI(Graphic User Interface) 방식으로 구현하였다.

고장 시뮬레이터는 각 게이트 내부에서 발생 가능한 합선고장 검출을 위해 필요한 테스트 대상 회로용 라이브러리를 회로에서 추출하여 구축하고 IDDQ 테스트시 테스트 대상회로의 각 게이트에 필요한 테스트 패턴을 위한 데이터 베이스를 구축한 후 임의의 테스트 패턴을 적용하여 고장 검출율 및 비 경험 합선고장 테스트 패턴을 추출하게 된다.

고장 시뮬레이터의 주 화면을 그림 5에 보여주고 있다. 그림 5에서 왼쪽 위쪽의 회로 읽기는 테스트 대상회로를 선택하는 버튼이며 이를 클릭 하면 회로 선택 대화상자가 나타나서 회로를 선택할 수 있으며 아래의 왼쪽 화면에 회로에 대한 정보를 나타낸다.

현재는 예로써 ISCAS 85의 C432를 대상으로 하였다. 그리고 인가된 테스트 패턴은 16-비트 LFSR을 이용하여 300개의 의사 무작위 패턴을 사용하였다.

그리고 주 메뉴의 목표 값보기를 선택하면 앞 절에서 모델링한 BF를 검출하기 위해 각 게이트의 입력에 인가되어야 할 패턴을 아래의 오른쪽 창에 보여주게 된다. 오른쪽 창의 ID는 각 게이트에 대한 입력단에 대한 고유 번

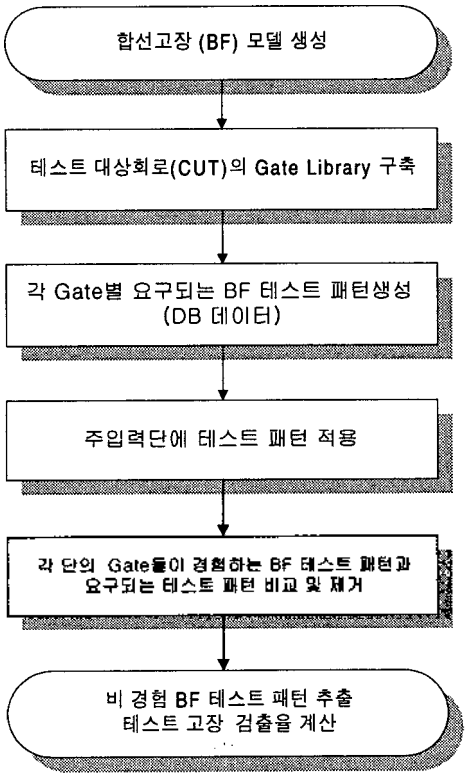

그림 4. 고장 시뮬레이터의 흐름도
Fig. 4. Flowchart of fault simulator.

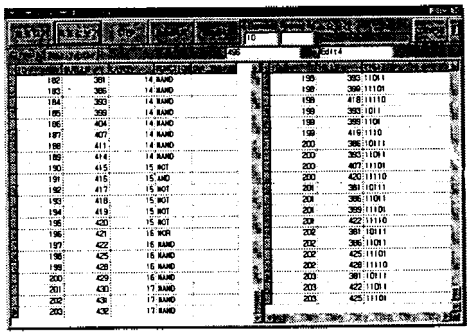

그림 5. 고장 시뮬레이터의 주화면
Fig. 5. Main window of fault simulator.

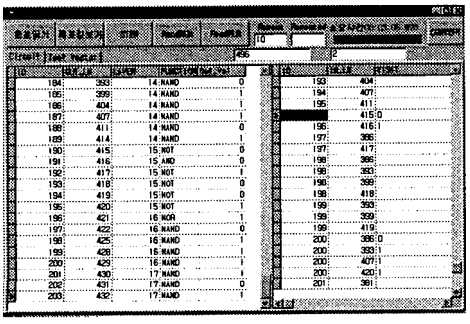

그림 6. ReadRUN 실행결과
Fig. 6. ReadRUN execution result.

호를 표시하며 IN_LN은 왼쪽 창에서 나타낸 출력 중 어떠한 것을 입력으로 인가되는지를 나타낸다. 그리고 VISIT는 게이트의 입력으로 인가되어야 할 패턴을 표시한다.

주 메뉴에서 ReadRUN은 선택한 테스트 패턴을 회로에 적용하는 것으로서 그림 6에는 테스트 패턴을 인가후의 테스트 결과를 보여주고 있다.

두 개의 창 위에 있는 숫자 496과 2는 각각 회로 내 가정한 고장을 검출하기 위해 필요한 총 테스트 패턴 수와 선택한 테스트 패턴을 인가한 후에 검출하지 못한 고장 테스트 패턴 수를 표현한다. 이 경우에는 고장 검출율이 $(496-2)/496\times100$된다. 고장을 검출하는데 영향을 준 테스트 패턴과 그렇지 못한 패턴을 추출해 준 결과 보기 위해 Test Vector라고 하는 버튼을 선택해야 하며 이를 그림 7에 보여 주었다. 왼쪽 창에 있는 테스트 패턴이 고장을 검출하는데 직접 영향을 준 패턴이고 오른쪽 창에는 고장 검출에 전혀 영향을 주지 못한 패턴을 나열해 놓았다.

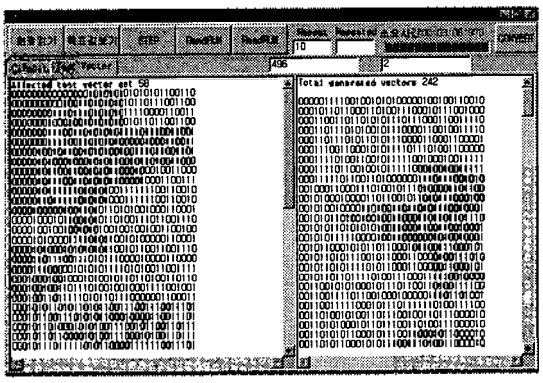

그림 7. 테스트 패턴 보기
Fig. 7. Test pattern generation example.

VI. 결 론

본 논문에서는 테스트 대상 논리회로의 각 게이트 내부에서 발생 가능한 합선고장에 대한 시뮬레이션을 수행하기 위한 IDDQ 테스트용 고장 시뮬레이터를 개발하였다. 개발한 고장 시뮬레이터의 구성 및 기능은 Windows 95 기반에서 Delphi를 이용하여 사용자의 편의를 위해 GUI (Graphic User Interface) 방식으로 구현하였다.

테스트 대상회로에 필요한 게이트 라이브러리, 각 게이트별 요구되는 테스트 패턴을 위한 데이터 베이스를 생성하여 기존의 기능 테스트 방식에서는 검출하기 힘든 게이트 내부의 합선 고장을 검출하기 위해 본 논문에서 제시된 효율적인 IDDQ 테스트 방식을 이용하여 높은 신뢰성을 요구하는 테스트를 행할 수 있다.

현재 개발한 고장 시뮬레이터의 효율성을 높이기 위해 시뮬레이션에 요구되는 시간을 줄이기 위한 연구가 지속적으로 수행되고 있다.

참 고 문 헌

1. R. Rajsuman, Iddq Testing for CMOS VLSI, Artech house, 1994.
2. 전병실 외, "메모리 테스트를 위한 BIST 기술," 전자공학회지, Vol.22, No.12, pp. 1442-1454, 1995.
3. R. Rajsuman, Digital Hardware Testing : Transistor-Level Fault Modeling and Testing, Artech house, 1992.
4. K. L. Kodandapani, et al., "Undectability of Bridging Faults and Validity of Stuck-at Fault Test Sets," IEEE Trans. on Comput., Jan. 1980, pp.55-59.
5. S. D. Millman, et al., "Detecting Bridging Faults with Stuck-at Test Sets," Proceeding of IEEE ITC, 1988, pp.773-783.
6. E. Isern, et al., "Test Generation with High Coverage for Quiescent Current Test of Bridging Faults in Combinational Circuits," Proceeding of IEEE ITC, 1993, pp.73-82.

▲배 성 환(Sung Hwan Bae)
한국음향학회지 15권 5호 참조
현재 : 한려대학교 정보통신학과 전임강사

▲김 대 익(Dae Ik Kim)
한국음향학회지 15권 5호 참조

▲이 창 기(Chang Ki Lee)
현재 : 서남대학교 전산정보학과 전임강사

▲전 병 실(Byoung Sil Chon)
한국음향학회지 15권 5호 참조