# 합선 고장을 위한 IDDQ 테스트 패턴 발생기의 구현

정회원 김 대 익*, 전 병 실**

# Implementation of IDDQ Test Pattern Generator for Bridging Faults

Dae-Ik Kim*, Byoung-Sil Chon** *Regular Members*

요 약

IDDQ 테스팅은 CMOS 회로에서 발생되는 여러 종류의 물리적 결함을 효율적으로 검출하는 테스팅 방식이다. 본 논문에서는 테스트 대상 회로의 게이트 내부에서 발생하는 단락을 고려하여, 이 결함을 검출하기 위한 테스트 패턴을 찾아 주는 IDDQ 테스트 패턴 발생기를 구현하였다.

테스트 패턴을 생성하기 위해 게이트 종류별로 모든 내부 단락을 검출하는 게이트 테스트 벡터를 찾아낸다. 그리고 10,000개의 무작위 패턴을 테스트 대상 회로에 인가하여 각 게이트에서 요구되는 테스트 벡터를 발생시켜 주면 유용한 테스트 패턴으로 저장한다. 입력된 패턴들이 모든 게이트 테스트 벡터를 발생시켜 주거나 10,000개의 패턴을 모두 인가했을 경우 테스트 패턴 발생 절차를 종료한다. ISCAS '85 벤치마크 회로에 대한 실험을 통하여 기존의 다른 방식보다 성능이 우수함을 보여주었다.

ABSTRACT

IDDQ testing is an effective testing method to detect various physical defects occurred in CMOS circuits. In this paper, we consider intra-gate shorts within circuit under test and implement IDDQ test pattern generator to find test patterns which detect considered defects.

In order to generate test patterns, gate test vectors which detect all intra-gate shorts have to be found by type of gates. Random test sets of 10,000 patterns are applied to circuit under test. If an applied pattern generates a required test vector of any gate, the pattern is saved as an available test pattern. When applied patterns generate all test vectors of all gates or 10,000 patterns are applied to circuit under test, procedure of test pattern generation is terminated. Experimental results for ISCAS'85 bench mark circuits show that its efficiency is more enhanced than that obtained by previously proposed methods.

## I. 서 론

CMOS 게이트는 nMOS 풀다운(pulldown) 블록과 pMOS 풀업(pullup) 블록으로 이루어져 있다. 고장이 없는 회로에서는 임의의 입력상태에 대하여 출력단이 VDD 또는 GND 중 한 노드만으로 연결되어 VDD와 GND 사이에 전류의 도통 경로가 형성되지 않는다. 따라서 이 경우의 정적 상태(steady state) 동안에는 수 nA정도의 무시할 만한 결합 누설 전류(junction 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 정적 상태 전류를 증가시켜 많은 전력을 소비하도록 야기 시킨다.[1]

* 전북대학교 전기전자회로합성연구소 ** 전북대학교 전자정보공학부
논문번호 : -, 접수일자 : 199 년 월 일

IDDQ 테스트(quiescent power supply current test)는 CMOS에서 발생 가능한 여러 종류의 물리적 결함을 효율적으로 검출할 수 있는 테스트 방식으로 알려져 있다. 즉, CMOS 회로의 정적 상태에서 VDD에 흐르는 전류의 양을 측정하여 결함의 유, 무를 검출하게 된다. IDDQ 테스트는 CMOS 기반 회로에서 물리적 결함을 40%~50% 정도 모델링할 수 있는 합선 고장(bridging fault)을 효율적으로 검출할 수 있다.[2] 합선고장은 두 선 혹은 그 이상의 선이 서로 의도하지 않게 연결되는 고장 형태이며 고착고장(stuck-at fault)으로 모델링되지 못하는 결함을 포함하고 있다.

기존의 전압 테스트에서 사용되는 합선고장의 모델은 wired-AND, wired-OR, 그리고 dominant 고장으로 구분된다. 그러나 CMOS 회로에서는 선간의 단락이 중간 전압(intermediate voltage)을 발생시키므로 wired 모델의 부적합함을 알 수 있다. IDDQ 테스트는 두 선 $l_1$과 $l_2$ 사이에 합선고장이 발생하였을 경우 $l_1$은 "1" 논리 값을 그리고 $l_2$는 "0" 논리 값을 갖도록 (혹은 그 반대로) 해 준다면 회로의 정적 상태에서도 VDD와 GND 사이에 전류의 경로가 생겨 많은 전류가 흐르게 되어 이 고장을 쉽게 검출할 수 있다.

합선고장을 유발시키는 단락은 내부단락(internal short)과 외부단락(external short)으로 나누어진다. 내부단락은 한 게이트 내의 내부 노드, 입력단, 출력단, 혹은 VDD와 GND와 같은 상수 노드 사이의 단락이며, 외부단락은 회로를 이루는 모든 논리 게이트의 입력단, 출력단, 또는 상수 노드 사이의 단락을 의미한다.

본 논문에서는 고집적화에 따른 주변 노드 사이의 합선, 게이트 옥사이드 두께의 감소 등으로 인한 게이트 내부단락을 고려하여 이 고장을 검출하는데 사용되는 IDDQ 테스트를 위한 패턴 발생기를 개발한다.

일반적으로 테스트를 수행하기 위해서는 회로에서 발생한 고장의 영향을 여기(excitation)시키기 위해 테스트 대상 회로의 주 입력단(primary input)에 적당한 입력 패턴을 인가하여 그 출력을 관찰해야 한다. 테스트 패턴을 발생시킬 때 고려해야 할 사항은 가정한 고장에 대한 검출률이 높아야 하며 패턴의 생성 시간이 짧아야 한다. 그리고 테스트에 소요되는 시간을 줄이기 위해 발생기에서 얻어진 테스트 패턴의 개수가 적어야 한다. 이러한 사항들은 테스트 비용에 직접적인 영향을 주기 때문이다.

본 논문에서는 [3]이 제안한 것과 같은 방식을 이용하여 IDDQ 테스트 패턴 발생기를 개발한다. 첫 번째로 기본 게이트(NAND, NOR, AND, OR, NOT, Buffer, EXOR, EXNOR)에 대하여 각 게이트에 대한 트랜지스터 구성도를 참조하여 게이트 내부의 모든 단락을 검출하기 위한 최적화된 테스트 벡터를 추출한다. 두 번째로 테스트 대상 회로에 인가되는 입력 패턴에 따라 결정되는 각 게이트의 입력 논리 상태를 앞에서 얻은 게이트 테스트 벡터와 비교하여 일치할 경우에는 인가된 입력 패턴을 유용한 테스트 패턴으로 저장한다.

이러한 방식으로 인가된 입력 패턴들이 회로내의 모든 게이트가 요구하는 모든 테스트 벡터를 만족시켰거나 또는 미리 정의해 놓은 수의 입력 패턴을 모두 인가한 경우에 패턴 발생 절차를 종료한다.
논문의 구성은 2장에서 고려한 고장 모델과 고장 검출에 요구되는 게이트 테스트 벡터의 추출 방법, 그리고 테스트 패턴 생성 방법에 대해 논의한다. 3장에서는 구현한 테스트 패턴 발생기에 대한 모의실험 결과를 검토하고 4장에서 결론을 맺는다.

## Ⅱ. 고장 모델 및 테스트 벡터

### 1. 고장 모델

가정한 고장 모델은 하나의 게이트 내부에서 발생 가능한 모든 단락으로 인한 합선고장이다. 게이트를 이루는 스위칭 수준의 회로에서 각 노드는 특성에 따라 다음과 같이 세 종류로 분류된다.[4]

1) 상수 노드 (constant node)
상수의 논리 값을 갖는 노드(VDD 또는 GND)

2) 입력 노드 (input node)
임의의 논리 값(1 또는 0)으로 설정해 놓을 수 있는 게이트의 입력 노드

3) 종속 노드 (dependent node)
논리 값이 게이트의 입력 신호에 종속되는 노드 (출력 노드, 내부 노드)

2-입력 CMOS NAND 게이트의 회로도를 그림 1에 도시하였다. 그림에서 상수 노드는 VDD와 GND로 표시된 곳이고 입력 노드는 IN1, IN2이며 종속 노드는 출력단인 OUT와 직렬로 연결된 두 개의 nMOS 사이의 A로 표시된 곳이다.

고장 모델은 스위칭 수준의 회로 내에서 분류된

세 종류의 노드들 사이에서 발생될 수 있는 모든 합선 고장이므로 노드들의 모든 가능한 조합으로 이루어져 다음과 같이 여섯 종류의 단락 형태로 나누어진다.

1) 상수전위를 갖는 두 노드 사이의 단락
2) 상수 노드와 입력 노드 사이의 단락
3) 상수 노드와 종속 노드 사이의 단락
4) 두 입력노드 사이의 단락
5) 입력 노드와 종속 노드 사이의 단락
6) 두 종속 노드 사이의 단락

그림 1에 6가지의 단락들 중에 2), 3), 4), 6)의 단락 형태를 보여주고 있다.

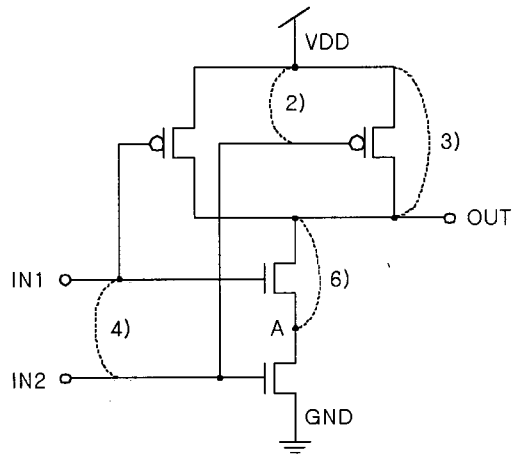


그림 1. 2-입력 CMOS NAND 게이트의 회로도

## 2. 테스트 벡터

다음으로는 앞 절에서 가정한 각 단락을 검출하기 위해 게이트의 입력으로 요구되는 테스트 벡터의 생성 방법에 대해 살펴보기로 한다.

앞에서 정의한 단락 종류 1)이 발생되었을 경우 게이트의 입력 값에 관계없이 상수전위 VDD와 GND 사이에 전류의 경로가 형성되므로 별도의 테스트 벡터가 요구되지 않는다. 그러나 다른 종류의 단락을 검출하기 위해서는 두 노드 사이에 VDD-GND 경로를 만들어 주어야하므로 두 노드의 논리 값이 서로 상반되도록 게이트의 입력 벡터를 인가해 주어야 한다. 즉, 그림 1에서 6)번 단락을 검출하기 위해서는 노드 OUT과 A 사이에 전류의 경로를 생성시켜 주어야하므로 OUT이 "1", A가 "0" 상태를 갖도록 하기 위해 IN1에 "0", IN2에 "1"을 인가시켜 주어야 한다. 따라서 이때 요구되는 2-입력 CMOS NAND 게이트의 테스트 벡터는 (IN1, IN2) = (0, 1)가 된다. 이러한 방식으로 게이트 내의 가정한 모든 단락을 검출할 수 있는 테스트 벡터를 찾아낸다.

게이트 종류에 따른 테스트 벡터를 찾아 유용한 테스트 패턴을 얻기 위해 다음과 같은 두 단계의 절차를 수행한다.

단계 1) 기본 게이트(NAND, NOR, AND, OR, NOT, Buffer, EXOR, EXNOR)의 스위칭 수준의 CMOS 회로를 참조하여 가정한 6가지 형태의 단락을 검출할 수 있도록 각 게이트에 따른 테스트 벡터를 찾아내어 게이트 종류와 입력 개수별로 라이브러리를 구축한다.

단계 2) 테스트 대상 회로에 인가되는 테스트 패턴에 따라 회로를 구성하는 모든 게이트들의 입력 값이 결정되어지며, 이 값을 단계 1에서 구축한 라이브러리내의 테스트 벡터와 비교하여 일치하는 테스트 벡터를 삭제한다. 즉, 삭제된 테스트 벡터에 해당되는 단락을 인가된 테스트 패턴으로 검출하였음을 의미한다. 따라서 인가된 패턴을 유용한 테스트 패턴으로 저장한다.

단계 1의 게이트에 대한 테스트 벡터 라이브러리 구축의 예로써 그림 2의 2-입력 CMOS NOR 게이트의 모든 내부 단락을 검출해 주는 테스트 벡터의 생성 단계에 대해 살펴보면 다음과 같다.

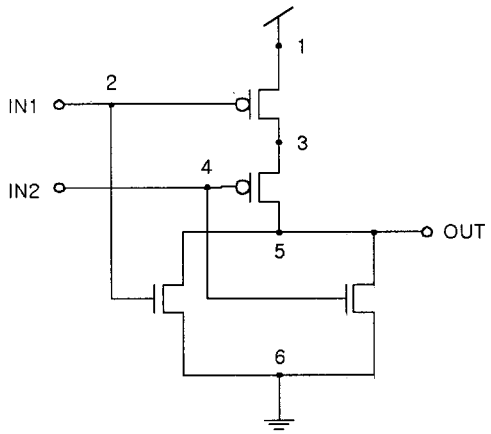


그림 2. 2-입력 CMOS NOR 게이트의 회로도

1) 그림 2의 2-입력 NOR 게이트의 노드를 종류

별로 나눈다.

- 상수 노드 : 1(VDD), 6(GND)
- 입력 노드 : 2(IN1), 4(IN2)
- 종속 노드 : 3(internal node), 5(OUT)

2) 각 노드 사이에 이루어질 수 있는 모든 단락을 조사한다. 그림을 살펴보면 노드 6개 중에 2개씩 짝을 이루는 형태가 되므로 ${}_6C_2$가 되어 모든 15개의 단락이 존재하게 된다. 노드 사이의 단락은 Sij로 정의하며, 여기에서 i, j는 단락이 발생된 노드의 번호가 된다. 즉, 노드 2와 3 사이의 단락은 S23으로 표현된다.

3) 모든 단락을 검출할 수 있는 테스트 벡터를 찾아내기 위해 표 1과 같은 결정표를 작성한다. 여기에서 고려해야 할 사항은 단락이 발생했다고 가정한 두 노드에 서로 상반된 값을 갖도록 해주어야 한다는 것이다.

표 1의 음영으로 표시된 곳을 모두 포함하는 테스트 벡터를 찾아내야 하며 여기에서 "1"은 그 위치에 해당되는 입력을 인가했을 때 왼쪽의 단락을 검출할 수 있음을 나타내고 "X"는 어떠한 패턴에 관계없이 그 단락을 검출할 수 있음을 의미한다. 또한 진하게 표시된 곳은 반드시 위쪽의 입력 값을 인가해야 검출할 수 있게 된다. 따라서 이러한 사항들을 종합하여 최적의 테스트 벡터를 구해보면 (IN1, IN2) = (00, 01, 10)가 된다.

### 3. 테스트 패턴 생성 방식

다음으로는 테스트 대상 회로의 주 입력단에 인가된 패턴으로 전파된 각 게이트의 입력 값이 라이브러리로 구성된 테스트 벡터를 형성시켜 주는지에 대한 검사 단계인 2 단계에 대해 살펴보자. 그림 3에 ISCAS '85의 벤치마크 회로 중에 하나인 c17을 도시하였으며 회로의 입력으로 "10101"을 인가했을 때, 각 게이트가 갖는 입력 값을 표시하였으며 이 입력 값이 각 게이트 별로 구축해 놓은 라이브러리의 테스트 벡터와 일치하면 삭제함으로서 고장 검출률을 얻어낼 수 있다. 테스트 패턴을 생성할 때 이용할 수 있도록 2-입력 NAND 게이트의 테스트 벡터인 (IN1, IN2) = (11, 10, 01)와 각 게이트의 현재 논리 상태를 비교하여 일치되는 경우에는 그 테스트 벡터에 해당되는 고장을 검출하게 되는 것이다. 즉, 빗금으로 표시된 NAND 게이트에서는 입력 "10"에 해당되는 단락을 검출하게 된다. 이러한 방식으로 회로내의 모든 게이트가 요구하는 테스트 벡터를 경험하도록 하는 유용한 테스트 패턴을 얻을 수 있다.

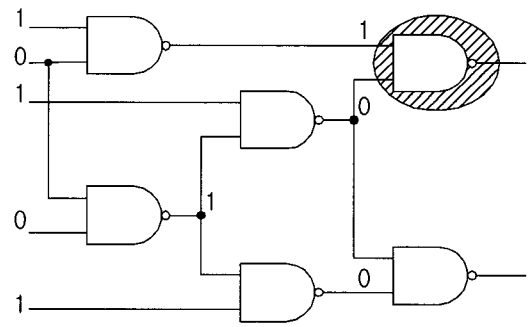


그림 3. ISCAS '85의 벤치마크 회로 c17.

표 1. 2-입력 NOR 게이트의 내부 단락을 위한 테스트 벡터 생성 결정표.

| IN1 IN2 | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| S12 | 1 | 1 | | |
| S13 | | | 1 | |
| 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 | 1 | | | |
| S35 | | 1 | | |
| S36 | 1 | 1 | | |
| S45 | 1 | 1 | | 1 |
| S46 | | 1 | | 1 |
| S56 | 1 | | | |

## Ⅲ. 테스트 패턴 발생기의 모의 실험 및 검토

개발한 테스트 패턴 발생기는 PC의 Windows 98 기반에서 Delphi 소프트웨어를 이용하여 사용자의 편의를 위해 GUI(Graphic User Interface) 방식으로 구현하였다. 그림 4에 고장 시뮬레이터 화면을 보여 주고 있다.

그림에서 Preprocess는 회로의 연결 정보, 게이트의 종류, 게이트의 식별 번호, 각 게이트 별로 요구되는 테스트 벡터 등을 추출하는 단계이고 Initialize는 테스트 패턴 생성 과정을 수행하기 전 초기화시키는 단계이다.

조합회로로 구성되어 있는 ISCAS '85 벤치마크 회로에 대해 무작위(random) 테스트 패턴 10,000개를 입력하여 테스트 패턴 발생기의 성능을 평가였다. 무작위 패턴을 이용한 것은 테스트를 회로 자체 내에서 빠른 속도로 수행할 수 있는 BIST (Built-In Self Test)를 적용할 수 있음을 의미한다. 또한 10,000개 이상의 무작위 테스트 패턴을 인가했을 경우에는 테스트 패턴 생성 시간이 많이 소모되지만 고장 검출률에 있어서는 별다른 향상을 보이지 않는다.[3]

그림 4에서는 c499에 대해 수행한 결과를 보여주고 있으며 총 10,000개의 무작위 패턴에서 90개의 테스트 패턴으로 모든 가정한 고장을 검출 할 수 있으며 0.72초의 수행 시간 동안 고장 시뮬레이션 및 패턴 생성을 하였음을 알 수 있다.

표 2에 구현한 패턴 발생기를 ISCAS '85 회로에 적용하여 Pentium Ⅱ 400MHz를 갖는 PC상에서 모의실험을 수행한 결과를 보여 주고 있다. 요구되는 총 테스트 벡터는 내부 단락을 검출하기 위해 각 게이트가 요구하는 테스트 벡터의 총 수를 의미하고 경험하지 못한 테스트 벡터는 게이트에서 요구하는 테스트 벡터를 주어진 입력 패턴으로 경험하지 못한 경우를 말한다. 고장 검출률은 (경험한 테스트 벡터)×100/요구되는 총 테스트 벡터로써 계산하였다. 그리고 경험한 테스트 벡터에는 중복(redundancy)고장에 대한 테스트 벡터는 제외되었다.

모의실험 결과를 살펴보면 중복고장이 없는 회로에서는 100%의 고장 검출률을 갖는 테스트 패턴을 발생시켰으며 나머지 회로에서도 높은 검출률을 갖는 테스트 패턴을 짧은 시간 안에 생성시켜 준다는 것을 알 수 있다.

본 논문에서 구현한 테스트 패턴 발생기의 효율성을 평가하기 위해 고장 검출률, 생성된 테스트 패턴의 개수, 테스트 패턴 생성 시간을 기존의 방식과 비교하였다.

먼저 구현한 테스트 패턴 발생기의 고장 검출률을 기존에 발표된 방식과 비교하여 표 3에 보였다. c6288을 제외한 모든 회로에서 Mahlstedt가 구현한 방식보다 높은 고장 검출률을 보이고 있으며 G-ATPG와는 거의 비슷한 결과를 확인할 수 있다.

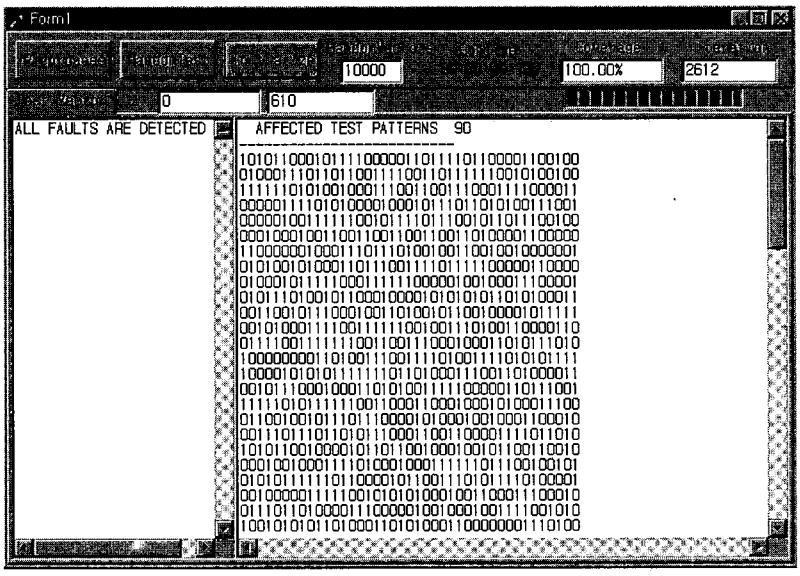


그림 4. 구현된 테스트 패턴 발생기.

표 2. ISCAS '85 벤치마크 회로에 대한 모의실험 결과.

| 회로 | 테스트 패턴 | 요구되는 총 테스트 벡터 | 경험하지 못한 테스트 벡터 | 고장 검출률 (%) | CPU time (sec.ms) |
|---|---|---|---|---|---|
| c432 | 39 | 496 | 0 | 100.00 | 0.11 |
| c499 | 90 | 610 | 0 | 100.00 | 0.72 |
| c880 | 50 | 1112 | 0 | 100.00 | 2.47 |
| c1355 | 92 | 1610 | 0 | 100.00 | 1.32 |
| c1908 | 129 | 2377 | 0 | 100.00 | 4.60 |
| c2670 | 63 | 3268 | 28 | 99.14 | 17.90 |
| c3540 | 81 | 4605 | 46 | 99.00 | 20.82 |
| c5315 | 77 | 6693 | 2 | 99.97 | 29.99 |
| c6288 | 37 | 7216 | 35 | 99.51 | 30.59 |
| c7552 | 109 | 9656 | 110 | 98.86 | 51.30 |

표 3. 고장 검출률 비교(%).

| 회로 | 제안한 방법 | Mahlstedt[3] | G-ATPG[5] |
|---|---|---|---|
| c432 | 100.00 | 96.10 | 100.00 |
| c499 | 100.00 | 94.43 | 100.00 |
| c880 | 100.00 | 99.98 | 100.00 |
| c1355 | 100.00 | 99.61 | 100.00 |
| c1908 | 100.00 | 98.63 | 100.00 |
| c2670 | 99.14 | 98.33 | 99.14 |
| c3540 | 99.00 | 98.72 | 98.94 |
| c5315 | 99.97 | 99.20 | 99.97 |
| c6288 | 99.51 | 99.80 | 99.51 |
| c7552 | 98.86 | 98.26 | 99.79 |

표 4. 발생시킨 테스트 패턴 개수의 비교

| 회로 | 제안한 방법 | Mahlstedt[3] | G-ATPG[5] |
|---|---|---|---|
| c432 | 39 | 48 | 30 |
| c499 | 90 | 97 | 99 |
| c880 | 50 | 65 | 30 |
| c1355 | 92 | 97 | 97 |
| c1908 | 129 | 148 | 168 |
| c2670 | 63 | 74 | 43 |
| c3540 | 81 | 98 | 106 |
| c5315 | 77 | 103 | 81 |
| c6288 | 37 | 59 | 126 |
| c7552 | 109 | 116 | 136 |

표 4는 생성된 테스트 패턴의 개수를 비교하였다. 모든 회로에서 Mahlstedt의 방식보다 적은 개수의 패턴을 발생시켰으며, G-ATPG와 비교해 보면 c432, c880, c2670을 제외한 다른 회로에서 적은 테스트 패턴을 생성하여 구현한 패턴 발생기의 우수한 성능을 검증할 수 있다.

테스트 패턴을 생성하는데 소요되는 시간을 표 5에 보여주었다. Mahlstedt의 경우에는 APPLO WS(20MIPS)를 사용하였고 G-ATPG는 Pentium 66MHz PC를 사용하여 서로 사용한 하드웨어 플랫폼이 다르기 때문에 정확한 성능 비교는 어렵지만 모든 회로에서 짧은 시간 내에 테스트 패턴을 생성해 줌을 확인할 수 있다.

앞의 성능 비교 평가를 통해 본 논문에서 구현한 테스트 패턴 발생기의 효율성이 기존의 다른 방식들에 비해 증대되었음을 알 수 있다.

표 5. 테스트 패턴 생성 시간의 비교

| 회로 | 제안한 방법 (sec.ms) | Mahlstedt[3] (sec) | G-ATPG[5] (sec.ms) |
|---|---|---|---|
| c432 | 0.11 | 3 | 0.80 |
| c499 | 0.72 | 5 | 7.30 |
| c880 | 2.47 | 4 | 2.49 |
| c1355 | 1.32 | 6 | 6.20 |
| c1908 | 4.60 | 10 | 8.39 |
| c2670 | 17.90 | 14 | 18.20 |
| c3540 | 20.82 | 20 | 34.77 |
| c5315 | 29.99 | 29 | 46.65 |
| c6288 | 30.59 | 27 | 57.15 |
| c7552 | 51.30 | 51 | 72.71 |

## Ⅳ. 결 론

본 논문에서는 고집적화에 따른 주변 노드 사이의 합선, 게이트 옥사이드 두께의 감소 등으로 인한 게이트 내부단락을 고려하여 이 고장을 검출하는데 사용되는 IDDQ 테스트를 위한 테스트 패턴 발생기를 개발하였다.

먼저 기본 게이트(NAND, NOR, AND, OR, NOT, Buffer, EXOR, EXNOR)에 대하여 각 게이트에 대한 트랜지스터 구성도를 참조하여 게이트 내부의 모든 단락을 검출하기 위한 테스트 벡터를 추출하였고 회로에 인가되는 입력 패턴에 따른 각 게이트의 입력 논리 상태를 앞에서 얻은 테스트 벡터와 비교하여 일치할 경우에는 유용한 테스트 패턴으로 저장하였다.

개발한 고장 시뮬레이터는 PC의 Windows 98 기반에서 Delphi 소프트웨어를 이용하여 사용자의 편의를 위해 GUI(Graphic User Interface) 방식으로 구현하였으며, ISCAS '85 벤치마크 회로를 이용한 모의실험 결과 고장 검출률, 생성된 테스트 패턴의 개수, 테스트 패턴 생성 시간에서 기존의 방식에 비해 효율성이 증가되었음을 확인하였다.

앞으로 게이트 수준의 회로를 트랜지스터 수준으로 변환하여 각 노드 사이에 발생 가능한 합선고장에 대한 테스트를 위한 효과적인 테스트 패턴 발생기에 대한 연구가 필요하다.

## 참 고 문 헌


[1] R. Rajsuman, Iddq Testing for CMOS VLSI, Artech house, 1994.

[2] P. Tardikaran, et al., "Fault Simulation of IDDQ Tests for Bridging Faults in Sequential Circuit," IEEE/ACM Fault-Tolerant Computing Symposium, 1995.

[3] U. Mahlstedt, et al., "Test Generation for IDDQ Testing and Leakage Fault Detection in CMOS Circuits," European DAC, pp. 486-491, Sept. 1992.

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

[5] 김강철 외, "CMOS VLSI의 IDDQ 테스팅을 위한 ATPG 구현", 대한전자공학회논문지, 제 33권, A편, 제 3호, pp. 176~186, 1996.



김 대 익(Dae Ik Kim) 정회원

제22권 제10호 참조

현재 : 전북대학교 전기전자회로합성연구소 연구원

<주관심 분야> 메모리 테스트, Built-In Self Test, Iddq testing

전 병 실(Byoung Sil Chon) 정회원

제24권 제5B호 참조

현재 : 전북대학교 공과대학 전자정보공학부 교수

<주관심 분야> Design for Testability, ATM switching