논문 2004-41SD-11-10

# 그라운드 바운스 영향과 지연고장을 위한 최소화된 테스트 패턴 생성 기법

# (A Minimized Test Pattern Generation Method for Ground Bounce Effect and Delay Fault Detection)

김 문 준*, 이 정 민*, 장 훈**

(Moon-Joon Kim, Jeong-Min Lee, and Hoon Chang)

요 약

본 논문에서는 ground bounce 영향과 지연고장 검출을 함께 고려한 효율적인 보드레벨 연결선 테스트 생성 알고리즘을 제안한다. 제안된 알고리즘은 IEEE 1149.1의 연결선 테스트, ground bounce 영향에 의한 바운더리 스캔의 오동작 방지, 그리고 연결선의 지연고장 검출 능력을 포함한다. 본 논문에서 제안하는 기법은 기존의 기법에 비해 연결선의 지연고장 검출능력을 새롭게 추가하였지만, 연결선 테스트에 필요한 총 테스트 패턴 수는 기존의 기법과 비교해서 큰 차이를 보이지 않음을 실험결과에서 확인할 수 있다.

Abstract

An efficient board-level interconnect test algorithm is proposed considering both the ground bounce effect and the delay fault detection. The proposed algorithm is capable of IEEE 1149.1 interconnect test, negative ground bounce effect prevention, and also detects delay faults as well. The number of final test pattern set is not much different with the previous method, even our method enables to detect the delay faults in addition to the abilities the previous method guarantees.

**Keywords**: IEEE 1149.1, Interconnect Test, Ground Bounce, Delay Fault Test

## I. 서 론

보드 위에 실장된 칩들 간의 연결선을 테스트하기 위해서는, 테스트 엔지니어가 연결선의 입력에 테스트 패턴을 인가하고 연결선의 출력으로부터 테스트 응답을 관측하여, 예측된 올바른 응답과 비교하여 고장 유무를 확인한다. 연결선 테스트는 IEEE 1149.1 표준안의 EXTEST 모드를 사용한다[1]. 연결선 고장 테스트를 위한 고장 모델은 단일 고착 고장과 다중 결합 고장(multiple-net shorts) 모델 등이 주로 사용되며, 이러한 보드 위의 연결선을 테스트하기 위한 여러 가지 테스트 패턴 생성 기법들이 제안되었다[2-7]. 이들의 목적은 원하는 고장 검출율을 보장하면서도 가능한 적은 테스트 패턴 셋을 생성하여 테스트 시간 비용을 최소화하는 것이다. Counting Sequence 알고리즘은 각각의 연결선에 서로 다른 테스트 패턴을 각각 인가하여 다중 결합 고장을 검출한다[2]. 만일 연결선사이에 합선(short)이 발생했다면 서로 다른 연결선에서 동일한 테스트 응답을 관측함으로써 고장을 발견할 수 있다. 하지만 이 알고리즘은 모든 연결선의 단일 고착 고장을 검출하지 못한다는 단점이 존재한다. 이러한 단점을 보완한 Modified Counting Sequence 알고리즘이 개발되었으며, 모든 연결선의 다중 결합 고장과 단일 고착 고장을 테스트 할 수 있게 되었다[3]. 또한 보드레벨 연결선 테스트뿐만 아니라

* 학생회원, ** 정회원 숭실대학교 컴퓨터학과
(Department of Computing, Soongsil University)
※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
접수일자: 2004년3월9일, 수정완료일: 2004년11월4일

진단 기능까지 포함한 테스트 생성 기법이 개발되었다[4]. 연결선에 고장이 발생하여 고장의 영향을 받은 테스트 응답이 고장이 발생하지 않은 다른 연결선의 테스트 응답과 동일한 경우 앨리어싱(aliasing) 문제가 발생한다. True/Complement 테스트 알고리즘은 기존의 고장 모델을 모두 검출할 수 있을 뿐만 아니라 테스트 응답의 앨리어싱 문제를 해결하였으며 현재 보드레벨 연결선의 테스트 생성 알고리즘으로서 널리 사용되고 있다. 또한 보드레벨 연결선의 지연 고장까지 테스트하기 위한 테스트 패턴 생성 기술이 제안되었다[5]. [5]에서 제안한 구조를 사용하면 연결선의 합선/단선 등의 고장과 지연 고장을 모두 검출할 수 있다. 하지만 제안된 테스트 패턴 생성 방법으로는 그라운드 바운스 (Ground Bounce: 이하 GB) 영향에 따른 바운더리 스캔의 오동작 문제를 검출할 수 없다.

최근 GB 현상으로 인한 IEEE 1149.1 바운더리 스캔의 오동작에 대한 관심이 집중되었다[6]. GB는 보드레벨 연결선을 테스트하기 위해 테스트 패턴이 인가될 때, 정해진 기준 수치 이상의 천이(switching)가 연결선에서 동시에 일어날 경우 칩과 보드의 전압 레벨에 변동이 생겨 로직 값이 잘못 인식되는 현상이다. 따라서 이러한 현상을 방지하기 위해서 설계 당시 동시 천이 한계수치 (Simultaneously Switching Outputs Limit: 이하 SSOL)를 지정한다. 이러한 GB를 고려한 보드레벨 연결선 테스트를 위한 테스트 패턴 생성 알고리즘이 최근 개발되었다[7]. 이 기법은 지연 고장을 제외한 기존의 연결선 테스트의 고장 모델을 모두 포함하며, 동시에 GB의 영향에 따른 바운더리 스캔의 오동작 발생을 방지한다.. 이 기법은 테스트 패턴 선택에 있어서 동시 천이의 개수를 최소화시킨 후, 인가할 패턴이 동시 천이 한계수치를 초과할 경우 더미(dummy) 테스트 패턴을 생성된 테스트 패턴 셋(set)에 추가적으로 삽입하여 최종 테스트 패턴의 개수를 가능한 최소화시켰다. 그러나 이 기법은 지연 고장 테스트를 검출하지 못한다는 단점이 존재한다.

본 논문에서는 기존의 기법에서 보장하는 보드 레벨 테스트를 위한 연결선 테스트 및 GB 영향에 의한 오동작 문제 해결을 그대로 지원하면서, 동시에 지연 고장을 검출할 수 있는 테스트 패턴 셋을 생성하는 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 기존의 GB 영향를 고려한 테스트 패턴 생성 알고리즘을 살펴보고 이 알고리즘의 문제점에 대해서 알아본다. Ⅲ장에서는 본 논문에서 제안하는 지연 고장을 함께 고려한 테스트 패턴 생성 알고리즘에 대해 설명하고 Ⅳ장에서 실험결과를 분석한 후, Ⅴ장에서 결론을 맺는다.

## Ⅱ. 기존의 Ground bounce 영향을 고려한 보드레벨 테스트 패턴 생성 기법

Ⅱ장에서는 GB 영향을 고려한 테스트 패턴 생성 기법에 대해 간략히 설명한다. 이 기법은 [7]에서 연구되었으며, 기존의 True/Complement 테스트 알고리즘에 기반하였다[4]. 알고리즘은 크게 2단계로 나뉜다. 1단계는 코드워드(codeword) 선택과정이며, 2단계는 테스트 패턴 재정렬 과정이다.

### 1. 코드워드 선택 (1단계)

$k$개의 연결선을 테스트하기 위해서는 테스트 패턴 $p(k)=2\lceil \log_2 k \rceil$ 개가 필요하다[4]. 그림 1에 예제가 나타나있다.

그림 1의 연결선 개수 $k$=13 이며, 따라서 테스트 패턴의 개수 $p(k)$=8 이다. 이 테스트 패턴 셋은 단일 고착 고장과 다중 결합 고장을 모두 검출할 수 있으며 앨리어싱 문제를 해결한다. 그림의 테스트 패턴 셋에서 각각의 열(column)을 테스트 패턴이라 부르고, 각 행(row)을 코드워드(codeword)라 한다. $b=\lceil \log_2 k \rceil$ 라고 할 경우, 코드워드가 될 수 있는 모든 경우를 후보 코드워드(candidate codeword)라 부르고, 그 개수 $c(k)=2^b$ 로 표현할 수 있다. 그림 2에 $k$=13인 경우의 후보 코드워드가 나타나있다.

그림 2의 경우, 연결선의 개수($k$)는 13이므로, 총 $2^4$=16개의 후보 코드워드가 생성된다. 이 중에서 최종적으로 필요한 코드워드의 개수는 연결선의 개수인 $k$개이다. 연결선에서 일어나는 천이를 최소화하기 위해서,

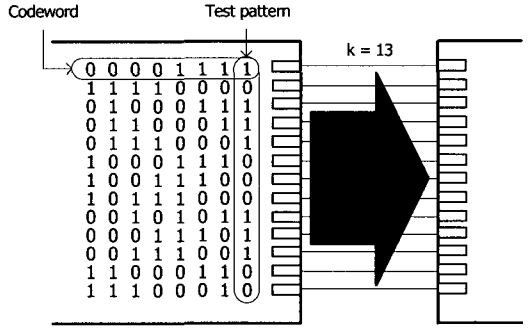



그림 1. 연결선 테스트를 위한 테스트 패턴 셋
Fig. 1. Test Pattern Set for Interconnect Test.

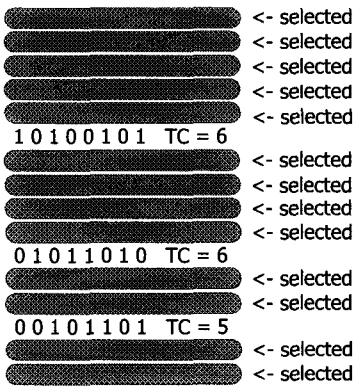



그림 2. 후보 코드워드 ($k$=13)
Fig. 2. Candidate Codewords ($k$=13).

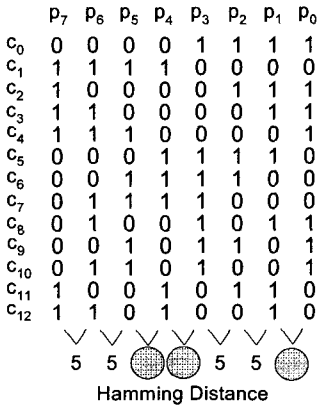



그림 3. 해밍 거리 계산
Fig. 3. Hamming Distance Calculation.

해당 연결선에 인가했을 경우 발생하는 천이 회수(transition count)가 가장 적은 $k$개의 코드워드를 선택한다. 각각의 연결선에서 일어나는 천이 회수를 최소화시키는 것이 연결선에서 일어나는 천이의 개수를 최소화시킬 수 있기 때문이다. 또한 [7]에서 이를 이용한 실험 결과가 가장 우수하였다. 이를 위해, 모든 후보 코드워드에 대해 각각 천이 회수를 구하여 가중치를 부여한다. 각각의 코드워드에 대한 천이 회수는 그림 2의 TC 값에 나타나있으며, 천이 회수가 가장 적은 코드워드 $k$개를 선택하여 최종 코드워드를 선택한다. 본 예제의 경우, 음영으로 처리된 코드워드들이 선택된 코드워드들이다.

$k$개의 연결선을 위한 코드워드의 선택이 완료되었으면 이 테스트 패턴 셋이 GB 영향에 의한 오작동 문제를 일으키는 지의 여부를 검증한다. 만일 1단계에서 선택된 테스트 패턴 셋이 바운더리 스캔의 오동작을 일으키지 않는다면 그대로 연결선을 위한 테스트 패턴으로 사용될 수 있기 때문이다. 설계 단계에서 GB 영향에 의해 오동작을 일으키지 않는 최대한의 동시 천이 계수 SSOL 값이 주어진다. 그림 3의 아래쪽에 나타난 숫자들은 현재 생성된 테스트 패턴 셋이 순차적으로 인가될 때 각각의 테스트 패턴끼리의 해밍 거리(hamming distance)를 표시한 것이다. 해밍 거리는 테스트 패턴이 인가될 때마다, 각 연결선에서 일어나는 천이의 회수를 모두 더한 값이다. 만일 계산된 해밍 거리 모두가 SSOL을 초과하지 않는다면, 즉 모두 같거나 작다면 이 테스트 패턴 셋은 GB 영향에 의한 바운더리 스캔 오동작을 일으키지 않으므로 알고리즘이 종료된다. 만약 그림 3의 음영처리된 부분처럼 SSOL을 초과하는 해밍거리가 존재한다면 2단계인 테스트 패턴 재정렬 과정을 거친다.

## 2. 테스트 패턴 재정렬 (2단계)

1단계 과정을 거쳐 생성된 테스트 패턴 셋에 SSOL 계수를 초과하는 해밍 거리가 존재할 경우, 이를 제거하기 위해 추가적으로 더미 패턴을 삽입하여 해밍 거리를 낮추어 GB 영향을 제거한다. 해밍 거리는 연속된 테스트 패턴간의 천이를 계산한 것이다. 하지만 각각의 테스트 패턴은 인가 순서에 관계없이 목표 고장 모델을 모두 검출할 수 있다[7]. 따라서 2단계에서 테스트 패턴의 재정렬 과정을 수행하는 것만으로도 해밍 거리가 SSOL을 초과하는 경우의 수를 줄일 수 있다. 즉, 해밍 거리를 최소화하여 가능한 적은 수의 더미 패턴을 삽입, 결과적으로 총 테스트 시간을 줄인다.

테스트 패턴을 최소한의 비용으로 재정렬하는 문제는 세일즈맨 문제[8]와 동일하다. 이 NP-난해(hard) 문제는 탐욕 휴리스틱 알고리즘(greedy heuristic algorithm)에 기반하여 해결하였다. 시작 테스트패턴을 무작위로 지정한 후, 해당 테스트 패턴에서 다음 테스트 패턴을 선택하였을 때, 추가 테스트 패턴 개수가 가장 적은 테스트 패턴을 선택하는 기법이다. 이를 반복하여 최종적으로 최소의 거리만을 이용한 모든 테스트 패턴의 인가순서를 결정한다. 기존의 기법에서는 이러한 탐욕 휴리스틱 알고리즘을 이용하여 테스트 패턴을 재정렬하였다. 이를 위해서 방향성이 없는 그래프(undirected graph)를 그린다. 각각의 테스트 패턴이 노드(node)가 되고, 테스트 패턴간의 해밍 거리를 간선(edge)으로 정한다. 단, 간선 값은 해밍 거리를 직접 사용하지 않고, 각각의 테스트 패턴 사이에 삽입해야 할 추가 패턴의 개수로 나타낸다. 해당 테스트 패턴 사이에 추가해야 할 개수는 $\lceil d/s \rceil - 1$로써 나타낼 수 있다. 여기서 $d$는 해밍 거리, $s$는 SSOL 계수를 나타낸다. 그림 3에서 보인 테스트

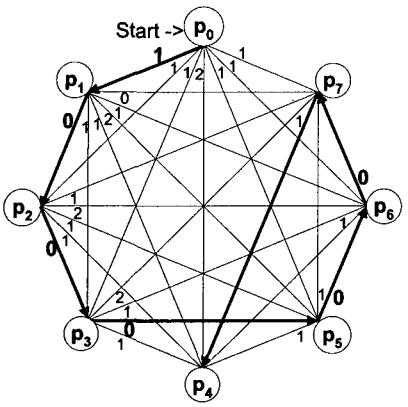



그림 4. 탐욕 휴리스틱 알고리즘을 이용한 테스트패턴 재정렬
Fig. 4. Reordering with Greedy Heuristic Algorithm.

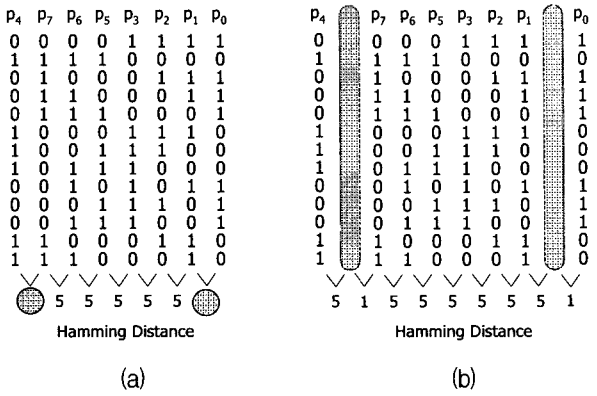



그림 5. 더미 테스트 패턴 삽입 과정
Fig. 5. Dummy Test Pattern Insertion.

패턴 셋의 해밍 거리를 이용하여 $k$=13, $s$=5인 경우의 재정렬을 위한 그래프를 그리면 다음의 그림 4와 같이 나타낼 수 있다.

그림 4에서 보는 바와 같이, 노드의 수는 테스트 패턴과 같이 8개 이고, 각 노드끼리의 간선은 총 28개이다. 간선에 나타나 있는 숫자는 테스트 패턴을 연속으로 인가할 경우 GB 영향에 의한 바운더리 스캔 오동작을 방지하기 위해 필요한 추가 테스트 패턴의 개수이다. 총 테스트 패턴의 수를 최소화시키기 위해서는 무작위로 첫 테스트 패턴을 선택 후, 해당 노드에서 최소의 추가 테스트 패턴의 개수가 적힌 간선을 반복적으로 따라가면 테스트 패턴의 재정렬 과정이 완료된다. 기존의 재정렬 과정을 $k$=13, $s$=5인 예제에 적용시킨 결과를 그림 4에 굵은 선으로 표시하였다. 최종적으로 재정렬 된 테스트 패턴은 $p_0$, $p_1$, $p_2$, $p_3$, $p_5$, $p_6$, $p_7$, $p_4$ 의 순이며, 이러한 순서로 해밍 거리를 다시 계산하면 그림 5의 (a)와 같이 나타낼 수 있다.

그림 5의 (a)의 해밍 거리 값을 살펴보면, 그림 3의 테스트 패턴 셋이 보이는 해밍거리에 비해서, 추가 테스트 패턴이 필요한 경우의 수가 3개에서 2개로 줄어들었음을 확인할 수 있다. 즉, 재정렬 과정을 수행하는 것만으로도 테스트 패턴 셋의 SSOL 계수를 초과하는 경우의 수가 줄어든 것을 알 수 있다. 만약 예제의 경우와 같이, 재정렬 과정을 수행한 후에도 SSOL 수치를 초과하는 해밍 거리가 아직 존재할 경우, 그림 5의 (b)에서와 같이, 해당 비트포지션에 $\lceil d/s \rceil - 1$ 개의 추가 패턴을 삽입해 줌으로써 GB 문제를 해결한다. 더미 테스트 패턴을 추가적으로 삽입함으로써 동시에 연결선에서 일어나는 천이의 수를 분산시켜 모든 해밍거리를 SSOL 수치 이하로 떨어뜨려 바운더리 스캔의 오동작을 방지한다. 반대로 SSOL 수치를 초과하는 해밍거리가 존재하지 않을 경우에는 재정렬 과정을 거친 테스트 패턴 셋을 연결선 테스트를 위한 최종 테스트 패턴으로 사용한다.

## 3. 기존의 보드레벨 연결선을 위한 테스트 패턴 생성 기법의 문제점

보드 레벨 연결선의 지연 고장 검출을 보장하기 위해서는, 각각의 연결선의 지연 고장 모델을 선정하고 이를 위한 테스트 패턴을 인가해주어야 한다[5]. 지연 고장은 해당 연결선의 로직 값이 변화할 경우, 주어진 시간 안에 천이하지 못하는 결함을 의미한다. 따라서 보드 레벨의 연결선의 지연 고장을 테스트하기 위해서는 각각의 연결선에 모든 천이, 즉 0에서 1, 그리고 1에서 0으로의 천이를 강제적으로 일으키고 난 후 테스트 응답을 분석해야 한다.

하지만 기존의 보드 레벨 연결선을 위한 테스트 패턴 생성 기법의 경우, 이러한 연결선의 지연 고장 테스트를 보장하지 못한다는 단점이 있다. 그림 5의 (b)에서 예시된 테스트 패턴 셋을 보자. 이는 $k$=13, $s$=5인 경우이다. 각각의 연결선의 지연 고장은 검출하기 위해서는, 해당 코드워드에 0에서 1, 그리고 1에서 0으로의 천이가 모두 존재해야 한다. 하지만 그림 5의 (b)를 보면 알 수 있듯이, 코드워드 $c_2$~$c_{12}$에는 두 가지 천이가 모두 적어도 한 번 이상씩 일어나기 때문에, 해당 연결선의 지연 고장을 모두 테스트 할 수 있지만, 코드워드 $c_0$와 $c_1$의 경우 각각 1에서 0으로의 천이만이, 그리고 0에서 1로의 천이만이 존재한다. 즉, $c_0$는 0에서 1로의 지연 고장을, $c_1$은 1에서 0으로의 지연 고장을 테스트하지 못하는 것이다.

기존의 보드 레벨 연결선을 위한 테스트 패턴 셋은, 그림 5의 (b)와 같이 연속적인 테스트 패턴 간의 해밍 거리가 SSOL 계수 $s$인 5보다 같거나 작게 되어 GB 영

향에 의한 바운더리 스캔의 오동작 문제를 올바르게 해결한 것을 알 수 있다. 하지만 코드워드를 살펴보면 $c_0$의 코드워드는 0에서 1로의 지연 고장 테스트가 불가능하고, $c_1$의 코드워드는 해당 연결선의 1에서 0으로의 지연 고장 테스트가 불가능하다. 이처럼 기존의 보드레벨 연결선 테스트 및 GB 문제를 해결하기 위해 개발된 테스트 패턴 생성 기법은 지연 고장을 완벽하게 지원하지 못한다. 본 논문에서는 기존의 기법이 제공하는 연결선 테스트 및 GB 문제를 해결하고, 또한 연결선의 지연고장 테스트를 어떤 경우라도 완벽하게 수행할 수 있는 테스트 패턴 생성 기법을 제안한다.

## Ⅲ. 제안된 보드레벨 테스트를 위한 테스트 패턴 생성 기법

Ⅲ장에서는 GB 영향을 고려하고, 동시에 지연 고장을 검출할 수 있는 보드레벨 연결선을 위한 테스트 패턴 생성 기법에 대해 설명한다.

기존의 기법 및 본 논문에서 제안하는 테스트 패턴 생성 기법은 공통적으로 True/Complement 알고리즘에 기반한다. True/Complement 알고리즘을 이용하여 기본적으로 생성된 테스트 패턴의 개수($p(k)$)는 $2\lceil \log_2 k \rceil$ 개로써, 항상 짝수 개로 구성된다. 그리고 각 코드워드는 반드시 $b$개의 논리값 0과 나머지 $b$개의 논리값 1로 구성된다. 이는 그림 6과 같이 우선적으로 생성된 $b$개의 패턴을 반전시켜 또 다른 $b$개의 패턴을 생성한 후 이들을 조합하여 전체 테스트 패턴을 생성되기 때문이다[4].

GB 영향에 따른 바운더리 스캔 오동작 문제와 연결선의 지연 고장 검출을 동시에 고려하기 위해서는, 기존의 기법에서 제공하는 기능을 모두 만족하면서 연결선의 지연 고장을 검출할 수 있는 테스트 패턴을 생성해야 한다. 연결선의 지연 고장을 테스트하기 위해서는 테스트 패턴 셋에 해당하는 각각의 코드워드가 0에서 1, 1에서 0으로의 천이가 각각 적어도 한 번씩 일어나면 된다. 이를 위해서 본 논문에서는 기존의 GB 문제만을 고려한 기법 중 1단계인 코드워드 선택 과정은 그대로 적용하고, 2단계인 테스트 패턴 재정렬 과정만을 교체하는 알고리즘을 제안한다. 1단계에서는 기존의 기법과 마찬가지로 True/Complement 알고리즘에 의해 후보 코드워드를 생성한 후, 각 코드워드의 천이 개수(TC)를 계산하여 가장 적은 TC 값을 가지는 $k$개의 코드워드를 선택한다. 그 후 2단계에서 GB 영향에 의한 바운더리 스캔의 오동작을 방지하기 위한 재정렬 과정을 거칠 때, 본 논문에서 제안된 재정렬 기법에 의해 지연 고장을 함께 고려하여 테스트 패턴의 인가 순서를 결정하게 된다. 본 논문에서 제안하는 기법의 재정렬 과정의 의사코드는 다음과 같다.

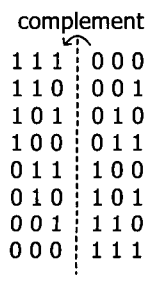



그림 6. True/Complement 알고리즘의 특성
Fig. 6. Characteristic of True/Complement Algorithm.

**Proposed_Reorder_Algorithm**

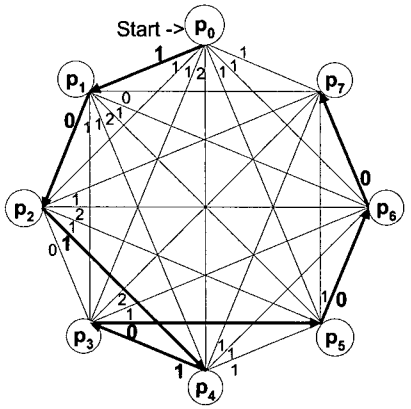
```
1 : T = 코드워드 선택과정이 완료된 테스트 패턴 셋 ;
2 : D = 각 테스트 패턴간의 해밍 거리 계산 (T) ;
3 : s = SSOL 계수 ;
4 : G = Draw_Graph (T, D, s) ;
5 : T_R = 초기화된 재정렬이 완료된 테스트 패턴 셋 ;
6 : Append (T_R, p_0) ;
7 : while ( !모든 테스트 패턴 정렬 완료 )
8 : {
9 :   if ( b번째 테스트 패턴 )
10 :  {
11 :      while (Any_Continuous_Codeword_With_All_1_or_0 (b,
          k) == YES)
          x = 다음 가중치로 선택된 테스트패턴 인덱스 ;
12 :      Append (T_R, p_x) ;
13 :  }
14 :  else
15 :  {
16 :      x = 가장 적은 가중치로 선택된 테스트 패턴 인덱스
;
17 :      Append (T_R, p_x) ;
18 : }
19 : return T_R ;
```

1단계인 코드워드 선택과정이 완료된 테스트 패턴 셋을 $T$라고 하고(line 1), 각각의 테스트 패턴 간의 해밍 거리 $d$를 계산한 값들을 $D$에 저장한다(line 2). 테스트 패턴 셋 $T$와 계산된 해밍 거리 값들이 저장된 $D$, 그리고 SSOL 계수인 $s$를 이용하여(line 3) 기존의 기법과 마찬가지로 그래프를 그린다. 그래프는 방향성이 없으며 각 노드는 각각의 테스트 패턴이다. 또한 각 노드 사이의 간선은 해당 테스트 패턴 사이에 삽입해야 할 추가 패턴의 개수인 $\lceil d/s \rceil - 1$ 값을 할당한다(line 4). $T_R$은 $T$의 패턴을 재정렬한 후 저장할 테스트 패턴 셋이다(line 5).

시작 노드를 첫 번째 테스트 패턴으로 지정한 후(line 6), 남은 테스트 패턴 개수인 $p(k)$-1번 만큼의 루프를 수행하면 모든 테스트 패턴의 재정렬 과정이 완성된다(line 7). 루프를 수행하면서 $b$번째 테스트 패턴을 제외한 모든 테스트 패턴에 대해서는, 각 노드에서의 가장 적은 비용의 가중치를 선택하여 다음 테스트 패턴을 선택한다(line 14-18). 재정렬 된 테스트 패턴 셋이 지연 고장을 검출하기 위해서는 모든 코드워드가 적어도 한 개씩의 0에서 1, 1에서 0으로의 천이를 가져야만 한다. 각 코드워드는 $b$개의 0, $b$개의 1로 구성되므로 처음부터 연속된 0 또는 연속된 1이 발생하지 않는 한, 해당 테스트 패턴 셋은 모든 연결선의 지연 고장을 테스트할 수 있다. 따라서 $b$번째 테스트 패턴을 할당할 때(line 9-13), 첫 테스트 패턴부터 $b$-1 번째의 테스트 패턴까지 각 코드워드에 연속된 0 혹은 연속된 1이 발생했는지 우선 체크하고, 테스트 패턴 후보들 중 가장 높은 우선순위부터 할당 가능 여부를 두 번째 while문을 반복하여 비교한다(line 11-12). 이 경우, 어떠한 코드워드라도 $b$개의 연속된 0 혹은 연속된 1을 존재하게 하는 테스트 패턴은 $b$번째 테스트 패턴으로서의 할당이 불가능하다. 지연 고장이 검출 가능한 올바른 테스트 패턴이 선택되었으면 $b$번째 테스트 패턴으로 해당 패턴을 할당한다. $b$+1번째 테스트 패턴부터는 다시 가장 적은 비용을 가지는 테스트 패턴을 단순히 할당하고, 재정렬이 완료된 테스트 패턴 $T_R$을 리턴함으로써, 제안된 테스트 패턴 생성 알고리즘을 종료한다.

그림 3에서 보인 1단계가 완료된 예제 테스트 패턴 셋을 제안된 기법에 따라 GB 영향과 지연 고장 검출을 모두 고려하여 재정렬을 거치는 과정을 그림 7의 (a)에서 나타내었다.

그림 7의 (a)는 $k$=13, $s$=5인 연결선을 위해 GB 영향과 지연 고장을 함께 고려한 제안된 기법에 의해 테스트 패턴을 재정렬하는 과정을 나타내었다. 그림에서 보는 바와 같이 제안된 알고리즘을 거친 테스트 패턴 셋은, 그림 3과는 달리 첫 테스트 패턴부터 테스트 패턴의 총 개수 8개의 절반인 4번째 비트포지션까지, 어떠한 코드워드도 연속된 0 혹은 연속된 1을 가지지 않는다. 따라서 전체 코드워드는 모두 0에서 1, 1에서 0으로의 천이를 각각 적어도 1개씩 가지게 되어, 보드 레벨 연결선의 지연 고장 테스트를 완벽하게 수행할 수 있다. 본 논문에서 제안하는 테스트 패턴 생성 기법은 II장에서 소개된 [7]의 휴리스틱 알고리즘에 기반했으며, 테스트 패턴이 가질 수 있는 최대, 최소 개수는 기존의 알고리즘과 동일하다[7].




(a)

| | $p_7$ | $p_6$ | $p_5$ | $p_3$ | $p_4$ | $p_2$ | $p_1$ | $p_0$ |
|---|---|---|---|---|---|---|---|---|
| $c_0$ | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| $c_1$ | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| $c_2$ | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| $c_3$ | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| $c_4$ | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| $c_5$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| $c_6$ | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| $c_7$ | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| $c_8$ | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| $c_9$ | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| $c_{10}$ | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| $c_{11}$ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| $c_{12}$ | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | | 5 | 5 | 5 | | 5 | |

Hamming Distance

(b)

그림 7. 제안된 지연고장 모델을 고려한 테스트 패턴 재정렬 기법 (k=13, s=5)

Fig. 7. Proposed Test Pattern Reordering Method for Delay Fault Model (k=13, s=5).

재정렬된 테스트 패턴 셋을 다시 정리하여 해밍 거리를 계산한 과정을 그림 7의 (b)에 나타내었다. 그림에서 보는 바와 같이 음영 처리된 경우가 3개로 늘어나, 기존의 기법에 비해서 추가해야 할 테스트 패턴의 개수가 1개 증가한 것을 알 수 있다. 이는 최소 비용으로 재정렬할 경우에 반해서 지연고장 테스트를 위해서 불가피하게 테스트 패턴간의 천이 개수가 향상되었기 때문이다.

그림 8은 제안된 기법에 의해 생성된 테스트 패턴 셋에 테스트 패턴을 추가 삽입하는 과정이다. 그림 7의 (b)에서 보인 테스트 패턴 셋은 연결선 테스트와 지연 고장 테스트가 가능하며, GB 영향을 최소화시키기 위해서 재정렬 한 것이다. 그러나 여전히 SSOL 계수를 초과하는 테스트 패턴이 존재하기 때문에, 그림 8의 음영 처리된 부분과 같이 더미 테스트 패턴 3개를 각각 삽입하여 GB 영향에 의한 바운더리 스캔의 오동작 문제를 해결한다.

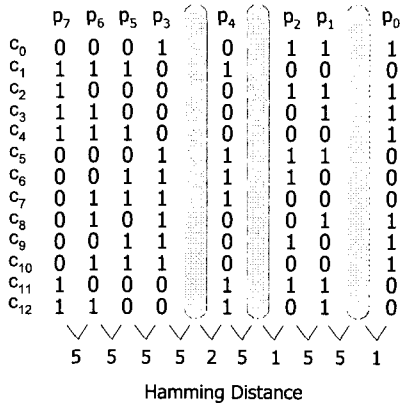



그림 8. 제안된 기법에 의해 생성된 테스트 패턴 셋의 더미 테스트 패턴 추가 삽입

Fig. 8. dummy test pattern insertion to the generated test pattern set from proposed method.

기존의 보드 레벨 연결선 테스트 기법이 연결선 테스트와 GB 영향만을 고려한 것에 비해, 본 논문에서 제안된 테스트 패턴 생성 기법은 연결선 테스트와 GB 영향에 의한 바운더리 스캔의 오동작 방지 뿐만 아니라 지연 고장 검출 기능까지 모두 보장하는 효율적인 알고리즘을 설계하였다.

## Ⅳ. 실험 결과

Ⅳ장에서는 기존의 보드 레벨 연결선 테스트와 GB 영향만을 고려한 기법에 의해 생성된 총 테스트 패턴 수와, 본 논문에서 제안된 지연고장 검출 기능까지 추가한 테스트 패턴 생성 기법의 총 테스트 패턴 수를 비교하였다. 알고리즘은 모두 C로 구현되었으며, 1GHz의 CPU를 이용한 PC에서 실험하였다. 실행시간은 모두 무시할 정도로 적은 시간만이 소요되었다.

본 논문에서 제안된 보드 레벨 연결선을 위한 테스트 패턴 생성 기법은 기존의 기법이 보장하는 기능을 모두 만족시키면서 보드 연결선의 지연 고장까지 모두 테스트할 수 있다. 하지만 본 논문에서 제안된 기법에서 생성된 테스트 패턴의 수는 기존의 기법과 비교하여 큰 차이를 보이지 않았으며, 어떤 경우에는 더 좋은 결과를 보였다. 표 1과 같이 여러 환경에서 두 기법에 의해 생성된 테스트 패턴의 개수를 비교하였다.

실험은 연결선의 숫자가 8000(a), 7000(b), 6000(c), 5000(d) 일 때 SSOL 계수인 $s$값을 각각 변화시키면서 결과를 비교하였다. 각각의 테이블에서 첫 번째 열은 SSOL계수 $s$값을 나타낸다. 두 번째 열과 세 번째 열의 숫자는 보드 레벨 연결선을 테스트하기 위한 총 테스트 패턴의 개수이다. 두 번째 열은 기존의 기법에서 제안된 연결선 테스트와 GB 영향만을 고려하여 생성된 테스트 패턴의 개수이며, 세 번째 열은 본 논문에서 제안하는 연결선 테스트와 GB 영향, 그리고 지연고장 검출 기능도 모두 보장하는 테스트 패턴의 개수이다. 실험 환경에 사용된 모든 경우($k$=8000, 7000, 6000, 5000)에 GB 영향과 지연 고장을 고려하지 않고 연결선 테스트만을 위한 테스트 패턴의 수는 $p(k)$개이므로 총 26개의 패턴만을 필요로 한다.

본 논문에서 제안하는 기법은 기존의 연결선 테스트와 GB 영향을 함께 고려한 기법에 지연 고장을 검출할 수 있는 기능을 추가하였다. 하지만 표 1의 (a), (b), (c)의 경우, 기존의 기법과 같거나 비슷한 수준의 테스트 패턴의 수를 필요로 함을 알 수 있다. 하지만 (d)의 경

표 1. 총 테스트 패턴 수의 비교

Table 1. Total Test Pattern Number Comparison.

$k$ = 8000

| SSOL ($s$) | 테스트 패턴 개수 | |
|---|---|---|
| | EXTEST + GB | EXTEST + GB + Delay |
| 400 | 251 | 251 |
| 800 | 126 | 126 |
| 1200 | 101 | 101 |
| 1600 | 76 | 76 |
| 2000 | 51 | 51 |
| 2400 | 51 | 51 |
| 2800 | 51 | 51 |
| 3200 | 51 | 51 |
| 3600 | 51 | 51 |
| 4000 | 26 | 26 |

(a)

$k$ = 7000

| SSOL ($s$) | 테스트 패턴 개수 | |
|---|---|---|
| | EXTEST + GB | EXTEST + GB + Delay |
| 350 | 251 | 252 |
| 700 | 126 | 127 |
| 1050 | 101 | 101 |
| 1400 | 76 | 76 |
| 1750 | 51 | 52 |
| 2100 | 51 | 51 |
| 2450 | 51 | 51 |
| 2800 | 51 | 51 |
| 3150 | 51 | 51 |
| 3500 | 26 | 27 |

(b)

$k$ = 6000

| SSOL ($s$) | 테스트 패턴 개수 | |
|---|---|---|
| | EXTEST + GB | EXTEST + GB + Delay |
| 300 | 227 | 234 |
| 600 | 126 | 127 |
| 900 | 77 | 81 |
| 1200 | 76 | 76 |
| 1500 | 51 | 52 |
| 1800 | 51 | 51 |
| 2100 | 51 | 51 |
| 2400 | 51 | 51 |
| 2700 | 27 | 29 |
| 3000 | 26 | 27 |

(c)

$k$ = 5000

| SSOL ($s$) | 테스트 패턴 개수 | |
|---|---|---|
| | EXTEST + GB | EXTEST + GB + Delay |
| 250 | 227 | 238 |
| 500 | 127 | 126 |
| 750 | 77 | 81 |
| 1000 | 76 | 76 |
| 1250 | 52 | 51 |
| 1500 | 51 | 51 |
| 1750 | 51 | 51 |
| 2000 | 51 | 51 |
| 2250 | 27 | 29 |
| 2500 | 27 | 26 |

(d)

우, 본 논문에서 제안된 기법이 기존의 기법에 비해 더 적은 수의 테스트 패턴만을 필요로 하는 경우도 존재함을 알 수 있다. 이러한 경우가 존재하는 이유는, 기존의 기법에서 NP-난해 문제인 재정렬된 순서를 찾기 위해 이상적인(optimal) 방법이 아닌 탐욕 알고리즘을 사용했기 때문에 기존의 기법이 항상 최적의 결과를 보이지 않기 때문이다. 즉, 본 논문에서 제안된 기법이 재정렬 과정에서 기존의 기법과 다른 경로를 찾음으로써 (d)의 경우에는 기존의 기법보다 더 나은 결과를 보인 것이다.

본 논문에서 제안된 기법은 기존의 기법과 마찬가지로 연결선 테스트 및 GB 영향을 위한 테스트 패턴 생성을 보장하여, 또한 지연 고장 검출이 가능하다. 생성된 테스트 패턴의 수는 기존의 기법에 의해 생성된 테스트 패턴의 수와 비교하여 큰 차이가 없음을 실험결과를 통하여 알 수 있었다.

## V. 결 론

본 논문에서는 GB 영향에 의한 바운더리 스캔의 오동작을 방지하고, 동시에 연결선의 지연 고장 테스트를 완벽하게 수행할 수 있는 효율적인 보드레벨 연결선 테스트 알고리즘을 제안하였다. 제안된 알고리즘은 IEEE 1149.1의 연결선 테스트, GB 영향에 의한 바운더리 스캔의 오동작 방지, 그리고 연결선의 지연고장 검출 능력을 모두 포함한다. 또한 본 논문에서 제안한 기법은 기존의 기법에 비해 연결선의 지연고장 검출 기능을 추가하였지만, 연결선 테스트에 필요한 테스트 패턴의 총 개수는 기존의 기법과 비교하여 큰 차이를 보이지 않았음을 실험결과를 통해서 확인할 수 있었다. 따라서 앞으로 양질의 보드 레벨 연결선 테스트를 위한 테스트 패턴 생성 알고리즘으로 널리 채택되어 사용될 수 있을 것으로 기대된다.

## 참 고 문 헌

[1] *IEEE Std. 1149.1-2001, Test Access Port and Boundary Scan Architecture*, IEEE, 2001.

[2] W. H. Kautz, "Testing of Faults in Wiring Inter-connects," *IEEE Trans. Computers,* vol. 23, no. 4, 1974.

[3] P. Goel and M. T. Mcmahon, "Electronic Chip-in-Place Test," *Proc. Int'l Test Conf.,* 1982.

[4] J. T. de Sousa and P. Y. K. *Cheung, Boundary Scan Interconnect Diagnosis,* Kluwer Academic, 2001.

[5] S Park, T Kim, "A New IEEE 1149.1 Boundary Scan Design for the Detection of Delay Defects", *Design, Automation and Test in Europe Conf.,* 2000.

[6] H. D. L. Hollmann, E. J. Marinissen, B. Vermeulen, "Optimal interconnect ATPG under a ground-bounce constraint,", *Proc. Int'l Test Conf.,* pp. 60-69, 2003.

[7] E. J. Marinissen, R. G. Bennetts, "Minimizing Pattern Count for Interconnect Test under a Ground Bounce Constraint," *IEEE Design & Test of Computers,* Vol. 20, Issue 2, pp. 8-19, 2003.

[8] D. S. Johnson and L. A. McGeoch, "The Trave-ling Salesman Problem: A Case Study," *Local Search in Combinatorial Optimization,* E. H. L. Aarts and J.-K. Lensta, eds., John Wiley & Sons, 1997.

## 저 자 소 개


김 문 준(학생회원)
2000년 숭실대학교 컴퓨터학부 학사 졸업.
2002년 숭실대학교 컴퓨터학과 석사 졸업.
2002년 3월~현재 숭실대학교 대학원 컴퓨터학과 박사과정
<주관심분야: VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>

장 훈(정회원)
1987년 서울대학교 공대 전자공학과 학사 졸업.
1989년 서울대학교 공대 전자공학과 석사 졸업.
1993년 University of Texas at Austin 졸업.
1991년 IBM Inc. Senior Member of Technical Staff.
1993년 Motorola Inc. Senior Member of Technical Staff.
1994년~현재 숭실대학교 컴퓨터학부 부교수.
<주관심분야: 통신, 컴퓨터, 신호처리, 반도체>

이 정 민(학생회원)
2004년 숭실대학교 컴퓨터학부 학사 졸업.
2004년 3월~현재 숭실대학교 대학원 컴퓨터학과 석사과정
<주관심분야: VLSI 설계 및 테스트, 컴퓨터구조, VLSI CAD>