논문 2005-42SD-9-6

시드 병합을 통한 테스트 데이터의 압축방법

(SMC: An Seed Merging Compression for Test Data)

이 민 주*, 전 성 훈*, 김 용 준*, 강 성 호**

(Min-joo Lee, Sung-hun Jun, Yong-joon Kim, and Sumg-ho Kang)

요 약

회로가 커짐에 따라 테스트 데이터양이 증가하고, 테스트 적용시간이 길어지고 있다. 따라서 테스트 데이터양과 테스트 적용시간을 줄이기 위해서, 테스트 데이터의 압축/복원을 위한 새로운 방법을 제안하고자 한다. 제안하는 방법은 시드 벡터를 생성할 때, 압축률을 높이기 위해 무상관비트를 사용하는 XOR 트리에 기반을 두고 있다. 시드 벡터가 생성이 되면, 2비트 길이를 가진 코드를 사용하여 시드를 병합한다. 이렇게 병합된 시드는 1 클럭 시간동안에 재사용될 수가 있어, 테스트 데이터 적용시간을 크게 감소시킬 수 있다. 제안하는 방법의 효율성은 ISCAS '89 벤치 회로에 대한 실험 결과로 알 수 있다.

Abstract

As the size of circuits becomes larger, the test method needs more test data volume and larger test application time. In order to reduce test data volume and test application time, a new test data compression/decompression method is proposed. The proposed method is based on an XOR network uses don't-care-bits to improve compression ratio during seed vectors generation. After seed vectors are produced, seed vectors can be merged using two prefix codes. It only requires 1 clock time for reusing merged seed vectors, so test application time can be reduced tremendously. Experimental results on large ISCAS '89 benchmark circuits prove the efficiency of the proposed method.

Keywords: seed, reuse, merging, prefix codes, XOR network

I. 서 론

작은 회로의 테스트를 위해 사용되던 순차적 테스트 데이터를 사용하는 방법은 회로가 복잡해짐에 따라서 적용이 어려워지고 있다. 특히 SoC의 등장으로 인해 칩을 테스트하기 위한 테스트 데이터양이 증가하면서 SoC테스트를 위한 새로운 테스트 용이화 설계 방법론(DFT: Design For Testability)들이 대두되고 있다[1]. ATE의 채널과 메모리의 한계를 해결하기 위한 방법으로는 정해진 채널 대역폭을 이용하는 방법[2][3]과 BIST(Built in self test)를 사용하는 방법,[4] 테스트 데이터를 압축하는 방법[5,6,7,8,9,10,11]이 있다. 테스트 데이터의 입력과 출력의 수를 이용하여 필요한 채널의 넓이를 줄이는 방법은 ATE와 테스트를 하고자 하는 칩(CUT) 사이에 동적으로 구성될 수 있는 회로를 집어넣어야 하고, 부분적으로 압축된 테스트 데이터들을 복원하기 위해서 여러 개의 확장기나, 압축기를 넣어야 하기 때문에 하드웨어가 매우 커지는 단점이 있다.

테스트 데이터를 압축하는 방법 중 간결화[5,6,7,8]하는 방법은 각각의 테스트 데이터가 찾을 수 있는 고장수를 증가시켜 전체 테스트 데이터수를 줄이는 방법이고, 압축[9,10,11]하는 방법은 각각의 테스트 데이터를 표현하는 테스트 벡터의 비트수를 줄이는 방법이다. 소프트웨어적인 방법으로는 statistical codes[9], Golomb codes[10], frequency directed codes[11]등이 있다. 이러한 소프트웨어적 방법은 높은 압축률을 얻을 수 있지만, 압축한 테스트 데이터를 복원하기 위하여 디코더가 커져야 하고, 디코딩과정이 복잡하다는 단점이 있다.

하드웨어적으로 테스트 데이터를 압축하는 방법[12,13]

* 학생회원, ** 평생회원 연세대학교 전기전자 공학부
(Department of Electrical and Electronic Engineering, Graduate School, Yonsei University)
※ 본 연구는 한국 과학재단 목적기초연구 (과제번호 : R01-2003-000-10150-0) 지원으로 수행되었음.
접수일자: 2005년4월21일, 수정완료일: 2005년8월9일

으로는 LFSR(linear feedback shift register)를 사용한 스캔 체인 아키텍처[12]가 있다. LFSR은 임의의 테스트 데이터를 채우기 위해서 일정한 길이를 가진 시드를 저장하는 방법이다. 시드의 길이가 긴 경우, 시드를 짧은 길이의 여러 개의 시드로 나누어, 스캔 체인의 입력으로 사용하는 multiple LFSR 방법도 있다[13]. Multiple LFSR방법은 LFSR방법보다 시드 벡터의 길이를 줄일 수 있지만, 디코더가 매우 커지게 된다. 이렇게 LFSR에 기반을 둔 방법들[12,13]은 일정한 주기시간이 지나면 동일한 패턴이 생성되고, 또 테스트 적용시간이 오래 걸리는 문제점이 있다.

앞서 언급한 기존의 압축 방법들은 소프트웨어적 방법이나 하드웨어적 방법에 무관하게 주어진 회로의 채널의 대역폭 문제를 극복할 수 없다. 이러한 채널의 대역폭 문제를 극복하는 방법으로, linear XOR 트리를 사용하는 압축방법이 제안되었다[14][15]. 그 중 SCC 압축방법[14]은 linear XOR mapping 트리를 사용하여 회로 내의 스캔 체인을, 외부 테스터에 있는 회로 내의 스캔 체인보다 적은 수의 스캔 체인으로 회로 내의 스캔 체인을 채우는 방법이다.

이러한 SCC 방법은 linear XOR mapping 트리를 사용하는 과정에서 트리의 입력과 출력의 수에 따라 압축률이 고정되는 문제점이 있다. 이러한 문제점을 해결하기 위해서 외부 테스터에 있는 회로 내의 스캔 체인보다 적은 수의 스캔 체인, 즉 시드 벡터에서 특정 비트를 다시 다음의 시드 벡터에 사용하고자 하는 seed vector overlapping 방법[15]이 제안되었다. 하지만 seed vector overlapping 방법은 다음의 시드 벡터에 사용되어야 하는 비트를 계산하는 과정과 다시 사용될 비트를 컨트롤하기 위한 시드 벡터와는 별도로 추가로 컨트롤 비트가 필요하다.

본 논문에서는 정해진 채널의 대역폭 문제를 극복하면서 테스트 적용시간을 줄일 수 있는 방법을 제안하고자 한다. 제안하는 SMC (seed merging compression) 방법은 XOR 트리[14,15]를 사용한 방법들의 단점을 보완하는 방법이다. SMC 방법은 XOR 트리를 사용하여 시드 벡터를 생성 한다. 그 후, 시드 벡터를 병합을 통하여 압축한다. 병합된 시드 벡터는 1 클럭 시간을 사용하여 다시 스캔 체인에 사용될 수 있기 때문에 테스트 적용시간을 줄일 수 있다. 또 시드 벡터를 압축할 수 있기 때문에 XOR 트리의 사용으로 인하여 발생하는 낮은 압축률 문제를 해결할 수 있다.

Ⅱ. XOR 트리

LFSR을 사용한 압축방법들은 높은 압축률을 얻을 수 있지만, 하드웨어가 크고, 테스트 데이터 적용시간이 길다는 문제점을 지니고 있다. 따라서 LFSR을 사용했을 때와 비슷하게 높은 압축률을 얻고 하드웨어 오버헤드를 줄일 수 있는 새로운 방법이 필요하다.

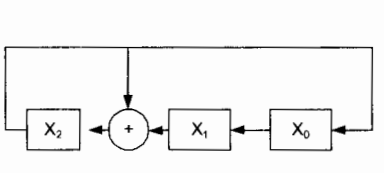


그림 1. LFSR의 동작 예
Fig. 1. An LFSR example.

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} O_0 \\ O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \\ O_6 \end{bmatrix} \quad (1)^*$$

그림 1의 LFSR은 일정한 주기 시간 동안 XOR 연산을 하면서, 테스트 데이터를 생성한다. 그림 1의 LFSR이 생성할 수 있는 테스트 데이터를 행렬 형태로 나타내면 식 1과 같다. 식 1에서 알 수 있듯이, LFSR방식에서는 X_0, X_1, X_2라는 3개의 입력 값이 다항식에 적용되어, 6개의 테스트 데이터 값을 구한다. 예를 들어 LFSR의 시드, 즉 X_0, X_1, X_2가 [1 1 0]이라면, LFSR은 6클럭 시간동안 각각의 클럭에서 테스트 벡터 O_0, O_1, O_2, O_3, O_4, O_5, O_6를 [0 1 0 0 1 1 1] 순서로 생성한다. 여기서 알 수 있듯이, LFSR방법은 순차적 테스트 데이터만을 생성 할 수 있기 때문에, 출력의 확장을 위해서는 테스트 데이터 길이만큼의 시간이 걸리게 된다.

XOR 트리 방식은 임의의 LFSR이 만들 수 있는 모든 출력 데이터 값을 구할 수 있도록 XOR 게이트를 사용하여 트리를 구성하는 방법이다. 그림 2의 XOR 트리를 사용하여 복원된 테스트 데이터는 그림 1의 LFSR을

* 결과 값을 구하는 과정에서, 행렬의 곱셈은 실제적으로는 XOR 연산을 의미한다.

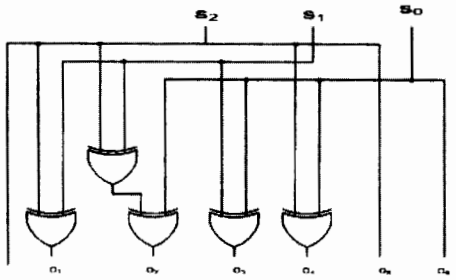


그림 2. 복원 하드웨어
Fig. 2. Decompression hardware.

사용하여 복원된 테스트 데이터와 동일하다.

LFSR방식에서 나올 수 있는 테스트 패턴을 모두 생성하도록 XOR 트리를 구성하려면, XOR 트리의 하나의 출력 값을 위해 몇 개의 XOR 게이트를 사용할지 결정해야 한다. 테스트 데이터(T_D *벡터*)에서 0/1로 명기된(specified) 비트 수가 24비트 이하인 경우, 하나의 XOR 트리의 출력 값을 위해 4개의 XOR 게이트를 사용하여 입력 값을 결정한다면, LFSR을 사용하여 복원하는 경우에 비해서 약 95%의 정확도를 가지고 테스트 패턴을 생성할 수 있다. 따라서 본 논문에서는 하나의 출력 값에 사용되는 XOR 게이트의 수를 4개로 할 것을 제안한다. 이럴 시 1706개의 XOR 게이트를 사용하는 LFSR 기반의 복원 하드웨어에 비하여, 384개의 XOR 게이트만 사용하기 때문에 하드웨어 오버헤드를 75%로 감소시킬 수 있다.[16]

테스트를 적용하기 위한 클럭 시간은 그림 1의 LFSR방식에서는 6클럭 시간이 필요로 하였지만, 그림 2의 XOR 트리에서는 오직 2클럭 시간만이 필요하기 때문에 테스트 적용 시간 면에서도 XOR 트리방식을 사용하는 것이 좋다.

LFSR 방법에서 하나의 다항식을 사용할 시, 테스트 데이터를 생성할 수 있는 시드 벡터가 존재하지 않을 가능성을 10^{-6} 이하로 낮추기 위하여, ATPG에 의해 생성된 테스트 데이터 중 가장 0/1로 명기된 비트 수가 많은 테스트 데이터의 0/1로 명기된 비트 수 길이보다 20비트 길게 시드 벡터의 길이를 결정하는 점에 착안하여, 마찬가지로 XOR 트리의 출력이 ATPG에 의해 생성된 테스트 데이터(T_D *벡터*)의 0/1로 명기된 비트 수보다 20비트 큰 수가 되도록 한다. 이렇게 하면 LFSR 방식과 마찬가지로 이미 구성된 XOR 트리를 사용하여 원래의 테스트 데이터(T_D *벡터*)를 복원하지 못할 가능성을 10^{-6}이하로 낮출 수가 있다.

XOR 트리를 사용 시, XOR 트리의 입력과 출력수가 정해지고, 이로 인해 압축률이 고정되는 문제점이 있다. 제안하는 SMC 방법은 XOR 트리의 압축률문제를 해결하는데 초점을 두고 있다.

Ⅲ. SMC를 위한 압축 코드

테스트 데이터(TD 벡터)의 많은 부분이 무상관(don't care) 비트이기 때문에, Ⅱ장에서 제안한 XOR 트리를 사용하여, 테스트 데이터를 압축하여 생성된 시드 벡터(Ts 벡터)도 많은 부분이 무상관 비트로 이루어진다. 제안하는 SMC 방법은 시드 벡터(Ts 벡터)를 압축하기 위해서 2비트길이를 가진 prefix codes를 사용하는 방법이다. prefix codes에 관한 내용은 1절에 나타내었다. prefix codes에 의하여, 압축된 테스트 데이터(Tns 벡터)는 그 자신의 길이와 무관하게, 1클럭 시간만을 사용하여 테스트를 위해 사용 될 수 있다. 테스트 적용 시간을 계산하는 방법은 2절에 나타내었다.

3-1. Prefix code의 사용

XOR 트리를 사용하여 복원 구조를 생성하면 XOR 트리의 입력과 출력의 개수가 정해지게 되고, 이로 인해 압축률이 고정된다. 시드 벡터(T_s *벡터*)를 압축하기 위해서 우선적으로 ATPG에 의하여 생성된 테스트 패턴(T_D *벡터*)를 살펴 볼 필요가 있다. ATPG에 의하여 생성된 테스트 패턴(T_D *벡터*)의 많은 부분들은 무상관 비트로 이루어져 있다. 이로 인해 기존방법들은 XOR 트리의 시드 벡터를 계산하기 전에 ATPG에 의해 생성된 시드 벡터의 무상관 비트를 임의로 맵핑하여 계산한다. 이런 방식으로 시드 벡터를 구하게 되면 모든 비트가 0/1로 명기된 비트가 되어 더 이상의 압축이 거의 불가능 하다. 따라서 SMC 방법에서는 무상관 비트를 0/1로 명기된 비트로 맵핑하지 않고, 시드 벡터(T_s *벡터*)를 생성한다.

제안하는 SMC 방법으로 시드 벡터(T_s *벡터*)를 압축하기 위해서는, XOR 연산을 다르게 한다. 즉, 4개의 XOR 로 이루어진 형태에서 결과 값이 1이면 입력 값들은 [1, 0, 0, 0] 혹은 [1, 1, 0, 1]의 형태로 4비트가 임의의 순서로 정해진다. 하지만 시드 벡터를 통합할 시 두 시드 벡터(T_s *벡터*)사이의 상충을 가능한 줄이기 위해서 출력 값이 1이면, 입력 값을 [1, 0, 0, 0] 순서로 정한다. 4개의 XOR으로 이루어진 형태에서 결과 값이 0이면, 입력 값으로 [1, 1, 1, 1], [0, 0, 0, 0],[1, 1, 0, 0]의 형태의 4비트가 될 수 있다. 하지만 출력 값이 1인 경우와

```
PROCEDURE SMC_method(CIRCUIT)
1.  FOR using XOR network
2.    if(scan_number > Num_pis) {
3.        add Test_cube_list='x';
4.    else
5.        break;
6.        }
7.  FOR generate seed vector (Ts vector)
8.  FOR generate  Tns vector using prefix code
9.     calculate prefix code {
10.      case no seed vector merging
11.        prefix code = '00';
12.      case two seed vector
13.        prefix code = '01';
14.      case three seed vector
15.        prefix code = '10';
16.      case four seed vector
17.        prefix code = '11';
18.          }
END SMC_method()
```

그림 3. SMC 방법의 알고리듬
Fig. 3. The algorithm of SMC method.

마찬가지로 제안하는 SMC 방법의 효율을 증가시키기 위해서, 입력 값으로 [0, 0, 0, 0]로 4비트를 결정 한다.

이러한 XOR 연산과정으로 구한, 시드 벡터(T_s *벡터*)를 압축하기 위해 2 비트의 길이를 가진 prefix codes를 사용한다. 두 시드 벡터 사이에 상충이 발생하여 압축이 전혀 불가능한 경우에 prefix code로 '00'을 부여하고, 두 시드 벡터가 압축이 되었지만, 세 번째 시드 벡터와는 상충이 발생하여 압축이 불가능한 경우에는 prefix code로 '01', 세 개의 시드 벡터가 압축이 되었지만 네 번째 시드 벡터와 상충이 발생하여 압축이 불가능한 경우는 prefix code로 '10', 네 개의 시드 벡터가 압축된 경우에는 다섯 번째 시드 벡터와의

상충 여부와는 무관하게 prefix code로 '11'을 부여하다. 이러한 SMC 압축 과정의 전체 알고리듬은 그림 3과 같다.

시드 벡터(T_{ns} *벡터*)가 압축하는 과정의 예는 그림 4에 나타내었다. 그림 4(a)의 테스트 데이터를 제안하는 방법대로 압축을 하게 되면 ①에서는 첫 번째 시드 벡터와 두 번째 시드 벡터는 압축이 가능하지만 세 번째 벡터와 두 번째 시드 벡터 사이의 상충이 발생하기 때

①X X X X X X X X X X X X X X X X X X X X
X X X X X X X 0 1 1 1 1 1 1 1 1 1 1 1 X
②1 1 1 1 1 1 1 1 X X X 1 X X X X X X X X
X X X X X X X X 0 1 1 1 1 1 1 1 1 X 1 X
1 1 1 1 1 1 1 1 X 1 1 X X X X X X X X
X X X X X X X X 0 1 X X X 1 X 1 X X X X
③X X X X X X X X X 1 1 1 0 1 X 0 X 1 X X
0 1 1 1 1 1 0 0 0 1 1 1 0 X 1 X X X X X
X X X X X 1 0 0 0 X X 1 0 1 X X 1 X X X
④1 X X 1 1 1 X X X X X X X X X X X 1 1 1

(a) 임의의 테스트 데이터
(a) Example of test data.

0 1 X X X X X X X 0 1 1 1 1 1 1 1 1 1 1 X
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 X 1 X
1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 1 1 0 1 1 X X
0 0 1 X X 1 1 1 X X X X X X X X X X X 1 1 1

(b) SMC 압축 알고리듬을 적용한 임의의 테스트 데이터
(b) Test data compression example using SMC.

그림 4. 제안하는 압축 알고리즘의 적용 예
Fig. 4. Example of SMC method application.

문에 prefix code '01'을 사용하여 첫 번째 시드 벡터와 두 번째 시드 벡터를 압축한다. ②에서는 첫 번째 시드 벡터에서 네 번째 시드 벡터까지 상충이 발생하지 않기 때문에 prefix code '11'을 사용하여 압축을 하게 된다. ③에서는 네 번째 벡터에서 상충이 발생하기 때문에 prefix code '10'을 사용하여 세 개의 시드 벡터를 압축한다. ④에서는 더 이상의 시드 벡터가 없기 때문에 압축이 일어나지 않았다는 의미로 prefix code '00'을 부여한다. 이렇게 되면 그림 4(a)의 200 비트가 그림 4(b)의 88 비트로 압축이 된다.

3-2. 테스트 적용 시간

테스트 적용시간과 테스트 데이터양을 구하기 위해서는 다음과 같은 변수들을 사용해야 한다.

- T : 테스트 패턴 수
- M : 스캔 체인의 수
- S : 스캔 셀의 수
- N : 스캔 셀의 수
- D : 테스트 적용 시간
- V : 테스트 데이터양

테스트 적용시간은 테스트 패턴수와 스캔 체인의 길이에 의해서 결정이 된다. 여기서 스캔 체인의 길이는 $\frac{S}{N}$로 구할 수 있다. 테스트 적용시간은 식 2[16]에서 알

수 있듯이, 테스트 패턴과 스캔 셀의 수에 비례한다. 테스트 데이터양도 테스트 적용시간과 마찬가지로 테스트 패턴수와 스캔 셀의 수에 의해서 결정이 된다. 테스트 데이터양은 식 3[16]에 나타내었다.

$$D = (T+1) \times \left[\left(\frac{S}{N}\right)+1\right] \quad (2)$$

$$V = T \times S \quad (3)$$

XOR 트리를 사용하여 실제로 존재하는 스캔 체인의 수보다 큰 내부 스캔 체인($M>N$)를 사용하게 되면, $\frac{S}{N}$로 정해졌던 스캔 체인의 길이가 $\frac{S}{M}$로 줄어들게 된다.

하지만 M으로 스캔 체인의 길이를 임의로 정하게 되면, ATPG과정에서 M으로 늘어난 스캔 체인을 채우기 위해서 테스트 데이터의 수는 T_c로 원래의 스캔 체인을 채우기 위한 T보다는 약간 증가하게 된다. XOR 트리를 사용했을 때의 테스트 적용시간과 테스트 데이터양은 식 4[16]와 식 5[16]와 같다.

$$D_c = (T_c+1) \times \left[\left(\frac{S}{N}\right)+1\right] \quad (4)$$

$$V_c = T_c \times \left[\frac{S}{M}\right] \times N \quad (5)$$

식 4와 식 5에서 알 수 있듯이, XOR 트리를 사용하면, 기존의 방법들보다 테스트 적용시간이 줄어들고 테스트 데이터양이 줄어들게 된다. 제안하는 방법에서 사용되는 스캔 체인의 수는 XOR 트리를 사용했을 때와 동일하다. 하지만 SMC 방법으로 압축된 테스트 데이터(T_{ns} 벡터)는 테스트 데이터(T_{ns} 벡터)의 길이와 무관하게 1클럭 시간동안 그대로 사용할 수 있다. 또 테스트 데이터를 압축하면서, 스캔 체인의 길이가 $\frac{S}{M}$에서 $\frac{S}{M} - N_{reuse}$로 줄어들게 된다. 여기서 N_{reuse}은 테스트 데이터(T_{ns} vector)를 다시 사용하는 횟수, 즉 SMC에서 사용한 prefix codes의 총합과 동일하다. SMC 방법의 테스트 데이터 적용시간과 테스트 데이터양은 식 6과 식 7과 같다.

$$D_F = (T_c+1) \times \left[\left(\frac{S}{M}\right) - N_{reuse} + 1\right] \quad (6)$$

$$V_F = T_c \times \left[\left(\frac{S}{M}\right) - N_{reuse}\right] \times N \quad (7)$$

Ⅳ. 제안하는 복원 하드웨어 구조

앞에서 제안한 압축 방법으로 압축된 테스트 데이터를 테스트 시에 이용하기 위해서는 ATE이 자체에 압축을 풀어주는 하드웨어가 내장되어 있거나 SoC 내부에 압축을 풀어주는 하드웨어가 내장되어야 한다. 일반적으로 SoC 내부에 압축을 풀어주는 복원 구조가 설계하기가 훨씬 수월하기 때문에 본 논문에서는 SoC 내부에 압축을 풀어주는 복원 방법을 사용할 것이다. SoC에 내장되는 일반적인 복원 구조는 디코더와 ATE사이의 신호를 통제하는 컨트롤러로 이루어진다. 제안하는 복원 구조는 그림 5와 같다.

그림 6의 제안한 디코더는 압축된 테스트 데이터(T_{ns} vector)를 시드 벡터(T_s vector)로 압축을 풀어주는 역할을 한다. 이 디코더는 24비트의 플립플롭과 FSM으로 구성되어 있다. FSM의 입력으로 bit_in으로 이 비트 길이를 가진 prefix codes가 들어오고, load신호는 FSM에서 새로운 prefix codes를 보내라는 신호이다. en 신호는 FSM과 플립플롭에 신호가 다 채워졌는지 여부를 알려주는 신호이다. 24 비트의 플립플롭의 bit_in신호는 T_{ns} 벡터에서 prefix codes를 제외한 나머지 비트들을 입력으로 받는다. FSM 출력 reset신호는 24비트의 플립플롭을 초기화 시킬 때 사용한다. FSM 의 또 다른 출력 in 신호는 24비트의 플립플롭에 저장되어 있는 24비트의 데이터를 T_s 벡터로 출력시킬 때 사용한다. FSM동작은 다음과 같다.

- FSM은 bit_in 신호를 사용하여, FSM에 2비트와, 24 bit 플립플롭에 24비트, 총 26비트를 집어넣는다. 26비트가 다 채워 질 때까지 en신호는 high 상태를 유지한다.
- FSM에 해석된 2비트의 prefix code에 의해서 24비트의 플립플롭에 채워진 24비트의 데이터를 재사용할 때는 in신호가 하이가 된다.
- FSM에 해석된 2비트의 prefix code에 의해서 24비트의 플립플롭에 채워진 24비트의 데이터를 재사용 되지 않을 시는 reset신호, load신호가 high가 되고, en신호는 로우상태이다.

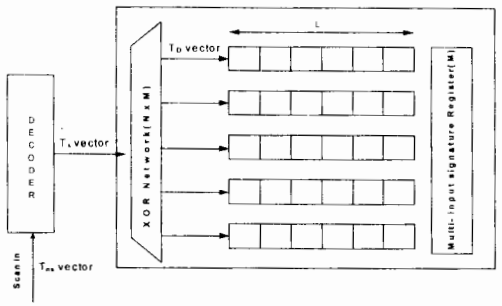


그림 5. SMC 압축 알고리듬을 위한 복원하드웨어 구조
Fig. 5. Decompression hardware structure for SMC.

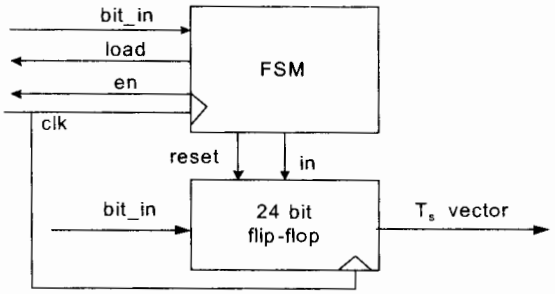


그림 6. 디코더의 블록 다이어그램
Fig. 6. Decoder block diagram.

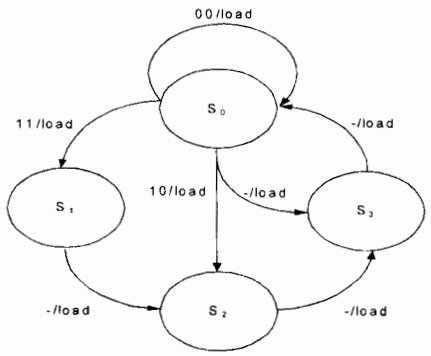


그림 7. 디코더의 상태 전이 다이어그램
Fig. 7. Decoder state Diagram.

이 FSM은 그림 7과 같은 4개의 상태를 지닌다. prefix code가 '11'일 때 S_0상태이고, load신호는 4 클럭 시간동안 발생을 한다. 여기서 load신호가 한 클럭 시간동안 발생하면 S_0상태에서 S_1상태로 된다. 다시 한 번 load가 되면 S_2상태로 다시 load가 되면 S_3으로 다시 load가 되면 S_0상태로 가게 된다. 그 동안 load신호는 로우를 발생하여 새로운 비트들이 들어오지 않는다. prefix code가 '10'이면 S_0상태에서, load신호를 3클럭 시간동안 내보내면서 S_2, S_3상태를 거쳐 S_0상태로 돌아온다. prefix code가 '01'이면 load신호를 2 클럭 시간동안 내보면서 S_0상태에서 S_3상태를 지나 S_0상태로 된다. prefix code가 '00'이면 load 신호를 1 클럭 시간동안 내보면서 S_0상태를 유지한다.

S_0상태는 디코더가 준비되어 bit_in으로 T_{ns} 벡터가 들어 올 수 있는 상태이다. FSM의 in신호로 24비트의 플립플롭들은 T_s 벡터로 내보내서 XOR 트리를 통하여 원래의 테스트 데이터로 복원된다.

V. 실 험

상업적인 툴을 고장 시뮬레이션에 사용하기 위해서는, 핵심 모듈을 수정해야 하기 때문에, 호프로 검증된 자체 툴을 사용하여, 1.2GHz의 썬 블레이드 2000 워크스테이션에서 제안된 압축알고리즘의 성능 평가를 하였다.

제안된 방법은 XOR 트리의 출력 수, 즉 스캔 체인의 수를 줄이는 데 주안점을 두고 있다. 테스트 패턴(T_D 벡터)는 deterministic pattern으로 간결화를 하여 테스트 데이터를 생성하였다. 그 결과 각 회로에서 생성되는 테스트 패턴수와 고장 수는 표 1에 나타내었다.

효율적인 XOR 트리를 구성하기 위해서, 실험적으로 XOR 트리의 입력과 출력 수를 바꾸어 가면서 s13207과 s15850과 s38417과 s38584의 실험을 하여 결과를 표 2, 3에 나타내었다. 또 SCC 방법[14]에서는 테스트 벡터를 생성할 때, 무상관비트를 임의로 맵핑하였지만, 본 논문에서는 제안하는 방법과 테스터 데이터 압축률을 비교하기 위해, 무상관비트를 맵핑하지 않았다.

SCC 방법은 임의로 맵핑한 테스트 데이터를 사용하는 경우에도, 평균적으로 86%정도의 압축률을 가지는 경향이 있다. 이에 따라 본 논문에서도 비슷한 압축률을 가지는 경향을 보이게 되고, SCC 방법으로 생성된 시드를 압축하는 SMC 방법도 각 회로에 따라 비슷한 압축률을 가지게 된다. 따라서 각 회로에서 XOR 트리의 입력과 출력에 따라 각 회로에서 압축률이 비슷한 경향을 보이고 있다.

표 2의 결과를 보면 XOR 트리로 20비트의 입력과 300비트의 출력을 가질 때가 압축률이 가장 우수하다.

표 1. 테스트 패턴수와 고장 수
Table 1. The number of test patterns and faults.

회로	고장 수	테스트 패턴 수
s13207	9809	3196
s15850	11725	3304
s38417	31180	10611
s38584	36303	13774

표 2. SMC 압축 알고리듬을 적용한 테스트 데이터 압축률

Table 2. The test data compression ratio using SMC.

회로	Scan Chains Number	XOR Input/ Output	SCC[14] (%)	SMC (%)
s13207	7	24/100	76.00	93.50
	5	24/150	76.00	93.50
	4	24/200	88.00	96.75
	3	20/300	93.33	98.16
	3	24/300	92.00	97.83
s15850	7	24/100	76.00	93.50
	5	24/150	76.00	93.50
	4	24/200	88.00	97.75
	3	20/300	93.33	98.16
	3	24/300	92.00	97.83
s38417	17	24/100	76.00	93.49
	9	24/200	88.00	96.74
	6	20/300	92.00	97.83
	6	24/300	80.00	94.49
s38584	NA	24/100	NA	NA
	8	24/200	88.00	96.75
	5	20/300	93.33	98.16
	5	24/300	92.00	97.83

NA : 스캔 셀의 수가 XOR 트리의 출력보다 커서, 시드 벡터가 생성되지 않는 경우

하지만 128비트의 길이를 가진 테스트 데이터는 확률적으로 0/1로 명기된 22비트를[16] 갖기 때문에, 모든 0/1로 명기된 비트가 복원될 수 있도록 하기 위해서 XOR 트리의 입력을 적어도 24비트로 해야 한다. XOR 트리의 입력을 24비트로 한 후 회로의 스캔 체인의 수를 100, 200, 300으로 하여 실험을 한 결과, XOR 트리의 출력수가 300비트일 때가 가장 압축률이 우수한 것을 알 수 있다.

s13207를 살펴보면 XOR 트리의 출력이 100인 경우 하나의 고장을 위해서 필요한 비트수는 700비트이지만, 출력이 300일 때는 하나의 고장을 위해서 필요한 비트수는 900비트로 200비트가 증가한다. 하지만 오히려 출력이 300일 때가 테스트 적용시간이 더 짧은 것을 알 수 있다. 그 이유는 제안하는 방법이 압축된 시드 벡터를 1 클럭 시간동안에 시드 벡터의 길이와 무관하게 테스트를 위해 사용할 수 있기 때문이다. s15850 와 s38417회로와 s38584의 회로에서도 동일한 결과를 얻을 수 있다.

표 3. SMC 압축 알고리듬을 적용한 테스트 적용 시간

Table 3. The test application time using SMC.

회로	XOR Input/ Output	SCC[14] TAT (cycles)	SMC TAT (cycles)	Reduction (%)
s13207	24/100	178984	44760	74.99
	24/200	63926	15990	74.89
	20/300	38356	9596	74.98
	24/300	38216	9576	74.98
s15850	24/100	185032	46272	74.99
	24/200	66085	16530	74.89
	20/300	39652	9920	74.98
	24/300	38356	9596	74.98
s38417	24/100	3246984	811782	74.99
	24/200	955000	238770	74.99
	20/300	445669	111426	74.99
	24/300	435436	104314	74.99
s38584	24/100	NA	NA	NA
	24/200	991737	247950	74.99
	20/300	445669	111426	74.99
	24/300	413226	103314	74.99

NA : 스캔 셀의 수가 XOR 트리의 출력보다 커서, 시드 벡터가 생성되지 않는 경우

표 3을 살펴보면, XOR 트리의 입력을 24비트로 한 후 회로의 스캔 체인의 수를 100, 200, 300으로 하여 실험을 하였을 시, XOR트리의 출력수가 300비트 일 때가 테스트 적용 시간이 가장 짧은 것을 알 수 있다. 다시 말해서 스캔 체인이 긴 회로, 즉 큰 회로일수록 제안하는 압축 방법이 능률적이라는 것을 알 수 있다. 따라서 SMC 압축 방법은 큰 회로에 최적화되어 있다고 할 수 있다. 또 테스트 적용시간은 XOR 트리의 구성과 무관하게 SCC 방법에 비해 일정한 비율로 줄어드는 것을 알 수 있다. 테스트 적용시간은 식 4와 식 6에 의해, 시드 벡터수와 시드 벡터의 길이의 영향만을 받는다. 따라서 동일한 스캔 체인의 길이를 가질지라도 XOR 트리의 입력과 출력 수에 따라 테스트 적용시간이 변하는 것을 확인 할 수 있다. XOR 트리의 구성과 무관하게 SCC 방법에 비해 일정한 비율로 줄어드는 이유는 다음과 같다. SMC 방법과 SCC 방법은 XOR 트리의 입력수와 출력 수에 맞춰서 테스트 패턴을 생성하기 때문에, 두 방법에서의 스캔 체인의 길이는 동일하고, 시드 벡터의 수도 동일하다. 하지만, SMC 방법에서 SCC 방법으로 구한 시드 벡터를 제안하는 방법으로 압축을 하

기 때문에, 시드 벡터의 길이가 prefix code의 길이만큼 늘어나는 대신, 시드 벡터의 수가 줄어들게 된다. 즉, SMC 방법은 SCC 방법에 비해, 필요한 데이터양이 일정비율로 줄어들게 된다. XOR 트리의 입출력이 24/300일 때와 20/300일 때는 테스트 적용 시간이 유사하다. 그 이유는 24/300으로 사용할 경우, 필요한 비트 수는 줄어들지만, 20/300으로 XOR 트리를 구성했을 때와 스캔 체인의 길이가 동일하고 테스트 패턴 수가 같기 때문이다.

하지만 표 2에서 알 수 있듯이, 시드 벡터를 압축할 때 상충이 일어나는 경우가 각 XOR구성마다 다르기 때문에, 테스트 데이터양이 SCC방법에 비해 줄어드는 비율은 일정하지가 않다. 하지만 시드 벡터를 테스트에 적용할 때, SCC 방법은 구해진 시드 벡터의 길이만큼의 클럭 시간을 필요로 하는 반면, SMC 방법은 병합된 시드 벡터의 길이와는 무관하게 1 클럭 시간만을 사용하여 병합된 시드 벡터를 테스트에 적용 할 수 있어, XOR 트리의 구성과 무관하게 테스트 적용시간을 감소하는 경향을 확인 할 수 있다.

기존에 제안된 여러 방법과의 압축률을 비교하여 표 4에 나타내었다. 표 4의 결과에서 알 수 있듯이, 기존의 제안된 방법들은 회로가 커질수록 압축률이 낮아지는 경향이 있다. 하지만 제안한 SMC 방법은 회로의 크기와 무관하게 기존의 제안된 압축 방법들보다 우수한 압축률을 보인다.

표 5를 살펴 보면, 기존의 발표들보다 테스트 적용시간이 가장 우수한 seed OL방법과 비교하였을 경우, 제안하는 방법이 보다 짧게 걸리는 것을 알 수 있다. 이유는 앞서 SCC방법과 비교하였을 때, 설명하였다.

표 4. 테스트 데이터의 압축결과 비교
Table 4. The comparison of test pattern compression result.

회로	Parallel Serial Full scan [17] (%)	Golomb Coding [10] (%)	Virtual Scan Chains [18] (%)	Geometric [19] (%)	FDR [11] (%)	Seed OL [15] (%)	SMC (%)
s13207	51.86	79.52	64.49	85.74	88.12	89.4	96.75
s15850	14.77	65.71	57.39	74.82	75.79	82.2	96.75
s38417	57.04	69.41	48.21	79.69	80.84	79.2	96.74
s38584	45.69	62.33	57.59	NA	70.52	87.2	96.75

표 5. 테스트 적용 시간의 비교
Table 5. Comparison of test application time.

회로	Seed OL [15]	SMC	Reduction (%)
	TAT(cycles)	TAT(cycles)	
s13207	35677	15990	44.81
s15850	40372	16530	40.94
s38417	560900	238770	42.56
s38584	681097	247950	36.40

Ⅳ. 결 론

큰 회로를 위한 테스트 데이터는 스캔 체인이 길어지는 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해서, 본 논문에서는 실제 칩의 I/O 핀 수를 증가시키지 않고 스캔 셀의 길이를 실제보다 길게 사용하여 스캔 체인의 길이를 줄일 수 있도록 XOR 트리의 사용을 제안한다. 하지만 기존의 논문들이 XOR 트리를 사용하였을 시, XOR 트리의 구성에 따라서 압축률이 고정되는 문제가 있다. 따라서 제안하는 방법은 이러한 문제점을 해결하고자 하였다. 제안하는 방법은 테스트 데이터의 무상관 비트를 0 또는 1로 맵핑을 하지 않는다. 이렇게 압축된 T_s 벡터들도 많은 부분들이 무상관 비트로 이루어져 있기 때문에, T_s 벡터들은 상충을 일으키지 않고 병합될 수 있다. 제안하는 SMC 방법은 4개의 T_s 벡터까지 병합할 수 있도록 2비트의 prefix codes를 사용하는 방법이다.

. 압축된 T_{ns} 벡터들은 1클럭 시간동안 테스트를 위해서 재사용할 수 있기 때문에, 테스트 적용시간을 줄일 수 있다. 또 제안한 2비트의 길이를 가진 prefix codes는 디코딩 과정이 간단하여 이를 디코딩하기 위한 복원 구조도 간단하다. 따라서 SCC 방법에 비하여 하드웨어 오버헤드를 크게 증가하지는 않지만, 고정된 압축률 문제를 해결하면서 보다 높은 압축률을 얻을 수 있고, 테스트 적용 시간을 효율적으로 줄일 수 있다.

ATE 관점에서 보면, 테스트 프로그램의 수정이 필요하지 않아 현재의 테스트 환경에 적합할 뿐 아니라, 점점 회로들이 커져가고 복잡해지 환경에서도 제안하는 방법은 적합한 테스트 압축 방법이라 할 수 있다.

참 고 문 헌

[1] Y. Zorian, S. Dey and M. J. Rodgers, "Test of future system on chips," *Proceedings of*

International Conference on Computer Aided Design, pp. 392 - 400, 2000.

[2] C. V. Krishna and Nur A. Touba, "Adjustable width linear combinational scan vector decompression," *Proceedings of International Conference on Computer Aided Design*, pp. 863 - 866, 2003.

[3] Hideyuki Ichihara and Kozo Kinoshita and Koji Isodono, "Channel width test data compression under a limited number of test inputs and outputs," *Proceedings of the 16th International Conference on VLSI Design*, pp. 329 - 334, 2003.

[4] I. Hamzaoglu and J. H. Patel, "Reducing test application time for Built-In-Self-Test test pattern generators," *Proceedings of IEEE VLSI Test Symposium*, pp. 260 - 267, 1999.

[5] S. Bhatia and P. Varma, "Test compaction in a parallel access scan environment," *Proceedings of Asian Test Symposium*, pp. 300-305, 1997.

[6] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," *Proceedings of International Test Conference*, pp. 283-289, 1998.

[7] I. Pomeranz and S. M. Reddy, "Static test compaction for scan based designs to reduce test application time," *Proceedings of Seventh Asian Test Symposium*, pp. 198 - 203, 1998.

[8] R. Sankaralingam, R. Oruganti, and N. A. Touba, "Static compaction techniques to control scan vector power dissipation," *Proceedings of VLSI Test Symposium*, pp. 35 - 40, 2000.

[9] A. Jas, J. Ghosh-Dastidar and N. A. Touba, "Scan vector compression/decompression using statistical coding," *Proceedings of VLSI Test Symposium*, pp. 25 - 29. 1999.

[10] A. Chandra and K. Chakrabarty, "System-on-a-Chip test data compression and decompression architectures based on golomb codes," *Proceedings of International Conference on Computer Aided Design*, vol. 20, n, 3, pp. 355 - 368, 2001.

[11] Anshuman Chandra and Krishnendu Chakrabarty, "Test data compression and test resource partitioning for System-on-a chip using frequency-directed run-length (FDR) Codes," *IEEE Transactions on Computers, vol. 52*, pp. 1076 - 1088, 2003.

[12] C. V. Krishna, Abhijit Jas, and Nur A. Touba. "Test vector encoding using partial LFSR reseeding," *Proceedings of International Test Conference*, pp. 885 - 893, 2001.

[13] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman and B. Courtois, "Built-in Test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," *IEEE Transactions on Computers*, vol. 44, n. 2, pp. 223 - 233, 1995.

[14] I. Bayraktaroglu and A. Orailogu, "Test volume and application time reduction through scan chain concealment," *Proceedings of The Design Automation Conference*, pp. 151 - 155, 2001.

[15] Wenjing Rad and Ismet Bayraktaroglu and Alex Orailoglu, "Test application time and volume compression through seed overlapping," *Proceedings of The Design Automation Conference*, pp. 732 - 737, 2003.

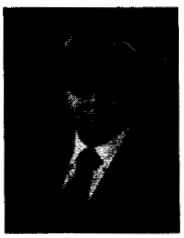
[16] Ismet Bayraktaroglu and Alex Orailoglu, "Concurrent application of compaction and compression for test time and data volume reduction in scan designs," *IEEE Transactions on Computers*, vol. 52, NO. 11, pp. 1480 - 1489, 2003.

[17] I. Hamzaoglu and J. H. Patel, "Reducing test application time for full scan embedded cores," *Proceedings of IEEE International Symposium. Fault-Tolerant Computing*, pp. 260-267, 1999.

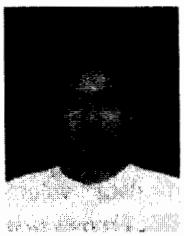
[18] A. Jas, B. Pouya, and N. A. Touba, "Virtual scan chains: A means for reducing scan length in cores," *Proceedings of IEEE VLSI Test Symposium*, pp. 73 - 78, 2000.

[19] A. El-Maleh, S. al Zahir, and E. Khan, "A geometric-primitives-based compression scheme for testing systems-on-a-chip," *Proceedings of IEEE VLSI Test Symposium*, pp. 49 - 54, 2001.

저 자 소 개



이 민 주(학생회원)
2005년 연세대학교 전기전자 공학과 학사 졸업.
2005년 현재 연세대학교 전기 전자공학과 석사과정.
<주관심분야 : DFT, SoC 설계>



김 용 준(학생회원)
2002년 연세대학교 전기공학과 학사 졸업.
2004년 연세대학교 전기전자 공학과 석사과정 졸업.
2005년 현재 연세대학교 전기 전자공학과 박사과정.
<주관심분야 : CAD, Testing>

전 성 훈(학생회원)
2002년 연세대학교 전기공학과 학사 졸업.
2005년 연세대학교 전기전자 공학과 석사과정 졸업.
2005년 현재 연세대학교 전기전자 공학과 박사과정.
<주관심분야 : DFT, CAD>

강 성 호(평생회원)
1986년 서울대학교 제어계측 공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터 공학과 석사 졸업.
1992년 The University of Texas, Austin 전기 및 컴퓨터 공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원
1994년 Motorola Inc. 선임 연구원
2005년 현재 연세대학교 전기전자공학과 교수
<주관심분야 : SoC 설계 및 SoC 테스트>