논문 2004-41SD-11-12

IR 기법을 이용한 효율적인 테스트 데이터 압축 방법

(An Efficient Test Data Compression/Decompression Using Input Reduction)

전 성 훈*, 임 정 빈*, 김 근 배*, 안 진 호*, 강 성 호*

(Sunghoon Chun, Jung-Bin Im, Gun-Bae Kim, Jin-Ho An, and Sungho Kang)

요 약

본 논문에서는 SoC 테스트를 위한 새로운 테스트 데이터 압축 방법을 제안한다. 제안하는 압축 방법은 테스트 데이터 압축을 위해 압축율과 하드웨어 오버헤드를 고려하여 최대 효율을 가지도록 하는데 기초하고 있다. 압축율을 높이기 위해서 본 논문에서는 IR 기법과 MSCIR 압축 코드를 사용하며, 뿐만아니라 이를 위한 사전 작업인 새로운 맵핑 기법 및 테스트 패턴 순서 재조합 방법을 제안한다. 기존의 연구와는 달리 CSR 구조를 사용하지 않고 원래의 테스트 데이터를 사용하여 압축하는 방법을 사용한다. 이렇게 함으로써 제안하는 압축 방법은 기존의 연구에 비해 훨씬 높은 압축율을 가지며 낮은 하드웨어 오버헤드의 디컴프레션 구조를 가진다. ISCAS '89 벤치 회로에 대한 기존의 연구와의 비교로서 그 결과를 알 수 있다.

Abstract

This paper proposes a new test data compression/decompression method for SoC(Systems-on-a-Chip). The method is based on analyzing the factors that influence test parameters: compression ratio and hardware overhead. To improve compression ratio, the proposed method is based on Modified Statistical Coding (MSC) and Input Reduction (IR) scheme, as well as a novel mapping and reordering algorithm proposed in a preprocessing step. Unlike previous approaches using the CSR architecture, the proposed method is to compress original test data and decompress the compressed test data without the CSR architecture. Therefore, the proposed method leads to better compression ratio with lower hardware overhead than previous works. An experimental comparison on ISCAS '89 benchmark circuits validates the proposed method.

Keywords : Test data compression, decompression, input reduction, statistical code, decompression architecture

I. 서 론

칩의 복잡도가 증가함에 따라 정확한 테스팅의 중요성이 점점 더 커지고 있다. 특히 SoC의 등장으로 인해 칩을 테스트하기 위한 테스트 데이터의 양이 증가하면서 SoC 테스트를 위한 새로운 테스트 용이화 설계 방법론(Design For Testability)들이 요구되고 있다[1]. 기존의 ATE(Automatic Test Equipment)로 많은 양의 테스트 데이터가 필요한 SoC를 테스트하기 위해서는 ATE 자체의 채널과 메모리의 한계로 인하여 ATE를 개조하거나 고가의 ATE를 구입해야 한다. ATE의 채널과 메모리의 한계를 해결하기 위해 연구되는 방법론은 크게 두 가지가 있는데 그 첫 번째 방법은 ITRS Roadmap[2]에 나타나 있듯이 BIST(Built In Self Test)를 사용하는 것이다. 하지만 BIST를 이용하기 위해서는 SoC에 들어가는 코어들이 모두 BIST-ready 설계되어야 하며 BIST로 인하여 동작에 많은 영향을 미치기 때문에 이를 고려한 설계를 하는데 많은 노력이 필요하게 된다. 또한 일반적으로 임베디드 코어가 내부를 쉽게 수정할 수 있도록 제공되지 않기 때문에 이러한 BIST 기법을 이용한 해결책은 적절한 해결책이 되기

* 정회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Graduate School, Yonsei University)
※ 본 연구는 한국 과학재단 목적기초연구 (과제번호 : R01-2003-000-10150-0) 지원으로 수행되었음.
접수일자: 2004년2월2일, 수정완료일: 2004년10월19일

어렵다. ATE의 채널과 메모리의 한계를 극복하기 위한 또다른 해결책은 테스트 데이터를 압축하는 방법이다. 이 방법은 이전에 언급한 BIST 방법과는 달리 칩이 정상적으로 동작하는 경우에 대하여 어떠한 영향도 끼치지 않기 때문에 실제적인 SoC 설계에 손쉽게 이용할 수 있다. 또한 ATE에서 SoC를 테스트하기 위해서는 단지 기존의 테스트 데이터를 압축하여 테스트 입력으로 사용하고 이를 칩의 내부의 디코더나 또는 ATE에서 제공하는 디코더를 이용하여 압축된 입력을 원래의 입력으로 바꾸어주기만 하기 때문에 BIST와는 달리 간단하고 효율적인 해결책이라고 할 수 있다.

테스트 데이터를 줄이기 위한 노력들은 몇 년전부터 계속 진행되고 있다. 테스트 데이터들 줄이는 방법으로는 테스트 벡터의 수를 줄임으로서 전체 테스트 데이터를 줄이는 방법[3, 4]과 ATE로 전달되는 테스트 데이터의 양을 줄이는 방법[5], 그리고 칩자체에 디코더 구조를 내장한 방법들[6-13]이 주로 연구되고 있다. 테스트 데이터를 압축하는 알고리듬은 정보의 손실이 없어야 하고 압축된 데이터를 원래의 테스트 데이터로 만들어주는 디코더가 간단해야 한다. 정보의 손실이 없어야 하는 조건은 기존의 정보의 손실이 없는 압축 알고리즘(예를 들어, 허프만 코드, Lempel-Ziv 압축 알고리듬 등)을 이용하면 쉽게 충족될 수 있지만 디코더를 간단하게 만드는 문제는 압축 알고리듬을 개발하는 단계부터 신중하게 고려되어야 한다. Iyengar는 statical coding(SC)[10]을 이용하여 순차회로를 위한 테스트 데이터 압축 방법을 제안하였다. 하지만 이 방법은 주입력의 수가 적은 회로에서만 효율적으로 사용될 수 있다는 단점이 있다. 이러한 단점을 해결하기 위해 새로운 방법이 제안되었다. 이 방법은 주어진 테스트 데이터를 일정한 길이의 블록으로 나누어서 statistical coding이라는 변형된 형태의 허프만 코드를 제안하였다. 이 방법은 디코딩은 간단하지만 블록의 크기가 커질수록 디코더가 복잡해지고 그로 인해서 전체 하드웨어 오버헤드가 커진다는 단점이 있다. Run-Length 코드를 이용한 압축 기법[11]은 실제 테스트에 영향을 미치는 테스트 패턴들은 서로 다른 비트들이 많지 않다는 사실에 기초한 방법이다. 이러한 사실에 기초해서 원래의 테스트 세트 TD를 각 패턴의 유사성을 계산하여 Tdiff로 변환하여 압축을 시도했다. 이렇게 재계산 되어진 Tdiff 테스트 세트를 다시 원래의 테스트 세트로 만들기 위해서는 CSR(Cyclic Scan Regiter)이라는 스캔 체인 구조가 칩 내부에 있어야 한다. 이 경우에는 압축 알고리듬을 디코드하기 위한 하드웨어 이외에 테스트 데이터가 들어가는 스캔 체인 길이만큼의 추가적인 CSR 스캔 구조가 필요하기 때문에 높은 하드웨어 오버헤드를 가질 수밖에 없다. [6], [7] 논문에서는 이러한 Tdiff 테스트 세트를 각각 FDR 코드와 Golomb 코드를 이용하여 압축하는 방법을 제안하였고 그 압축률 역시 굉장히 높다는 결과를 보여주었다. Golomb 코드의 경우 단지 run의 길이에 의존한 압축 코드인 반면에 FDR 코드의 경우는 run의 길이와 빈도수를 고려하여 만들어졌기 때문에 더 좋은 압축률을 보인다. 또한 [13]의 경우 역시 Tdiff 테스트 세트를 이용한 VIHC 코드를 제안하였는데 기존의 Golomb 코드나 FDR 코드와 비슷하거나 높은 압축율을 보이면서 더 적은 하드웨어 오버헤드를 가진다. 그러나 Tdiff를 이용하는 압축 방법은 CSR 스캔 구조를 이용하기 때문에 디코더 이외의 하드웨어를 고려해야 한다.

본 논문에서는 모든 SoC에 쉽게 적용할 수 있도록 하드웨어 오버헤드를 최소화한 온 칩 디코더를 제안하고 이를 사용하는 새로운 테스트 데이터 압축 알고리듬을 제안한다. 이전의 방법과는 달리 CSR 스캔 구조를 이용하지 않고 이에 대한 압축률의 손실을 보완하기 위해 IR(Input Reduction) 방법을 이용하여 적은 하드웨어 오버헤드를 가지고 높은 압축률을 가지는 혼합적인 압축 방법을 제안한다.

Ⅱ. IR (Input Reduction) 기법

IR 기법은 [14]에서 BIST를 위한 테스트 세트를 줄이기 위해 맨처음 제안되었고 본 논문에서는 이 방법을 수정하여 압축율을 높이기 위한 IR 기법을 제안한다. 기존의 IR 기법은 회로 자체를 분석하여 테스트 패턴 생성시 같은 값을 가질 수 있는 가능성이 있는 입력을 줄이는 방법이지만 본 논문에서 제안하는 IR 기법은 주어진 테스트 패턴을 이용하여 기존 테스트 패턴의 고장 검출율의 손실 없이 테스트시에 테스트 입력을 동일하게 사용할 수 있는 입력을 찾는 방법이다.

입력이 N이고 테스트 패턴의 개수가 L인 테스트 데이터 T에서 $v(i,k)$는 입력 i $(0 \le i \le N-1)$ 의 k $(0 \le k \le L-1)$번째 테스트 패턴의 값이라 하면, 입력의 호환 가능은 다음과 같이 정의할 수 있다.

정의 1. 호환가능 : 주어진 테스트 데이터 T에서 두 개의 입력 i와 j는 $0 \le k \le L-1$에서 $v(i,k) = v(j,k)$이

면 호환가능이다. $v(i,k) = X$ 또는 $v(j,k) = X$일 때는 서로 호환가능 또는 역호환가능한 서로 다른 입력의 주어진 값과 상충되지 않아야 한다.

$\overline{v}(i,k)$는 $v(i,k)$의 반대값을 나타낸다고 하면 역호환가능은 다음과 같이 정의할 수 있다.

정의 2. 역호환가능 : 주어진 테스트 데이터 T에서 두 개의 입력 i와 j는 $0 \le k \le L-1$에서 $v(i,k) = \overline{v}(j,k)$이면 역호환가능이다. $v(i,k) = X$ 또는 $v(j,k) = X$일 때는 서로 호환가능 또는 역호환가능한 서로 다른 입력의 주어진 값과 상충되지 않아야 한다.

위와 같이 정의한 호환가능과 역호환가능 입력들을 결정하며, 테스트 시에 필요한 입력을 줄이기 위하여 제안하는 IR 알고리듬은 다음과 같다.

```
input_reduction()
T_D: test set
N : the number of inputs
L : the length of test sequence
input_check : the queue to check whether target input is reduced previously
{
  int i ;
  int j;
  Int k; // sequence k(0 ≤ k ≤ L-1)
  int check;

  initialize_input_check();

  for(i=0; i<N; i++)
   {
    if(input_check[i] == UNIQUE)
      {
       v(i, k); // target input
       for(j=i+1; j<N; j++)
         {
          if(input_check[j] == UNIQUE)
            {
             v(j, k); // comparison input
             compatible_check = is_compatible(v(i,k), v(j,k));
                    // is_compatible function includes the conflict_check function
             switch(compatible_check)
              {
              case COMPATIBLE:
                input_check[j] = COMPATIBLE;
                break;
              case INV_COMPATIBLE:
                input_check[j] = INV_COMPATIBLE;
                break;
              case DONT_CARE:
                input_check[j] = DON'T_CARE;
                break;
              case NONE:
                input_check[j] = UNIQUE;
                break;
              }
            }
          else
            break;
         }
      }
   }
}
```

그림 1. 제안하는 IR(Input Reduction) 알고리듬
Fig. 1. The proposed IR(Input Reduction) algorithm.

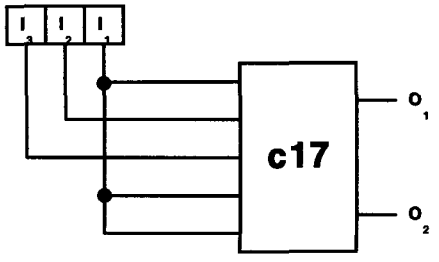

그림 2. IR 기법을 적용한 테스트를 위한 회로 구조
Fig. 2. The circuit under test using the IR scheme.

위의 정의를 이용하여 호환가능한 입력과 역호환가능한 입력을 찾아내서 테스트 데이터를 줄이는 IR 기법을 이용하면 그림 2의 예에서 볼 수 있듯이 전체 테스트 데이터의 정보의 손실없이 쉽게 테스트 데이터 압축을 할 수 있다. c17의 경우 원래 5개의 주입력을 가진 회로이나 IR 기법을 이용하면 테스트 시에는 3개의 입력만을 사용하면 된다. 또한 그림 3에서 보는 바와 같이 호환가능으로 인한 하드웨어 오버헤드는 단지 테스트를 위한 입력선의 길이만 증가할 뿐이며, 역호환가능 입력을 위해서는 입력선의 길이 증가와 NOT 게이트 하나만이 추가된다. 따라서 큰 하드웨어 오버헤드의 증가없이 간단하게 테스트 데이터를 줄이는 것이 가능하다.

정리 1. 만약 두 개 이상의 입력이 호환가능하거나 역호환 가능하다면 그 입력들은 NOT 게이트와 fanout으로 구성되어진 간단한 회로로 구성하여 하나의 입력으로 압축된다.

III. 제안하는 압축 방법

최근에 제안된 압축 방법[6, 7, 13]은 Tdiff를 효과적으로 압축하기 위한 코드로 한정되어 있기 때문에 이러한 테스트 데이터(Tdiff)를 사용하지 않는다면 압축률이 떨어지

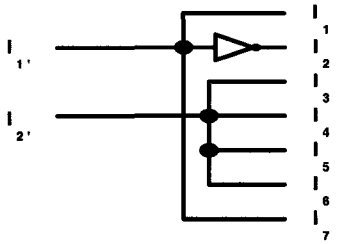

그림 3. 호환가능 및 역호환가능 입력 구조
Fig. 3. The compatible and inverse compatible input architecture.

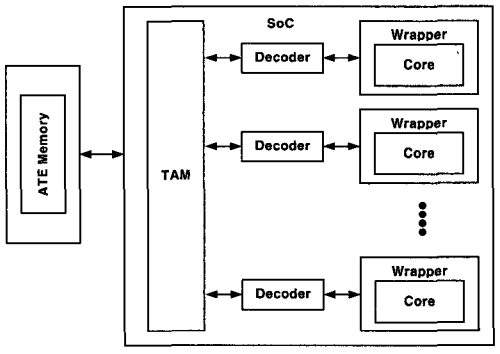

그림 4. 일반적인 SoC 내부의 디컴프레션 구조
Fig. 4. General decompression architecture in a SoC.

게 된다. 또한 앞에서 언급했듯이 Tdiff를 이용하였을 경우 필연적으로 CSR 구조를 사용하기 때문에 하드웨어 오버헤드가 늘어나는 것이 당연하다. 일반적으로 SoC 구조 내부에 내장된 디컴프레션 구조는 그림 4와 같다.

그림 4에서 보는 바와 같이 SoC 내부에 존재하는 각각의 임베디드 코어에 압축된 테스트 데이터를 원래의 테스트 데이터로 만들어 주기 위해서는 각 코어를 위한 디코더가 따로따로 존재해야 한다. 따라서 T_{diff}를 이용한 기존의 압축 방법은 압축 코드를 풀어주는 디코더 외에 각각의 임베디드 코어당 적어도 하나 이상의 테스트 입력의 수만큼의 플립플롭과 하나의 XOR로 구성된 CSR 구조가 추가되기 때문에 전체 SoC에서의 하드웨어 오버헤드가 엄청나게 크다고 할 수 있다. 게다가 디코더의 FSM 과 그것을 제어하는 회로의 크기 자체도 크기 때문에 전체적으로 SoC 내부에 디코더를 내장하기에는 비효율적이다. 그러므로 이것을 해결할 수 있는 좀더 효율적인 압축 방법이 필요하며, 이를 위하여 본 논문에서는 T_{diff}를 이용하지 않고 효과적으로 압축하면서도 간단한 디코더 구조를 가지는 새로운 압축방법을 제안한다.

3.1. 제안하는 압축 코드

기존의 T_{diff}를 이용하지 않는 많은 압축 코드[4, 10]들은 테스트 패턴 자체에 존재하는 많은 X값들을 적절히 특정한 값으로 채워 넣어서 압축을 위한 블록의 출현 빈도수를 높이는 것이 주요 요지였다. 하지만 이전에 제안된 이러한 방법들은 허프만 코드나 SC를 기반으로 하였기 때문에 기본적으로 적절한 크기로 나누어진 코드워드로 이루어진 블록의 출현 빈도수가 비슷하면 압축률이 높아지기보다 오히려 증가하는 단점을 여전히 가지고 있다. 또한 압축을 위한 블록의 개수가 많아질수록 하드웨어 오버헤드가 기하급수적으로 증가하는 단점 역시 내포하고 있다. 따라서 이러한 문제들을 해결하기 위한 방안이 필요하다.

본 논문에서는 이러한 기존의 연구들에서 들어나는 단점을 해결하기 위해 기존의 SC에 기반한 새로운 압축 코드(MSCIR : Modified Statistical Code using Input Reduction)를 제안한다. 제안하는 압축 코드는 기본적으로 가장 출현 빈도수가 높은 4비트 코드워드로 이루어진 블록 하나만을 1비트의 압축 코드로 압축하고 나머지 비트는 2비트로 이루어진 코드워드의 블록으로 원래의 값을 그대로 가지고 있도록 만들어 준다. 여기서 가장 출현 빈도가 높은 블록을 **압축 블록**이라 정의한다. 일반적으로 많은 결정 테스트 패턴들은 많은 X값들을 가지고 있기 때문에 X값을 특정한 값으로 채워주면서 오직 하나의 4비트로 이루어진 블록의 출현 빈도수를 증가시키는 것은 간단하다.

이러한 예로서 그림 5(a)는 완전스캔 구조를 가정한 ISCAS 89 벤치마크 회로 중의 하나인 s13207 벤치 회로의 압축 코드를 적용하기 이전의 첫 번째 테스트 패턴이다. s13207회로의 경우 테스트 패턴을 분석해 보면 '0000' 블록이 가장 많이 출현하며 이를 근거로 하여 그림 5(b)는 이 패턴에서 X값을 '0000' 블록이 많이 출현하도록 '0'으로 채워 넣은 예를 보여준다. 이 예에서 알 수 있듯이 X값을 특정한 값으로 채워 넣음으로서 압축을 위한 특정 블록의 출현 빈도수를 높이는 것은 간단하게 구현할 수 있으면서도 압축률을 높일 수 있는 효율적인 방법이다. 여기에서 압축이 되지 않은 비트를 2비트의 블록으로 할당하는 이유는 다음과 같다. 예를 들어 '000010000010'과 같은 패턴이 있다면 이 경우 모든 블록을 4비트의 코드워드로 할당하고 압축할 특정한 하나의 블록이 '0000'이라면 압축할 수 있는 블록은 단지 1개의 블록일 것이다. 하지만 압축하지 않는 비트를 2비트의 블록으로 할당한다면 '0000'블록이 2개가 된다. 이처럼 압축을 위한 특정한 블록의 출현 빈도수를 증가시켜서 압축률을 높이기 위해 압축하지 않는 블록의 길이는 2비트로 할당하도록 한다. 그림 5의 예에서 알 수 있듯이 지정된 특정 압축 블록의 출현 빈도수를 좀 더 높이기 위해 압축하지 않는 비트를 2비트의 블록으로 할당하는 방법을 사용한다. 이러한 MSCIR 코드의 예를 보이기 위해 그림 5의 패턴을 기반으로 생성한 MSCIR

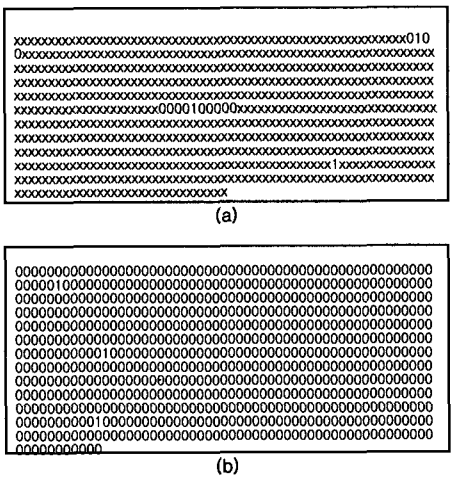

그림 5. 완전 스캔을 가정한 s13207 벤치 회로의 첫 번째 결정 패턴

Fig. 5. First test vector for the full scan version of s13207.

표 1. 그림 5를 기반으로 한 허프만 코드와 MSCIR 코드의 예

Table 1. The example of huffman and MSCIR codes for Fig. 5.

패턴	허프만 코드	패턴	MSCIR
0000	10	0000	0
0100	00	00	100
0010	110	01	101
0001	010	10	110
1000	0110	11	111

코드와 이를 비교하기 위한 허프만 코드를 표 1에 나타내었다.

3.2. 압축 알고리듬

제안하는 MSC 코드를 생성하기 위한 압축 알고리듬은 크게 3개의 과정으로 나눌 수 있다. 먼저 테스트 데이터에 존재하는 X값들을 효과적으로 압축하기 위한 특정한 값으로 채워주는 과정을 거치고 압축할 블록의 크기에 따라서 압축할 블록이 최대로 만들어 질 수 있도록 패턴의 순서를 재조합한다. 그리고 나서 이렇게 생성된 테스트 데이터를 제안하는 새로운 압축 코드로서 압축한다.

2장의 IR 기법을 이용하여 압축된 테스트 데이터를 T_{IR}이라 한다. 테스트 데이터 T_{IR}에는 여전히 'X'값들이 존재하기 때문에 이 값들을 효과적으로 압축하기 위한 압축 코드에 따라서 적절히 'X'값을 채워줄 필요가 있다. 제안하는 압축 알고리듬은 먼저 'X'값들을 압축 블록이 패턴 상에서 많이 존재할 수 있도록 적절히 '0' 또는 '1'값을 채워준다.

다음에 이렇게 'X'값을 채워준 테스트 데이터는 패턴 순서를 재조합함으로써 압축 블록이 좀더 자주 출현할 수 있도록 만들 수 있다. 각 패턴의 맨 처음 값과 맨 나중 값을 저장하고 그 길이도 먼저 계산해 둔다. 각 패턴의 맨 마지막 값과 다음 순서에 나타나는 패턴이 같은 값이 되도록 하고 그 길이가 압축을 위한 블록이 많이 생성될 수 있는 순서로 패턴 순서를 재조합한다. 그림 6에 한 블록을 4비트의 코드워드로 하였을 때의 패턴 순서 재조합의 예를 나타내었다. 여기서의 압축 블록은 '0000'으로 가정한다.

이렇게 재조합을 한 테스트 데이터를 3.1장에서 제안한 MSCIR 코드를 이용하여 가장 출현 빈도수가 높은 블록 하나를 선택하여 압축을 시도한다. 기본적으로 테스트 패턴에는 많은 'X'값을 가지고 있기 때문에 특정 블록 하나의 출현 빈도수를 높이는 것은 'X'값을 적절히 잘 채워 넣으면 그렇게 어렵지 않은 과정이다.

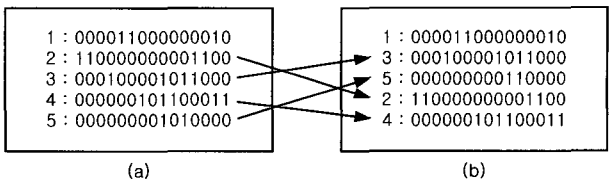

그림 6. 테스트 패턴 순서 재조합 과정

Fig. 6. Test pattern sequence reordering.

3.3. 디컴프레션 구조

위에서 제안한 압축 방법으로 압축된 테스트 데이터를 테스트 시에 이용하기 위해서는 ATE 자체에 압축을 풀어주는 하드웨어가 내장되어 있거나 SoC 내부에 압축을 풀어주는 하드웨어가 내장되어 있어야 한다. 일반적으로 ATE 자체에 압축을 풀어주는 하드웨어를 내장하기보다는 SoC 내부에 압축을 풀 수 있는 디컴프레션 구조를 넣어주는 것이 훨씬 수월하기 때문에 이 방법을 사용한다. SoC 내부에 내장되는 일반적인 디컴프레션 구조는 디코더와 이 디코더와 ATE 사이의 신호를 통제하는 컨트롤러로 이루어진다. 앞에서 언급했듯이 제안하는 압축방법은 디컴프레션 구조에 필요한 하드웨어 오버헤드를 줄이기 위해 기존의 연구[6,7,11,13]와는 달리 CSR 구조가 필요하지 않다. 또한 제안되는 디컴프레션 구조는 [16]에서 볼 수 있는 것처럼 ATE가 외부 클럭 동기화를 할 수 있다고 가정한다.

먼저 MSCIR 코드를 풀기 위한 디코더는 간단한 FSM으로 구현된다. 이 디코더는 테스터의 클럭과 테스터의 채널로부터 압축된 테스트 데이터가 전달되는 입력으로 구성된 2개의 입력을 가진다. 그리고 출력으로는 압축된 데이터를 풀었을 때 원래의 데이터를 전송하는 데이터 출력단과 3개의 컨트롤 신호를 내보내는 출력으로 이루어진다. 3개의 컨트롤 신호는 parallel_load, serial_load과 wait 신호로 구성된다. 이 신호들은 압축된 데이터가 디코드될 때 serializer로 신호를 내보내고 버퍼링 및 ATE와의 동기화를 위해서 필요한 신호이다. 디코더 FSM을 위한 상태 전이 다이어그램은 그림 7과 같이 간단히 나타낼 수 있다. 압축된 각각의 코드워드는 패턴이 압축된 것인지 아닌지 알려주는 비트를 가지고 있다. 제안한 압축 코드는 첫 번째 비트가 '1'이면 인코딩 되지 않은 패턴임을 나타내는 것이고 '0'이면 인코딩된 패턴임을 나타내는 것이다. 따라서 코드워드의 첫 번째 비트가 '1'이 들어오면 압축되지 않은 패턴을 나타내는 것이기 때문에 디코더는 단순히 serializer로 Ser

컨트롤 신호와 함께 다음에 들어오는 비트들을 2 클럭 사이클 동안 그대로 전달해 주기만 하면 된다. 또한 코드워드의 첫 번째 비트가 0이 들어오면 압축된 하나의 블록을 가르키는 것이기 때문에 그에 해당하는 블록의 비트에 해당하는 P0를 Par 컨트롤 신호와 함께 seria -lizer에 병렬적으로 전달한다.

FSM 디코더를 통해 복원된 테스트 데이터를 테스트하고자 하는 회로(CUT)의 스캔 체인에 넣어주고 ATE와 FSM의 신호를 통제하기 위한 컨트롤러는 그림 8과 같다. 제안된 압축 방법을 위한 컨트롤러는 칩 테스트 클럭에 맞도록 스캔체인에 복원된 테스트 데이터를 밀어주기 위한 serializer와 칩테스트 클럭과 FSM 클럭을 맞추어 주기 위한 부분으로 나누어져 있다. serializer에서 sync 신호가 1이 되면 FSM 클럭이 안정화 됨으로써 FSM 디코더의 디코딩 동작을 멈추고 serializer에서 스캔체인으로 테스트 데이터를 전송하고 sync 신호가 0이 되면 FSM 디코더에서 디코딩 동작을 함과 동시에 serializer에서 스캔체인으로 테스트 데이터를 전송한다.

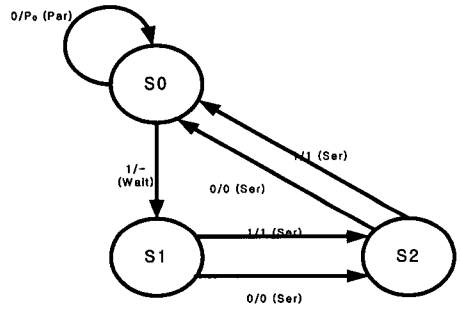

그림 7. MSCIR 코드를 위한 FSM 디코더의 상태 전이 다이어그램

Fig. 7. The state transition diagram of the FSM decoder for MSCIR code.

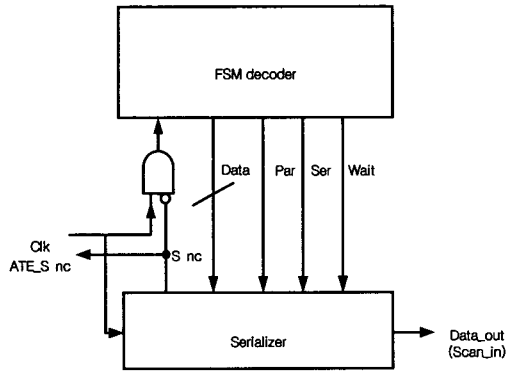

그림 8. 제안된 압축 방법의 FSM 디코더를 위한 컨트롤러

Fig. 8. The decompression controller for the proposed FSM decoder.

Ⅳ. 실험 결과

제안된 압축 방법의 성능 평가를 위해서 ISCAS '89 벤치마크 회로를 이용하여 실험을 수행하였다. 실험은 펜티엄 3 667MHz의 Linux 시스템에서 C로 구현하여 수행하였다. 각 회로를 위한 테스트 패턴은 기존의 제안된 방법들과의 비교를 위해서 MINTEST라는 ATPG 툴로 생성된 테스트 데이터를 이용하였다. 또한 기존의 논문들의 경우에는 블록의 크기에 따라 결과의 차이를 보였는데 본 논문의 실험은 [13]의 실험 결과 중 각 벤치회로에서 가장 좋은 성능을 보인 블록의 크기를 기준으로 하였다. 하지만 제안된 압축 방법의 블록의 크기는 4비트로 고정하여 사용하였다. 또한 정확한 실험을 위해서 SC[10], Golomb[6], FDR[7], VIHC[13]을 이용한 방법도 IR 기법을 적용한 후의 T_{IR} 테스트 패턴을 이용하여 압축하였다. 그 결과는 표 2와 같다.

표 2의 결과에서 볼 수 있듯이 제안된 압축 방법은 모든 회로에 대해서 기존의 압축 방법보다 훨씬 높은 압축율을 보이고 있다. 그 이유는 기존의 압축 방법과는 달리 MSCIR 코드가 IR 기법을 이용하여 입력되는 테스트 입력의 수를 줄임으로써 줄어든 테스트 데이터에 맞도록 만들어졌기 때문이다.

표 3에서는 기존의 압축 방법과 본 논문에서 제안된 MSCIR 코드를 위한 디컴프레션 구조의 하드웨어 오버

표 2. 기존의 압축 방법과 제안된 압축 방법의 압축율 비교 (%)

Table 2. The comparison of the proposed method and previous compression methods (%).

회로	블록 크기	IR+SC [10]	IR + Golomb [6]	IR+FDR [7]	IR + VIHC [13]	MSCIR
s5378	4	66.95	70.53	79.04	70.64	79.64
s9234	4	61.82	76.56	76.59	67.15	76.14
s13207	16	73.81	76.29	77.06	86.36	86.20
s15850	4	72.90	75.63	87.14	75.69	85.44
s38417	4	81.19	79.82	82.50	84.90	92.38
s38584	4	77.78	73.70	78.50	73.60	93.53

표 3. 각 압축 기법의 디컴프레션 하드웨어 오버헤드의 비교

Table 3. The comparison of decompression hardware overhead for each compression method.

블록 크기	SC [10]	Golomb [6]	FDR [7]	VIHC [13]	MSCIR
4	349	125	320	136	120
8	587	227		201	
16	900	307		296	

헤드를 Synopsys의 Design Compiler를 이용하여 계산된 크기로서 비교하였다. 객관적인 비교를 위하여 모든 하드웨어는 Synopsys의 기본 라이브러리인 lsi 10k 라이브러리를 이용하여 계산되었다. 단, 제안된 MSCIR 코드는 블록 크기를 4로 고정시켜서 사용하였다.

표 3의 결과에서 볼 수 있듯이 제안된 MSCIR 코드를 위한 디컴프레션 구조는 가장 작은 하드웨어 오버헤드를 가진다. 또한 기존의 Golomb[6], FDR[7], VIHC[13] 코드의 경우, 결과로 제시된 압축율을 얻기 위해서는 CSR 구조가 필요한데 CSR 구조에 필요한 하드웨어 오버헤드 역시 앞에서 언급했듯이 굉장히 크다. 예를 들어 ISCAS '89 벤치회로중 큰 회로의 하나인 s35932 벤치 회로의 경우 주입력과 스캔 입력을 포함하여 총 1763개의 입력이 필요하다. 이를 하나의 스캔 체인으로 구성하는 것을 가정하면 CSR 구조를 만들기 위해서는 총 1763개의 플립플롭과 하나의 XOR 게이트가 필요하게 된다. 물론 이를 여러개의 스캔 체인으로 나눈다면 그 하드웨어 오버헤드가 줄어들겠지만 그래도 여전히 하드웨어 오버헤드는 커지게 되고 회로가 커지고 스캔 입력을 포함한 회로의 입력이 커지면 커질수록 그 하드웨어 오버헤드는 커지게 된다. 또한 이를 해결하기 위해서 [11]에서 제안한 사용되지 않는 스캔 체인을 CSR 구조 대용으로 사용하는 방법을 이용한다면 대용으로 사용되는 스캔 체인의 길이는 압축을 풀고자 하는 스캔 체인 입력의 수와 같아야 한다는 제한이 있으며 이를 모든 회로에 적용하는 것은 아주 어려운 일이다. 또한 이렇게 CSR 구조를 기존에 존재하는 사용되지 않는 스캔 체인을 이용하기 위해서는 사용되지 않는 스캔 체인을 CSR 구조로 만들어 주고 이를 통제하기 위한 컨트롤 회로가 추가로 필요하기 때문에 하드웨어 오버헤드가 커지는 것을 막을 수는 없다. 게다가 SoC의 경우 그림 4와 같이 테스트를 하고자 하는 코어마다 디컴프레션 구조가 따로 붙기 때문에 SoC 내부에 코어가 많을수록 디컴프레션 구조의 하드웨어 오버헤드는 더욱더 커지게 된다. 따라서 본 논문에서 제안하는 압축 방법은 기존의 압축 방법에 비해 아주 적은 하드웨어 오버헤드와 높은 압축율을 가지는 가장 효율적인 압축 방법이라고 할 수 있다.

마지막으로 테스트 데이터 압축시 중요하게 생각되는 것은 테스트 적용시간의 단축이다. 이를 비교하기 위해서 표 4에 최대 테스트 적용 시간 단축을 얻기 위한 f_{sys}/f_T 의 최저값을 비교하였다. 여기서 f_{sys}/f_T의 최저값이 높으면 높을수록 테스트 적용시간을 줄이기 어렵고 이 값이 적을수록 테스트 적용시간을 줄이기 용이하다는 것을 나타낸다.

표 4. 각 압축 기법의 f_{sys}/f_T의 최저값
Table 4. The lower bound of f_{sys}/f_T.

회로	블록 크기	IR+SC [10]	IR + Golomb [6]	IR+FDR [7]	IR + VIHC [13]	MSCIR
s5378	4	6	4	128	4	4
s9234	4	6	4	64	4	4
s13207	16	6	16	512	16	4
s15850	4	6	4	512	4	4
s38417	4	6	4	1024	4	4
s38584	4	6	4	512	4	4

V. 결 론

본 논문에서는 IR 기법을 이용한 효율적인 테스트 데이터 압축 방법을 제안하였다. 이전의 연구[6, 7, 10, 13]와는 달리 테스트 데이터의 효율적인 압축을 위해서 CSR 구조를 이용하는 T_{diff} 테스트 데이터를 이용하지 않고 ATPG에서 생성된 테스트 데이터를 그대로 이용하면서 CSR 구조와 같이 디컴프레션 구조외의 추가적인 하드웨어 오버헤드를 가지지 않는 방법을 제안하였다. T_{diff} 테스트 데이터를 이용하지 않는 대신에 IR 기법을 이용하여 테스트 시에 필요한 테스트 입력을 줄이고 이렇게 함으로서 테스트 데이터의 아무런 손실 없이 테스트 데이터를 효과적으로 줄일 수 있다. 이렇게 줄어든 테스트 데이터 T_{IR}을 제안하는 압축 방법인 MSCIR 코드를 이용하여 압축하도록 한다. 제안된 MSCIR 코드는 디코드 과정이 간단하여 이를 디코드하기 위한 디컴프레션 구조 역시 간단하다. 따라서 기존의 제안된 압축 방법에 비해 적은 하드웨어 오버헤드를 가진다. 4장의 실험결과에서 볼 수 있듯이 제안하는 압축 방법은 적은 하드웨어 오버헤드를 가지면서 압축율은 기존의 연구보다 훨씬 높은 효율적인 압축 방법이라고 할 수 있다.

참 고 문 헌

[1] Y. Zorian, S. Dey, and M. J. Rodgers, "Test of Future System on Chips," In Proceedings International Conference on Computer Aided Design, pp. 392 - 400, 2000.

[2] *The International Technology Roadmap for Semiconductors*, 1999 Edition, ITRS

[3] I. Hamzaoglu and J. H. Patel, "Test set compac

-tion algorithms for combinational circuits," In Proceedings International Conference on Compu -ter Aided Design, pp. 283-289, 1998.

[4] I. Pomeranz, L. Reddy, and S. Reddy, *"Compactest: A method to generate compact test set for combinational circuits,"* IEEE Trans -actions on Computer Aided Design, Vol. 12, pp. 1040-1049, 1993.

[5] M. Ishida, D. S. Ha, and T. Yamaguchi, "Compact: A hybrid method for compressing test data," In Proceedings IEEE VLSI Test Sympo -sium, pp. 62 - 69, 1998.

[6] A. Chandra and K. Chakrabarty, "Frequency -Directed Run-Length (FDR) Codes with Application to System on a Chip Test Data Compression," In Proceedings IEEE VLSI Test Symposium, pp. 114 - 121, 2001.

[7] A. Chandra and K. Chakrabarty, *"System-on-a -Chip Test Data Compression and Decompre -ssion Architectures Based on Golomb Codes,"* IEEE Transactions on Computer Aided Design, Vol. 20, pp. 113 - 120, 2001.

[8] A. El-Maleh, S. al Zahir, and E. Khan, "A Geometric Primitives Based Compression Scheme for Tesing System-on-Chip," In Proceedings for IEEE VLSI Test Symposium, pp. 114 - 121, 2001.

[9] V. Iyengar, K. Chakrabarty and B. Murray, *"Deterministic Built In Pattern Generation for Sequential Circuits,"* Journal of Electronics Testing: Theory and Applications, Vol. 15, pp. 97 - 114, 1999.

[10] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, "Scan Vector Compression/Decompression Using Statistical Coding," In Proceedings IEEE VLSI Test Symposium, pp. 114 - 121, 1999.

[11] A. Jas and N. Touba, "Test Vector Decompre -ssion Via Cyclical Scan Chains and Its Applica -tion to Testing Core Based Designs," In Proceedings IEEE International Test Conference, pp. 458 - 464, 1998.

[12] A. Jas and N. Touba, "Using Embedded Proce -ssor for Efficient Deterministic Testing of System-on-Chip," In Proceedings International Conference on Computer Design, pp. 418 - 423, 1999.

[13] P. Y. Gonciari, B. M. Al-Hashimi, and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-Chip Test Data Compression /Decompression," In Proceedings Design, Automa -tion and Test in Europe Conference and Exhibition, pp. 604 - 611, 2002.

[14] C. A. Chen and S. K. Gupta, *"Efficient BIST TPG Design and Test Set Compaction via Input Reduction,"* IEEE Transactions on Com -puter Aided Design of Integrated Circuit and Systems, Vol. 17, pp. 692 - 705, 1998.

[15] H. K. Lee and D. S. Ha, *"On the Generation of Test Patterns for Combinational Circuits,"* Tech. report no.12_93, Department of Electrical Engi -neering, Virginia Tech

[16] D. Heidel, S. Dhong, P. Hofstee, M. Immediato, K. Nowka, J. Silaberman, and K. Stawiasz, "High-speed Serialiazing/Deserializing Design for Test Methods for Evaluating a 1 GHz Micropro -cessor," In Proceedings IEEE VLSI Test Symposium, pp. 234 - 238, 1998.

저 자 소 개

전 성 훈(정회원)
2002년 연세대학교 전기공학과 학사 졸업.
2004년 현재 연세대학교 전기전자공학과 석사 과정
<주관심분야: DFT, CAD>

임 정 빈(정회원)
2003년 연세대학교 기계전자공학부 전기전자전공 학사 졸업
2004년 현재 연세대학교 전기전자공학과 석사 과정
<주관심분야: DFT, Testing>

김 근 배(정회원)
2003년 연세대학교 전기공학과 학사 졸업.
2004년 연세대학교 전기전자공학과 석사 과정 졸업
2004년 현재 연세대학교 전기전자공학과 박사 과정
<주관심분야: On-line Test, DFT>

안 진 호(정회원)
1995년 연세대학교 전기공학과 학사 졸업.
1997년 연세대학교 전기전자공학과 석사 과정 졸업
2002년 LG전자 DTV 연구소 선임연구원
2004년 현재 연세대학교 전기전자공학과 박사 과정
<주관심분야: SoC 설계 및 응용, DFT>

강 성 호(정회원)
1986년 서울대학교 제어계측공학과 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
미국 Schlumberger 연구원, Motorola 선임 연구원
현재 연세대학교 전기전자공학과 교수
<주관심분야: SoC 설계 및 SoC 테스트>