SoC 테스트를 위한 테스트 데이터 압축 방법

▶ 서론

최근 VLSI 설계 기술과 공정 기술이 급격하게 발달함에 따라서 회로들의 집적도와 복잡도가 크게 증가하고 있다. 이에 따라 테스트를 위한 비용이 증가하고 있으며 이제는 테스트를 고려하여 설계가 이루어지는 Design-for test(DFT)가 설계 시점에서부터 고려되고 있다.

International Business Strategies (IBS)의 2007년 리포트에 따르면 디자인 비용 대비 테스트 비용이 22nm 공정으로 들어가면 36%의 비용을 차지할 만큼 크게 늘어날 것이다.[1] 칩을 테스트하기 위해서 기본적이고 일반적인 방법은 ATE를 이용하여 테스트 데이터를 칩에 입력해주고 그 응답을 받아서 고장을 찾아내는 것이다. 테스트 비용 증가의 요인 중 하나가 칩을 테스트하기 위한 테스트 데이터의 양이 급격하게 증가하는 것이다.

2006년 Test & Measurement World에서 Mentor Graphics에서 발표한 내용에 따르면 최근의 설계 기술의 발달과 이에 따른 테스트 품질의 필요조건을 만족하기 위한 테스트 어플리케이션 시간과 테스트의 양은 엄청나게 증가할 것으로 그림1처럼 예측된다. 이런 증가의 이유는 늘어나는 회로의 크기에 따른 표준 고착 테스트 패턴, 미세공정에 따른 타이밍 관련 결함을 위한 at-speed 테스트를 위해 필요한 테스트 패턴, n-detection 및 새로운 추가적인 DFM 기반 장애 모델을 테스트하기 위한 테스트 패턴 등에 기인한다.

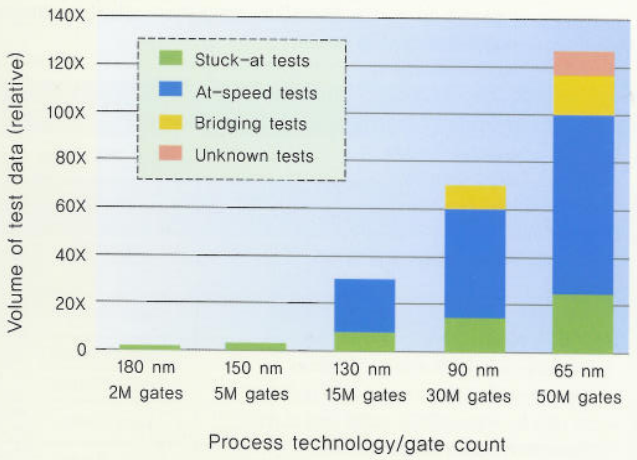


그림 1. 기술 발전에 따른 테스트 데이터의 양의 증가

이를 해결하기 위해 대두되는 방법 중의 하나가 테스트 데이터 압축 기법이 있다. 테스트 데이터 압축은 제한된 테스트 데이터 bandwidth를 통과하는 테스트 데이터의 크기를 줄여서 테스트 시간을 줄이는 방법이다. 압축기(compressor)와 해제기(decompressor)를 만들어서 압축기를 이용해 큰 용량의 데이터를 작은 크기로 압축하고 이를 다시 해제기를 통하여 원래의 데이터를 복원하는 방식이다.

ATPG를 통해 생성된 데이터(테스트 패턴)를 테스터에 저장하기 위해 여러 가지 압축 알고리즘을 적용하여 소프트웨어적으로 압축을 할 수 있다. 이렇게 압축된 데이터를 이용하여 테스트를 진행하기 위해서는 테스트 대상이 되는 칩에 인가하기 전에 원래 데이터로 복원해주는 회로가 포함되어야 한다. 이 해제기는 on-chip 하드웨어로 설계되어야 하며 칩 내부에 삽입되어야 하기 때문에 되도록 작은 크기로 간단하여야 하다. 따라서 초기 압축을 위한 알고리즘에서부터 이 해제기를 고려하여 쉽게 풀어낼 수 있는 알고리즘이 적용되어야 한다.

또한 해제기로 인한 비용 및 해제기의 고장이 발생할 여지도 생기므로 되도록 작고 신뢰성 있는 해제기를 설계하여야 한다. 해제기를 통하여 압축이 풀린 데이터가 테스트 대상 회로의 스캔입력으로 입력되고 테스트가 진행되고 응답 데이터가 다시 스캔 체인에 저장되게 된다. 새로운 테스트 데이터를 입력하는 동시에 응답 데이터가 스캔을 통해 나오게 되며 이 데이터를 다시 compactor와 같은 압축 알고리즘을 이용하여 작은 응답 데이터로 변환하여 ATE로 전송함으로써 테스트 시간을 줄이면서 고장여부를 판단할 수 있게 된다. 이를 도식적으로 나타내면 그림 2와 같다.

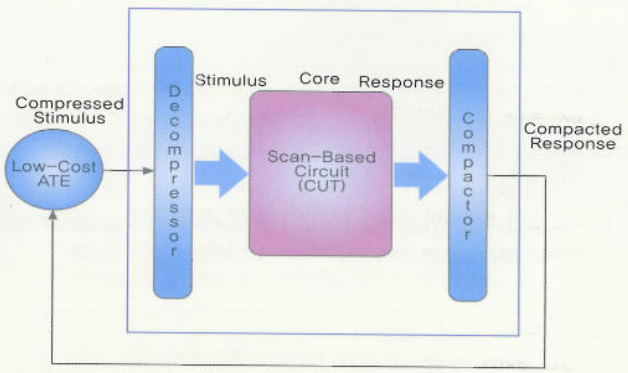
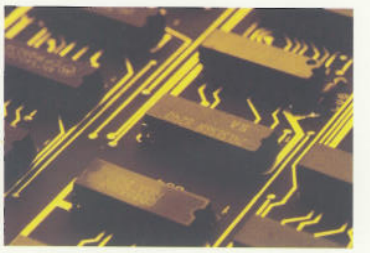


그림 2. 테스트 압축 구조



▶ 테스트 데이터 압축 방법

테스트 데이터 압축을 위한 여러 알고리즘들은 크게 3가지로 분류할 수 있다. 그 첫 번째가 테스트 데이터를 압축 코드로 이용하여 인코딩하여 데이터의 크기를 줄이는 코드 기반 압축 방법이 있으며, 테스트 데이터를 선형 알고리즘을 이용하여 압축해제를 하는 선형 해제기 방식의 압축방법도 있다.

마지막으로 여러 스캔 체인에 같은 값을 브로드캐스트 방식으로 전송하여 압축과 테스트 시간을 줄이는 브로드캐스트 스캔 기반의 압축 방식으로 나눌 수 있다.

■ 코드 기반 압축 방법

코드 기반 압축 방식은 테스트 데이터를 여러 알고리즘의 압축 코드를 이용하여 인코딩하여 그 크기를 줄이는 방식이다. 이러한 코드 기반 방식은 심벌이 고정되었냐 아니면 가변적이냐 두 가지의 경우와 코드워드가 고정인지 가변적인지에 따른 조합으로 크게 4가지로 나눌 수 있다.

고정-고정 방식은 테스트 데이터를 n 비트의 고정 길이의 심벌로 나눈 다음 이것을 b 비트의 고정 코드로 변환한다. 압축이 일어나기 위해서 b비트가 n비트보다 무조건 작아야 한다. b 비트의 인덱스를 가지는 dictionary 즉 사전 테이블을 만들어서 n 비트의 심벌을 이 테이블에 엔트리로 삽입한다.

n 비트의 심벌은 최대 2^n 개의 심벌을 가질 수 있으며 b 비트의 코드는 최대 2^6 개의 코드를 가질 수 있다. 사전 테이블에 저장되는 심벌의 집합이 원래 데이터의 전체 심벌의 집합을 포함하면 이 사전을 완전 사전이라고 한다.[2]

완전 사전을 이용하면 b비트의 외부 입력으로 n비트의 스캔 체인에 입력이 가능하게 된다. 이와 같이 압축 방법을 사용하여 테스트 데이터의 크기 뿐 아니라 전체적인 테스트 시간을 감속할 수 있다. 완전 사전을 이용한 압축 방식은 데이터의 양이 늘어남에 따라 사전의 데이터의 크기도 따라서 커지게 되고 이것으로 인한 하드웨어 오버헤드가 큰 문제가 되므로 부분 사전[2]을 사용하여 압축률은 약간 손해를 보고 하드웨어의 오버헤드를 줄이기도 한다.

고정-가변 방식은 원 테스트 데이터의 n비트의 블록으로 나누고 이 블록들을 다양한 길이를 가지는 코드로 인코딩한다. 가장 많이 나타나는 블록은 작은 길이의 코드로 변환하고 드물게 나타나는 블록은 긴 길이로 변환함으로써 전체적인 테스트 데이터의 크기를 줄일 수 있다.

Huffman code 방식[3]은 Huffman tree를 이용하여 빈도수에 따른 가장 효과적인 각 코드의 길이를 구함으로써 전체적인 패턴의 크기를 줄일 수 있다. 이 방식은 블록의 길이가 늘어나고 그에 따라 코드의 길이가 늘어나게 되면 해제를 하기 위한 디코더의 하드웨어 오버헤드가 크게 증가하는 문제가 발생한다.

이의 해결을 위해 Huffman tree로 변환을 하는 코드를 상위 빈도수가 높은 몇 개의 블록으로 제한하여 디코더의 크기를 줄일 수 있는 방법[4]이 제시되었다.

가변-고정 방식은 원 테스트 데이터를 다양한 길이의 심벌로 나누고 심벌들을 고정된 b 비트의 코드로 인코딩한다. 이 방식의 대표적인 알고리즘은 Run-length 코딩[5,6]이 있다.

0이나 1이 연속적으로 나타나는 코드를 하나의 심벌로 잡고 이를 정해진 길이의 코드로 인코딩함으로써 전체적인 데이터의 크기를 줄여서 압축을 하는 방식이다. 이러한 Run-length 방식의 경우 0이나 1이 연속적으로 나타나는 길이가 길어질수록 압축 효율은 더욱 좋아진다.

가변-가변 방식은 원 테스트 데이터를 다양한 길이의 심벌로 나누고 심벌들을 다양한 길이의 코드로 인코딩한다. run-length 코드로부터 발전된 형태인 Golomb 코드가 이 방식의 대표적인 압축 알고리즘이다[7]. Golomb 코드를 만들기 위해서는 그룹 크기를 정하고 0이나 1의 연속적인 반복의 길이를 나눠서 각각의 그룹에 포함시킨다.

각 Golomb 코드는 그룹 prefix와 tail로 두 부분으로 구성되며 그룹 prefix는 반복되는 길이가 속하는 그룹을 나타내는 값으로 0으로 끝나는 반복된 1로 나타내며 tail은 각 그룹 속에서 이 길이가 가지게 되는 인덱스를 의미한다. Golomb 코드에서 모든 그룹이 같은 양의 run-length를 가지게 되면 비효율적인 현상이 발생할 수 있기 때문에 이를 해결하기 위한 다양한 방법들로 FDR[8], 패킷 코드 방식[9]와 nine 코드[10] 등이 있다.

■ 선형 해체기 기반 압축 방법

또 다른 테스트 데이터 압축 방식으로는 선형 해제기를 이용하여 테스터로부터 입력되는 작은 크기의 데이터를 확장하여 스캔 체인에 채워주는 방식이다. 선형 해제기의 성능을 나타내는 지표는 인코딩 효율이며 이는 다음 식으로 나타내진다. 선형 해제기의 인코딩 효율이 최적의 성능을 나타내는 1에 어느 정도 가까운 비율을 보이는 것인지를 나타내며 일반적으로 이 값은 1을 넘을 수 없다.

인코딩 효율은 테스터에 저장되는 비트수 중에 구체화 된 비트수를 말한다. 가장 간단한 선형 해체기 기반 압축 방식은 XOR 조합 네트워크만을 사용하여 해제기를 구성하여 압축하는 방식이다.

각 스캔 체인이 테스터로부터 입력되는 몇 개의 채널들의 XOR 조합에 의해서 값이 채워진다. 낮은 인코딩 효율을 보완하기 위하여 [11]에서는 각 클럭에 채워지는 스캔 체인의 수를 동적으로 조절하였다.

■ 브로드캐스트 스캔 기반 압축 방법

브로드캐스트 스캔 방식[12]은 독립적인 여러 개의 회로를 테스트하기 위해 고안된 알고리즘이다. 일반적으로 사용하는 테스트 패턴은 임의 패턴과 결정 패턴으로 이루어져 있다. 여러 개의 독립적인 회로에 같은 임의 패턴을 입력하여 많은 수의 고장을 찾아내고 임의 패턴으로 찾아내기 힘든 고장에 대해 결정 패턴을 사용하게 된다.

이 경우에는 많은 수의 don't care 비트가 테스트 데이터에 존재하게 되는데 don't care 비트를 이용하여 같은 패턴을 가지고 여러 개의 회로에 남아 있는 고장들을 찾아낼 수 있는 패턴을 만들 수 있다. 만약 브로드캐스트 스캔 방식을 각각의 스캔 체인에 연결된 내부 회로들이 독립적이지 않은 하나의 회로에 적용하여 사용하게 된다면 모든 고장을 찾아낼 확률이 떨어지게 된다.

왜냐하면 두 개 이상의 스캔 체인이 하나의 채널을 통해서 연결되기 때문에 다른 스캔 체인의 같은 위치에 있는 스캔 셀은 모두 같은 값을 가지게 되므로 만약 어떤 고장을 검출하기 위해서 두 개의 스캔 체인에서 서로 다른 값을 가진 경우가 필요하다면 이와 같은 구조에서는 그것을 검출해내기 불가능하다.

이러한 문제를 해결하기 위해서 [13]과 [14]에서 일리노이 스캔 구조를 제안하였다. 이 구조는 브로드캐스트 모드와 직렬 스캔 모드라고 이름 붙인 두 개의 동작모드를 사용한다.

그림 3에서 볼 수 있듯이 브로드캐스트 모드는 앞의 브로드캐스트 스캔 구조와 동일하며 스캔 체인을 세그먼트라고 불리는 멀티 스캔 체인으로 나누고 여기에 하나의 스캔 입력을 사용하여 같은 테스트 데이터를 입력하여 많은 수의 고장을 찾아내게 된다.

그리고 브로드캐스트 모드에서 찾아내기 힘든 다른 고장들은 직렬 스캔 모드로 변환하여 원하는 테스트 검출율을 얻기 위한 테스트 패턴을 입력하여 고장을 검출한다. 일리노이 방식의 큰 단점은 직렬 스캔 모드의 경우에 전혀 압축이 되지 않은 테스트 데이터를 입력해줘야 하기 때문에 이 모드에서 얼마나 많은 테스트 데이터가 입력되는 가에 따라서 전체적인 압축률에 영향을 주게 된다.

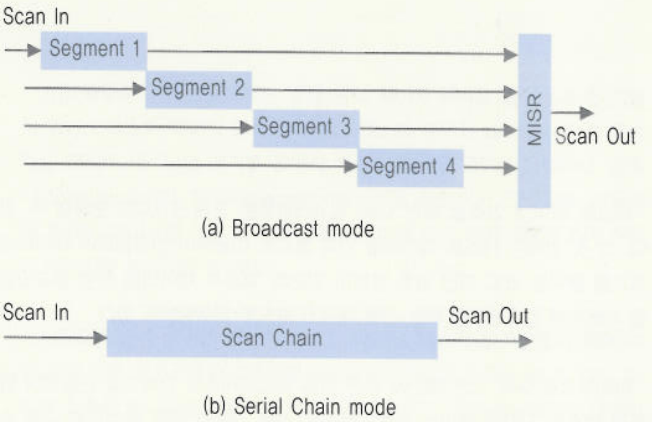


그림 3. 일리노이 스캔 구조의 두 가지 모드

[15]에서는 모든 스캔 체인에 하나의 채널을 이용하여 입력을 하는 대신에 여러 개의 입력 채널을 사용하는 multiple-input 브로드캐스트 스캔 구조를 제시하였다. 만약 두 개의 스캔 체인이 고장은 검출하기 위해 완벽히 독립적으로 컨트롤 할 수 있다면 이 두 개의 체인은 서로 다른 채널을 가질 수 있다.

많은 채널을 사용하고 각각의 짧은 스캔 체인을 사용함에 따라서 ATPG 과정에서 더 적은 제약 조건을 가지기 때문에 일리노이 스캔 구조에 비해서 더 많은 고장을 찾아 낼 수 있다.

multiple-input 브로드캐스트 scan의 경우 높은 고장 검출율을 얻기 위해서는 큰 숫자의 입력 채널이 필요한 문제가 있다. 이와 같이 채널의 숫자가 증가하는 문제를 해결하기 위해서 재구성 가능한 브로드캐스트 스캔 구조가 제안되었다. 이 방법은 각 채널 입력에 따라서 스캔 체인 세트의 구성을 변경하는 기능을 제공한다. 정적 재구성[16]은 새로운 테스트 패턴을 입력하는 경우에만 구성을 바꿀 수 있는 방법이다.

동적 재구성[17]은 패턴이 스캔 체인으로 입력되는 중간에도 동적으로 입력 채널 구성을 바꿀 수 있는 방법으로 좀 더 유연성을 가질 뿐 아니라 더 적은 채널을 가지고도 더 좋은 결과를 얻을 수 있다. 그러나 동적 재구성 방법은 제한된 경우에만 재구성이 가능한 정적 재구성 방법에 비해서 많은 양의 컨트롤 정보를 가지고 있어야 하는 단점이 있다.

▶ 결 론

칩의 크기가 커지게 됨에 따라 내부의 회로가 복잡해지며 고장이 발생 가능한 대상의 숫자가 늘어나게 되며 이를 테스트하기 위한 테스트 데이터 또

한 증가하게 된다. 단순히 회로의 크기가 커져서 테스트 데이터가 증가하는 것뿐 아니라 새로운 결함들을 찾아내기 위해서 추가적으로 테스트 패턴이 필요하게 된다. 이를 해결하기 위해서 다양한 테스트 데이터 압축 방법이 개발되고 있다.

테스트 데이터 압축의 효율성은 하드웨어의 크기를 최소화하면서 압축률을 높이는데 있다. ITRS의 2007년 리포트[18] 에 따르면 2013년까지 약 700배의 압축 효율을 보이는 데이터 압축 기술이 필요하다고 제시하고 있다. 이는 지금까지의 압축 알고리즘으로는 아직 도전적인 과제이며 지속적인 연구와 개발을 통해서 새로운 압축 방법들을 찾아나가야 할 것이다.

[참고자료]

[1] A. Chandra and K. Chakrabarty, "A unified approach to reduce SOC test data volume, scan power and testing time", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 22, Issue 3, pp. 352-363, 2003.

[2] L. Li, K. Chakrabarty, S. Kajihjara, and S. Swaminathan, Efficent space/time compression to reduce test data volume and testing time for IP cores, in Proc. VLSI Design, January 2005, pp. 53-58.

[3] D. A. Huffman, A method for the construction of minimum redundancy codes, in Proc. IRE, 40(9), 1098-1101, 1952

[4] A. Jas, J. Ghosh-Datidar, M.Ng, and N. A. Touba, An efficient test vector compression scheme using selective Huffman coding, IEEE Trans. Comput.-Aided Des., 22(6), 797-806, 2003.

[5] A. Jas and N. A. Touba, Test vector compression via cyclical scan chains and its application to testing core-based designs, in Proc. IEEE Int. Tet Conf., October 1998, pp. 458-464

[6] F. G. Wolff and C. Papachristou, Multiscan-based test compression and hardware decompression using LZ77, in Proc. IEEE Int. Test Conf., October 2002, pp. 331-339.

[7] A. Chandra and K. Chakrabarty, System -on-a-chip testdata compression and decompression architectures based on Golomb codes, IEEE Trans. Comput.-Aided Des., 20(3), 355-368, 2001

[8] A. Chandra and K. Chakrabarty, Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes, IEEE Trans. Comput. 52(8), 1076-1088, 2003.

[9] E. H. Volkerink, A. Khoche, and S. Mitra, Packet-based input test data compression techniques, in Proc. IEEE Int. Test Conf., October 2002, pp. 154-163

[10] M. Tehranipoor, M. Nourani, and K. Chakrabarty, Nine-coded compression technique for testing embedded cores in SoCs, IEEE Trans. VLSI, 13(6), 719-731, 2005

[11] C. V. Krishna and N. A. Touba, Adjustable width linear combinational scan vector decompression, in Proc. Int. Conf. on Computer-Aided Design, September 2003, pp. 863-866

[12] K.-J. Lee, J. J. Chen, and C. H. Huang, Using a single input to support muliple scan chains, in Proc. Int. Conf. on Computer-Aided Design, November 1998, pp. 74-78

[13] I. Hamzaoglu and J. H. Patel, Reducing test application time for full scan embedded cores, in Proc. Int. Symp. on Fault-Tolerant Computing, July 1999, pp. 260-267.

[14] F. F. Hsu, K. M. Butler, and J. H. Patel, A case study on the implementation of Illinois scan architecture, in Proc. IEEE Int. Test Conf., October 2001, pp. 538-547

[15] M. A. Shah and J. H. Patel, Enhancement of the Illinois scan architecture for use with multiple scan inputs, in Proc. Annual Symp. on VLSI, February 2004, pp. 167-172.

[16] A. R. Pandey and J. H. Patel, Reconfiguration technique for reducing test time and test volume in Illinois scan architecture based designs, in Proc. IEEE VLSI Test Symp. April 2002, pp. 9-15.

[17] L. Li and K. Chakrabarty, Test set embedding for deterministic BIST using a reconfigurable interconnection network, IEEE Trans. Comput.-Aided Des., 23(9), 1289-1305, 2004.

[18] International Technology Roadmap for Semiconductors, Test and Test Equipment, 2007

강성호 교수
연세대학교 전기전자공학과
E-mail:shkang@yonsei.ac.kr
http://soc.younsei.ac.kr