Logic Built In Self Test 구조의 내부 특성 패턴 매칭 알고리즘

전 유성 , **김 인수** . 민 형복
성균관대학교 정보통신공학부

Internal Pattern Matching Algorithm of Logic Built In Self Test Structure

Yusung Jeon , Insoo Kim, Hyoung Bok Min
School of Information&Communication Engineering, Sungkyunkwan University

Abstract - The Logic Built In Self Test (LBIST) technique is substantially applied in chip design in most many semiconductor company in despite of unavoidable overhead like an increase in dimension and time delay occurred as it used. Currently common LBIST software uses the MISR (Multiple Input Shift Register) However, it has many considerations like defining the X-value (Unknown Value), length and number of Scan Chain, Scan Chain and so on for analysis of result occurred in the process.

So, to solve these problems, common LBIST software provides the solution method automated.

Nevertheless, these problems haven't been solved automatically by Tri-state Bus in logic circuit yet.

This paper studies the algorithm that it also suggest algorithm that reduce additional circuits and time delay as matching of pattern about 2-type circuits which are CUT(circuit Under Test) and additional circuits so that the designer can detect the wrong location in CUT: Circuit Under Test.

1. 서 론

Logic Built In Self Test(LBIST)기법은 적용에 따라 발생하는 면적 증가 및 시간 지연 등과 같은 필연적인 오버헤드에도 불구하고, 실질적으로 대부분의 많은 반도체 회사에서 칩 설계시 광범위하게 적용하고 있다[1][2]. 현재 상용 Logic BIST 소프트웨어들은 MISR(Multiple Input Shift Register)를 사용한다. 그러나 이러한 과정에 발생되는 결과 분석을 위한 X값(Unknown Value)의 문제[3][4], Scan Chain의 길이 및 수, Scan Chain 등 많은 고려사항이 존재한다. 이와 같은 문제들을 해결하기 위해 상용 Logic BIST 소프트웨어들은 자동화된 해결 방식을 제공하고 있다.[5][6][7]

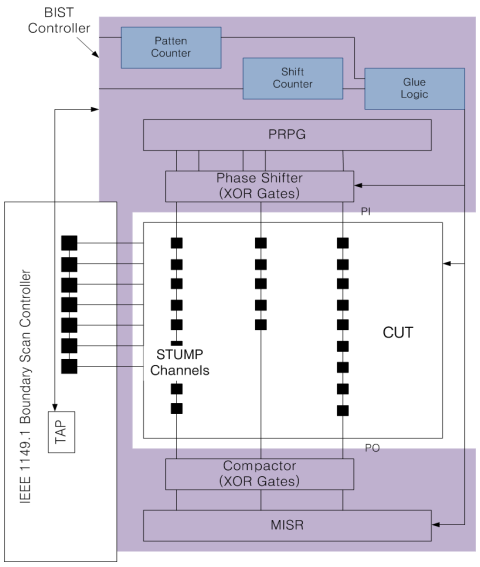
그러나 로직 회로 내에 삼상태 버스(Tri-state Bus)는 자동적으로 해결하지 못하고 있다. 본 논문에서는 설계자가 테스트 대상회로(CUT: Circuit Under Test)내에 잘못된 부분을 찾아내기 위하여 테스트 대상회로 및 설계자의 회로 두 가지 회로에 대한 Pattern Matching을 실행하여 부가회로 추가와 시간지연 현상을 줄일 수 있는 알고리즘을 제안한다.

2. 본 론

2.1 기존 Logic BIST에 문제점

그림 2.1는 Scan구조의 가장 대표적인 STUMP(Self-testing Using MISR and Parallel pseudo random pattern generator)구조를 가진 Logic BIST의 구조이다.

Logic BIST는 의사 무작위 테스트 패턴을 테스트 대상회로에 인가하기 위해 LFSR(Linear Feedback Shift Register)을 사용하며 테스트 대상 회로의 출력값 MISR에 의해 그 결과를 압축하고 테스트 대상회로의 고장 유무만을 판단한다. 대다수 고집적 회로들은 플립플롭에 의한 자장요소들을 가지고 있으며 LFSR에 의해 의사 무작위 테스트 패턴(Pseudo Random Test Patten)을 생성하여 테스트 대상회로에 인가하고 대상회로의 저장요소(Register Elements)는 주사(Scan)구조로 묶어 조합 논리회로와 같은 상태를 만들어 테스트를 수행하며, Logic BIST에 절차는 패턴 생성, 패턴 인가, 그리고 응답 압축 과정으로 나누어진다. 패턴 생성과 패턴 인가는 일정 수의 의사무작위 테스트 패턴을 패턴 생성기를 통하여 생산하고 회로로 인가하는 과정을 의미한다. 응답 압축은 응답 압축기를 통하여 회로의 출력 응답을 압축하여 회로의 고장 유무를 결정할 수 있는 압축치를 생산하는 과정이다. 응답 압축 과정에서 부딪히는 가장 큰 문제 중의 하나는 테스트 패턴에 의한 회로의 출력 응답 중 특정 출력으로 X값이 전파되어 나타나는 경우이다[3][4]. 응답 압축과정에서 X값을 반영한다면 올바른 압축치를 예상할 수 없게 된다.



<그림 1> Logic BIST 구조

2.2 Pattern Matching 알고리즘

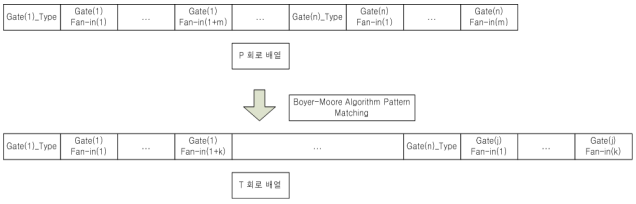
본 논문에서는 CUT내에 존재하는 X값의 존재여부를 CUT회로와 Good Circuit에 대하여 Pattern Matching하여 X값을 사전에 방지해주는 효과가 있으며, Pattern Matching의 알고리즘은 그림2와 같다.

```
Procedure PatternMatching(gate_list, gate_list1)
Begin
    Make array of gate_list
    Make array of gate_list1
end
Begin
    BM(T circuit information, P circuit information)
end
```

<그림 2> Pattern Matching 알고리즘

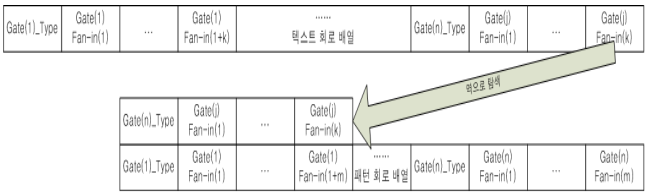
패턴 매칭을 수행할 때 신속한 탐색을 수행하기 위해서 BM알고리즘(Boyer-Moore Algorithm)을 사용한다. BM 알고리즘은 문자열에서 패턴을 찾아내는 데에 좋은 성능을 가지고 있다. 이 알고리즘을 적용하려면 우선 패턴 매칭에 사용하려는 회로에 대해서 데이터의 정리가 필요하다. 즉, 주어진 회로의 데이터 문자열화를 수행하여 BM알고리즘이 적용 가능하도록 별도의 데이터 정렬을 해야 한다. 이를 위해서 패턴 회로 데이터의 타입과

fan-in에 대한 데이터를 일정한 배열에 순차적으로 저장하고 이를 문자열로 간주하여 BM 알고리즘으로 하여금 검색 대상 회로에서 패턴을 발견하도록 한다. 이 때 알 수 있는 정보는 검색 대상 회로에서 존재하는 패턴의 인덱스 위치를 알 수 있다. 그리고 검색 대상 회로에 패턴이 1개 이상 존재할 경우 1개 이상의 인덱스를 보여 줌으로써 패턴의 존재 개수도 확인이 가능하다.



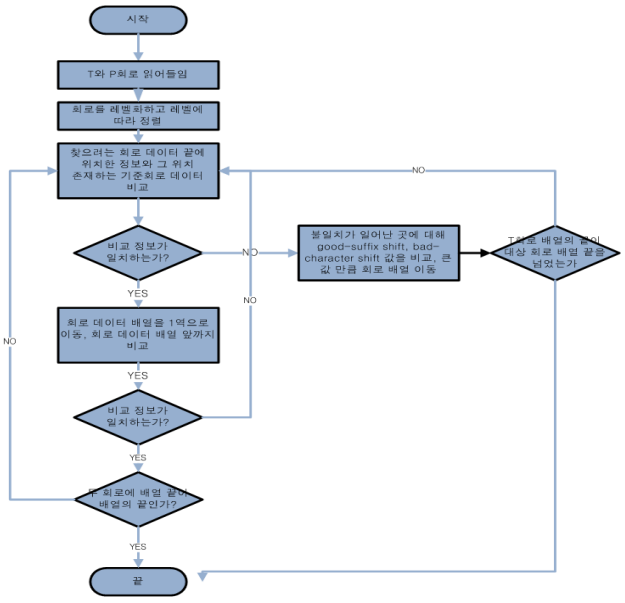
<그림 3> 회로의 데이터 배열화 작업

그림 3에서 보는 바와 같이 회로 정보를 배열에 순차적으로 정렬한다. 이는 BM알고리즘 수행을 위한 사전 데이터 정렬 작업으로 패턴 회로와 검색 대상 회로를 따로 따로 데이터 정렬화 작업을 수행한다.



<그림 4> BM알고리즘에 의한 패턴 검색

그림 5.4는 BM알고리즘에 의한 패턴 검색을 보여 주고 있다. 검색 대상회로를 역으로 탐색하여 패턴을 찾아낸다. 이때 검색하려는 패턴이 검출되면 이때 검색대상회로의 인덱스를 출력하여 패턴이 존재함을 알려준다.



<그림 5> Pattern Matching 순서도

BM알고리즘은 불필요한 비교를 최대한 줄이기 위하여 일정한 정보를 저장하게 되는데 이는 알고리즘의 성능을 최대한으로 이끌어 내기 위한 것이다. 지금까지 알려진 텍스트 탐색 알고리즘으로 가장 좋은 성능을 보여 준다.

표 1는 알려진 매칭 알고리즘의 수행시간을 비교한 것이다. 표에서 보는 바와 같이 BM알고리즘의 수행시간이 제일 뛰어난 것으로 나타나있다.

<표 1> 매칭 알고리즘 의 수행시간 비교

스트링 매칭	수행시간	
	평 균	최 악
Brute-Force	O(nm)	
Rabin-Karp	O(n+m)	O(nm)
KMP	O(n+m)	
Boyer-Moore	O((n-m+1)m+|Σ|)	

3. 실 험 결 과

고장 시뮬레이션, 테스트 용이화 설계, ATPG 및 BIST의 성능을 검증하기 위해 ISCAS85 벤치마크 회로 및 ISCAS89 회로를 사용하였다. 하지만, 설계 방식과 CAD 기술의 발전 및 변화에 따라, 현재 사용되고 있는 기술을 나타낼 수 있는 크고, 복잡한 벤치 마크 회로를 사용할 필요성이 존재한다. 이에 따라 ITC99 벤치마크 회로[8]를 사용하였다.

테스트 대상회로들은 Synopsys사의 DesignVision[9]로 합성하였고 Pattern Matching 시뮬레이터는 C언어로 구현되었었다.

<표 2> Pattern Matching 실험 결과

벤치 마크 회로	Tri-state bus 1x2	Tri-state bus 2x4	Tri-state bus 3x8	Tri-MUX 2x1	매칭률
B02	1	0	0	0	100%
B03	0	2	0	2	100%
B04	1	0	0	0	100%
B05	0	3	5	1	100%
B06	0	2	0	1	100%
B08	1	0	1	0	100%
B10	0	0	1	2	100%
B11	0	0	0	1	100%
B12	1	2	1	0	100%
B13	0	0	0	3	100%

4. 결 론

본 논문에서는 삼상태 버스 드라이버들과 두개의 삼상태 버퍼들의 동작허용 단자에 연결되어 멀티플렉서 기능을 하는 것에 대하여 상용 Logic BIST 소프트웨어들은 설계자의 몫으로 남겨놓고 있던 기존의 한계성을 극복하여 설계자가 원하는 회로가 존재하는지 또는 몇 개가 존재하는지 테스트 대상 회로와 설계자의 부가회로를 패턴 매칭 하여 두개의 회로를 비교한 후 잘못된 부분을 검출해줄 수 있는 알고리즘을 제안하였다.

[참 고 문 헌]

[1] Laung-Terng Wang, Cheng-Wen Wu, Xiaoqing Wen, "VLSI Test Principles and Architectures", Morgan Kaufmann Publishers(Elsevier), 2006.

[2] Laung-Terng Wang, Charles E. Stroud, Nur A.Touba, "System-On-Chip Test Architectures", Morgan Kaufmann Publishers(Elsevier), 2007.

[3] E. H. Volkerink and S. Mitra, "Response compaction with any number of unknowns using a new LFSR architecture," Proc. of Design Automation Conference, pp. 117-122, 2005.

[4] G. Hetherington, T. Fryars, N. Tamarapalli, M.Kassab, A. Hassan, and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," Proc. of International Test Conference, pp. 358-367, 1999.

[5] TurboBIST-Logic Manual, Syntest, 2006.

[6] B. Cheung and L.-T. Wang, "The seven deadly sins of scan-based design", Integrated Syst. 1996.

[7] A. A. Al-Yamani, S. Mitra, and E. j. McCluskey, "Avoiding Illegal States in Pesudorandom Testing of Digital Circuits, Technical Report (CRC TR) No. 02-2, Center for Reliable Computing, Stanford University, 2002.

[8] F. Corno, M. S. Reorda and G. squillero, "RT-Level ITC99 benchmarks and first ATPG result," IEEE Design & Test of Computers, pp. 44 - 53, 2000.

[9] DesignVision User Guide, Synopsys, 2007.