내장 자체 테스트의 low overhead를 위한 공간 압축기 설계

정 준 모†

요 약

본 논문에서는 VLSI 회로의 내장 자체 테스트(Built-In Self-Test)를 위한 효율적인 공간 응답 압축기의 설계 방식을 제안한다. 제안하는 공간 응답 압축기의 설계 방식은 테스트 대상 회로의 구조와는 독립적으로 적용할 수 있다.

기존의 공간 응답 압축기는 하드웨어 오버헤드(hardware overheads)가 크고, 고장 응답을 비고장 응답으로 변환시키는 에일리어싱(aliasing)에 의해 고장 검출률(fault coverage)을 감소시켰으나, 제안하는 방식에 의해 설계된 공간 응답 압축기는 기존의 방법에 비해 하드웨어 오버헤드가 작고, 고장 검출률을 감소시키지 않는다.

또한, 제안하는 방식은 일반적인 N-입력 논리 게이트로 확장이 가능하여 테스트 대상 회로의 출력 시퀀스에 따른 가장 효율적인 공간 응답 압축기를 설계할 수 있다.

제안한 설계 방식은 SUN SPARC Workstation 상에서 C 언어를 사용하여 구현하며, ISCAS'85 벤치마크 회로를 대상으로 선형 피드백 시프트 레지스터(Linear Feedback Shift Registers)에 의해 생성된 의사 랜덤(pseudo random) 패턴을 입력원으로 사용하여 시뮬레이션을 수행하므로써 그 타당성과 효율성을 입증한다.

A Design of Space Compactor for low overhead in Built-In Self - Test

Jun Mo Jung†

ABSTRACT

This thesis proposes a design algorithm of an efficient space response compactor for Built-In Self-Testing of VLSI circuits. The proposed design algorithm of space compactors can be applied independently from the structure of Circuit Under Test.

There are high hardware overhead cost in conventional space response compactors and the fault coverage is reduced by aliasing which maps faulty circuit's response to fault-free one. However, the proposed method designs space response compactors with reduced hardware overheads and does not reduce the fault coverage comparing to conventional method. Also, the proposed method can be extended to general N-input logic gate and design the most efficient space response compactors according to the characteristics of output sequence from CUT.

The proposed design algorithm is implemented by C language on a SUN SPARC Workstation, and some experiment results of the simulation applied to ISCAS'85 benchmark circuits with pseudo random patterns generated by LFSR(Linear Feedback Shift Register) show the efficiency and validity of the proposed design algorithm.

† 정 회 원 : 김포대학 전자과 교수

논문접수 : 1998년 1월 15일, 심사완료 : 1998년 7월 2일

1. 서 론

최근 반도체 기술의 발달로 한 칩에 집적되는 논리 게이트의 수가 급격히 증가함에 따라 반도체 제품의 기능(functionality)을 검증하는 테스팅 과정은 더욱 중요하고 많은 시간을 필요로 하는 문제로 대두되고 있다. 특히, 높은 신뢰도를 요구하는 제품의 경우에는 보다 효율적인 테스팅 과정이 요구되고 있다[1].

일반적으로 디지털 논리 회로의 테스트는 대상 회로에 일련의 입력을 인가하고, 외부 출력(Primary Output)을 통하여 관측된 결과와 예상되는 정상 결과를 비교하는 과정으로 구성된다. 그러나 회로의 크기가 커짐에 따라 인가할 테스트 패턴을 생성하는 데 걸리는 시간이 매우 증가하고, 예상되는 정상 결과를 저장하는 데 많은 메모리가 요구된다.

테스트 패턴의 생성과 생성된 패턴을 테스트 대상 회로에 인가하는 어려움을 해결하기 위하여 DFT(Design-For-Testability) 방식이 널리 이용되어 왔다. 그러나 DFT 방식도 VLSI 회로의 집적도와 복잡성이 증가함에 따라 테스트 패턴의 생성에 관련된 시간과 그 설계의 복잡성이 더 크게 증가한다. 이러한 문제를 해결하기 위한 방식이 내장 자체 테스트(Built-In Self-Test) 기법이다[1].

BIST는 크게 테스트 패턴 생성기(pattern generator)와 응답 압축기(response compactor)로 구분된다. 테스트 패턴 생성기로는 선형 피드백 시프트 레지스터(Linear Feedback Shift Registers)가 주로 사용된다. 응답 압축기에는 대상 회로의 출력수를 줄이기 위한 공간 응답 압축기(space response compactor)와 최종 시그너처(signature)를 생성하는 시간 응답 압축기(time response compactor)가 있다. 다중 입력 선형 피드백 시프트 레지스터(Multiple Input Linear Feedback Shift Registers)가 공간과 시간 응답 압축기로 사용되어 왔으나, 회로의 크기가 커짐에 따라 하드웨어 오버헤드가 매우 증가한다.

홀수 패러티(odd parity) 함수[2]를 이용하여 공간 응답 압축기를 구성하는 경우에도 여전히 테스트 패턴의 생성 시간이 많이 요구되며, 부가 제어 회로에 의한 테스트 출력의 지연이 문제가 된다. 출력 시퀀스에 따른 에러 전파 정도(Error Propagation Measure)를 계산하여 가장 큰 EPM을 갖는 논리 함수를 선택하는 방법[3]도 있으나, 대상 논리 함수에 대한 탐색 공간이 너무 크다는 문제가 있다.

본 논문에서는 VLSI 회로의 내장 자체 테스트를 위한 효율적인 공간 응답 압축기의 설계 방식을 제안한다. 제안하는 압축기의 설계 방식은 테스트 대상 회로의 구조와는 독립적으로 적용할 수 있다.

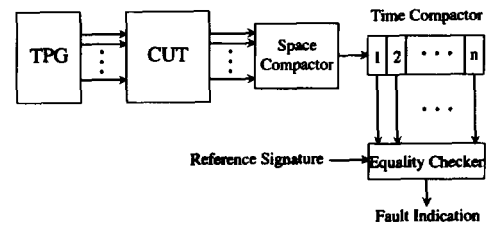
기존의 공간 응답 압축기는 하드웨어 오버헤드가 크거나, 에일리어싱에 의해 고장 검출률을 감소시켰으나, 제안하는 방식에 의해 설계된 공간 응답 압축기는 기존의 방법에 비해 하드웨어 오버헤드가 작고, 고장 검출률을 감소시키지 않는다.

또한, 제안하는 방식은 일반적인 경우로 확장이 가능하여 테스트 대상 회로의 출력 특성에 따른 가장 효율적인 공간 응답 압축기를 설계할 수 있다.

제안한 설계 방식은 SUN SPARC Workstation 상에서 C 언어를 사용하여 구현하며, ISCAS'85 벤치마크 회로를 대상으로 LFSRs에 의해 생성된 의사 랜덤(pseudo random) 패턴을 입력원으로 사용하여 시뮬레이션을 수행하므로써 그 타당성과 효율성을 입증한다.

2. BIST 기법

일반적인 BIST 구조는 (그림 1)에서와 같이 테스트 대상 회로에 인가할 테스트 패턴을 생성하는 테스트 패턴 생성기(TPG)와 대상 회로의 테스트 출력 응답을 압축하는 압축기(compactor)로 구성된다[1].



(그림 1) 일반적인 BIST 구조
(Fig. 1) General Strucure of BIST

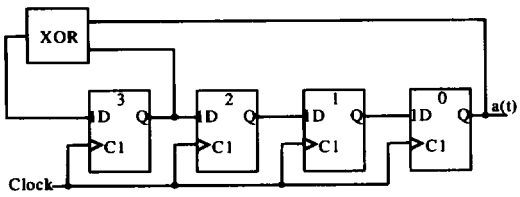
테스트 패턴 생성기로는 LFSRs이 주로 사용된다. 출력 응답 압축기에는 대상 회로의 출력 수를 줄이기 위한 공간 응답 압축기(space response compactor)와 최종 시그너처(signature)를 생성하는 시간(time) 응답 압축기가 있다.

2.1 테스트 패턴 생성

테스트 벡터는 테스트 패턴 생성 프로그램에 의해

만들어져 저장되거나(오프-라인 테스트 패턴 생성) 인가되는 중에 생성될 수 있다(동시 테스트 패턴 생성). 오프-라인으로 테스트 벡터를 생성하여 칩에 내장된 ROM에 저장하는 것은 이론적으로는 가능하지만 그다지 관심을 끄는 방법은 아니다. 왜냐하면, 이 방법은 테스트 패턴의 생성 비용을 줄이지 못 할 뿐만 아니라 매우 큰 ROM을 필요로 하기 때문이다.

테스트 패턴을 생성하는 방법은 LFSR을 이용한 랜덤 테스팅이 주로 이용된다. 이것은 자동 선형 피드백 시프트 레지스터(Linear Feedback Shift Registers)라 불리는 간단한 회로를 이용하여 생성된다. LFSRs은 외부 입력이 없고 모든 피드백이 exclusive-OR 게이트(XOR)로 연결된 지연 요소(D 플립플롭)로 구성된다. 4단 LFSRs과 일반적인 표준 형태의 LFSRs이 (그림 2)에 나타나 있다.



(a) $a(t+4) = a(t+3) \oplus a(t)$ $< h_4, h_3, \ldots, h_0 > = < 1, 1, 0, 0, 1 >$

생성 함수 : $f(x) = x^4 + x^3 + 1$

(그림 2) 자동 LFSRs의 표준 형태(4단 회로)
(Fig. 2) Standard Form of Automatic LFSRs(4 Stage)

(그림 2)의 LFSRs에 대한 상태들의 순서를 <표 1>에 나타내었다. 각 시퀀스가 15(24-1)클럭마다 반복되는 것을 알 수 있다. 이것이 4단 LFSRs이 생성할 수 있는 최대 주기(maximum period)이다. 즉, XOR를 사용하여 피드백 신호를 형성하기 때문에 모든 플립플롭이 0인 상태는 차기 상태 또한 0인 상태를 생성하게 된다. 일반적으로, N단 LFSRs이 생성할 수 있는 최대 주기는 2N-1이다. 모든 N의 값에 대해 최대 주기를 갖도록 LFSRs을 구현할 수 있다. 최대 주기의 LFSRs에 대응하는 생성 함수를 원시 다항식(primitive polynomial)이라고 부른다. 원시 다항식은 많은 논문과 책에서 찾아볼 수 있다. (그림 2)의 생성 함수는 원시 다항식이다.

<표 1> (그림 2)에 대한 상태 시퀀스
<Table 1> State Sequence for Fig. 2

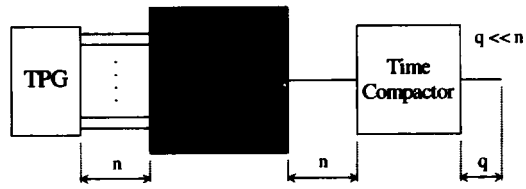
상태	Q1	Q2	Q3	Q4
0	1	0	0	0
1	1	1	0	0
2	1	1	1	0
3	1	1	1	1
4	0	1	1	1
5	1	0	1	1
6	0	1	0	1
7	1	0	1	0
8	1	1	0	1
9	0	1	1	0
10	0	0	1	1
11	1	0	0	1
12	0	1	0	0
13	0	0	1	0
14	0	0	0	1
15=0	1	0	0	0

2.2 출력 응답 분석

BIST의 테스트 응답 압축에는 두 가지 형태가 존재하는데, 하나는 잘 알려진 시간 압축이고 다른 하나는 공간 압축이다.

2.2.1 시간 압축

시간 압축은 긴 비트 시퀀스를 시그너처라 부르는 짧은 비트 시퀀스로 변환시키는 과정이다. 긴 비트 시퀀스의 길이는 105이나 106 비트에 이르지만 시그너처의 길이는 단지 16이나 32 비트에 불과하다. 단일 출력 회로에 대한 시간 압축의 과정이 (그림 3)에 나타나 있다.



(그림 3) 시간 압축 과정
(Fig. 3) Time Compaction Process

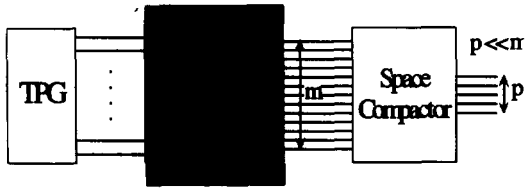
(그림 3)에서 TPG 블럭은 CUT에 인가할 n개의 테스트 벡터를 생성한다. 차례로 CUT는 n 비트의 출력

응답을 생성하고, 이 응답은 총 q 비트(일반적으로, q ≪ n)만을 출력하는 시간 압축기의 입력이 된다.

천이 계수(transition count), 일의 계수(ones count), 신드롬 계수(syndrome count), Walsh 스펙트럼 계수(Walsh spectrum coefficients) 등이 전형적인 시간 압축기로 사용된다. 일반적으로 많이 사용되는 방법은 시그너처 분석(Signature Analysis)으로 알려져 있는데, 대상 회로의 출력 응답 시퀀스를 선형 피드백 시프트 레지스터(LFSR이나 MISR)에 인가하고, 이것의 최종 상태(final state)를 시그너처로 사용한다.

2.2.2 공간 압축

공간 압축은 대상 회로의 출력이 2개 이상일 때 일반적으로 사용되는 다른 형태의 압축이다. 개념적으로, BIST 구조의 공간 압축은 다중 입력 비트 시퀀스를 더 적은 출력 비트 시퀀스로 감소시키는 변환과정으로 볼 수 있다. BIST 구조에서, 공간 압축된 비트 시퀀스는 대개 시간 압축기로 인가된다. 공간 압축 과정이 (그림 4)에 나타나 있다.



(그림 4) 공간 압축 과정
(Fig. 4) Space Compaction Process

대상 회로로부터의 m 비트 출력 벡터는 공간 압축기로 인가되고, 차례로 p 비트(일반적으로, p ≪ m)의 벡터로 압축된다. 다중 출력 회로로부터의 테스트 데이터를 단일 시그너처로 압축시킬 때 이러한 공간 압축기가 요구된다.

공간 압축은 시간 압축에 비해 많은 연구가 진행되지 않았다. 다중 입력 선형 피드백 시프트 레지스터(MISRs)와 같은 잘 알려진 구조가 시간과 공간 압축 모두를 수행할 수 있기 때문이다. 그러나 회로의 크기가 커지고 높은 테스트 특성이 요구됨에 따라 단일 MISRs을 사용하여 공간 압축을 수행하는 방법은 하드웨어 오버헤드 비용이 수용할 수 없을 정도로 커지고 에일리어싱 확률이 높아 고장 검출률이 많이 감소된다.

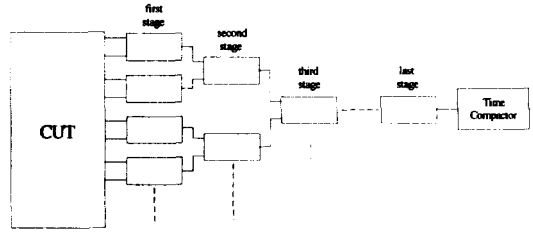
본 논문에서는 에일리어싱 확률이 높지 않고, 하드웨어 오버헤드가 적은 프로그래머블 공간 압축기의 설계 방식을 제안한다. 제안하는 공간 응답 압축기는 대상 회로의 다중 출력을 단일 출력으로 압축시킨다.

본 논문에서 제안하는 프로그래머블 공간 응답 압축기는 공간, 독립성, 비선형성, 조합 회로 등의 속성을 가지며, m : 1 의 공간 압축율을 가진다. 여기서 m은 대상 회로의 출력 수이다.

3. 2-입력 공간 응답 압축기

Li와 Robinson[4]은 회로의 구조와는 독립인 공간 응답 압축기를 제안하였다. 이 압축기를 Hybrid Space Compactor(HSC)라 하며, 2-입력 논리 게이트를 사용하여 구성한다.

공간 압축은 대개 exclusive-OR(XOR) 게이트를 사용하여 캐스케이드(cascade)나 트리 구조로 구성되어 왔다. XOR 연산의 특성 때문에 압축에서의 손실은 XOR 게이트의 구조에는 독립적이다. HSC에서는 AND와 OR 및 XOR 게이트를 기본으로 하여 공간 응답 압축기를 트리 구조로 구성한다. HSC의 일반적인 구성은 (그림 5)와 같다.



(그림 5) 일반적인 HSC
(Fig. 5) General HSC

2-입력 XOR 게이트는 어느 한 입력에서 에러가 발생한다면 이 에러를 반드시 게이트의 출력으로 전파시킨다. 그러나 두 입력에서 동시에 에러가 발생한다면, 이 에러는 전파되지 않는다. AND나 OR 게이트는 이러한 이중 에러(double error)중의 일부를 전파시킬 수 있고, 한 입력이 비제어 값(noncontrol value)을 갖는다면 다른 한 입력 상에서 발생하는 단일 에러(single error)도 전파시킬 수 있다. (그림 5)에 있는 각각의 2-입력 게이트에 사용될 논리 함수는 각 게이트에 대한 입력 시퀀스의 특성에 따라 결정된다.

Bayes의 정리[5]를 이용하여 단일 에러가 발생하는

사건과 이중 에러가 발생하는 사건에 대한 확률을 계산하면 그 합은 1이 된다. 단일 에러 확률을 S1, 이중 에러 확률을 S2라 표시한다. (그림 5)와 같이 2-입력 게이트로 구성되는 HSC의 경우는 각 게이트의 한 입력이나 또는 두 입력 모두에서 에러가 발생할 수 있으므로 단일 에러와 이중 에러만을 고려한다. 단일 에러의 경우는 두 입력중 어느 한 쪽에서만 에러가 발생하므로 경우의 수가 2이며, 이중 에러는 두 입력 모두에서 에러가 발생하므로 경우의 수는 1이다. 따라서 S1=2/3, S2=1/3이 된다. 이로부터 길이가 L인 게이트의 입력 시퀀스가 주어졌을 때의 2-입력 논리 함수에 대한 검출 가능한 에러 확률 E를 유도하면 다음과 같다.

$$E = S1\frac{R1}{2L} + S2\frac{R2}{L} \qquad (1)$$

여기서 R1은 그 논리 게이트에 대한 입력 시퀀스에서 검출 가능한 단일 비트 에러의 개수이고, R2는 그 논리 게이트에 대한 입력 시퀀스에서 검출 가능한 이중 비트 에러의 개수이다. S1과 S2는 각각 2/3, 1/3이다. 예를 들면, 2-입력 XOR 게이트에 대한 검출 가능한 에러 확률은 S1과 같다. 왜냐하면 2-입력 XOR 게이트는 모든 단일 에러는 검출할 수 있지만 모든 이중 에러는 검출할 수 없기 때문이다.

HSC로 공간 응답 압축기를 구성하려면 n개의 출력을 갖는 대상 회로에 대해 n-1개의 2-입력 게이트가 요구되고, $\lceil \log_2 n \rceil$ ($\lceil\ \rceil$: 올림함수) 단(stage)이 필요하다. 따라서 많은 2-입력 게이트가 사용되고, 압축기의 단 수가 매우 커지며 또한, 대상 회로의 출력 시퀀스가 여러 단의 게이트를 통과해야 하므로 에러 마스킹(error masking) 확률이 커지는 단점이 있다.

4. 프로그래머블 공간 응답 압축기

BIST(Built-In Self-Test) 구조에서 가장 문제가 되는 것은 하드웨어 오버헤드와 고장 검출률의 감소 즉, 에일리어싱(aliasing) 현상이다. 본 논문에서는 고장 검출률을 감소시키지 않으면서, 하드웨어 오버헤드를 줄일 수 있는 효율적인 프로그래머블 공간 응답 압축기의 설계 알고리듬을 제안한다. 제안하는 방식은 대상 회로의 구조와는 독립적으로 적용할 수 있고, 일반적인 경우로 확장이 가능하다.

4.1 검출 가능한 에러 확률

본 논문에서 제안하는 공간 응답 압축기는 3-입력과 4-입력 논리 게이트를 사용하여 구성된다. 그러나 제안한 설계 방식을 적용하면, 일반적인 N-입력 논리 게이트를 사용하여 공간 응답 압축기를 구성할 수 있다.

기본적인 3-입력과 4-입력 논리 게이트로는 AND, OR, XOR 게이트를 사용한다. XOR 게이트는 입력에서 발생한 단일 에러(single error)와 삼중 에러(triple error)를 모두 전파시키지만 이중 에러(double error)는 전파시키지 못한다. 그러나 AND와 OR 게이트는 이러한 이중 에러중의 일부를 전파시킬 수 있으며, 다른 입력이 비제어 값(noncontrol value)을 가질 경우 단일 에러를, 3개의 입력 값이 같을 경우 삼중 에러를 전파시킬 수 있다. 따라서 대상 회로의 출력 시퀀스의 특성에 따라 AND와 OR 게이트의 에러 전파 확률이 XOR 게이트보다 크게 된다. 결국 에러 전파 확률이 더 큰 게이트를 선택함으로써 에일리어싱 확률을 줄일 수 있다.

Bayes의 정리[5]를 이용하여 단일 에러가 발생하는 사건과 이중 에러가 발생하는 사건 및 삼중 에러가 발생하는 사건에 대해 확률을 계산하면 그 합은 항상 1이 된다. 단일 에러가 발생할 확률을 S1, 이중 에러가 발생할 확률을 S2, 삼중 에러가 발생할 확률을 S3라 하면 3가지 에러에 대해서만 고려하므로 모든 경우의 수는 8이 되며, 이 중에서 에러가 없는 경우를 제외하면 에러가 발생할 모든 경우의 수는 7이 된다. 단일 에러가 발생하는 경우는 3가지(각 입력에서의 에러)이며, 이중 에러가 발생하는 경우도 3가지(각 두 입력에서의 에러)이다. 마지막으로 삼중 에러가 발생하는 경우는 1가지(세 입력 모두에서의 에러)이다. 이로부터 S1=3/7, S2=3/7, S3=1/7이 됨을 알 수 있다. 이러한 에러 확률로부터 테스트 벡터의 길이가 L인 대상 회로의 출력 시퀀스가 주어졌을 때의 3-입력 논리 게이트에 대한 검출 가능한 에러 확률 E를 유도하면 다음과 같다.

$$E = S1\frac{R1}{3L} + S2\frac{R2}{3L} + S3\frac{R3}{L} \qquad (2)$$

여기서 R1은 그 논리 게이트에 대한 입력 시퀀스에서 검출 가능한 단일 비트 에러의 개수이고, R2는 그 논리 게이트에 대한 입력 시퀀스에서 검출 가능한 이중 비트 에러의 개수이며, R3는 그 논리 게이트에 대

한 입력 시퀀스에서 검출 가능한 삼중 비트 에러의 개수이다. (2)식을 이용하여 각 AND, OR, XOR 게이트에 대한 E를 계산하고, E의 값이 가장 큰 게이트를 선택한다.

4-입력 게이트에 대해서도 마찬가지로 Bayes의 정리를 적용하면, 4중 에러 확률 S4만을 추가적으로 고려하면 된다. 4-입력 논리 게이트에 대한 검출 가능한 에러 확률 E는 다음과 같다.

$$E = S1\frac{R1}{4L} + S2\frac{R2}{6L} + S3\frac{R3}{4L} + S4\frac{R4}{L} \qquad (3)$$

여기서 R1, R2, R3는 (2)식에서와 같으며, R4는 그 논리 게이트에 대한 입력 시퀀스에서 검출 가능한 4중 비트 에러의 개수이다. 에러 확률 S1, S2, S3 및 S4는 각각 4/15, 6/15, 4/15 및 1/15이다.

본 논문에서 제안한 공간 응답 압축기에 사용되는 3-입력 게이트의 개수는 다음 식에 의해 구할 수 있다.

$$N_3 = \left\lceil \sum_{k=1}^{m} \left(\frac{n}{3}\right)\left(\frac{1}{3}\right)^{k-1} \right\rceil = \left\lceil \frac{1}{2}(n-1) \right\rceil \qquad (4)$$

여기서 N3는 필요한 3-입력 게이트의 개수이고, n은 대상 회로의 출력 수이며, m은 구성되는 공간 응답 압축기의 단 수로써 $\log_3 n$ 이 된다. 따라서 본 논문에서 제안하는 방식은 기존의 방법[4]에 비해 약 30% 정도의 하드웨어 오버헤드를 줄일 수 있다.

4.1.1 에어리어싱 발생확률 계산

일반 게이트 자체의 에러전파확률을 먼저 구하고 단(Stage)을 고려하면 된다.

먼저 AND,OR등과 같은 일반 Primitive Gate와 XOR gate로 나누어서 먼저 게이트 자체의 고장전파확률을 고려하면 다음과 같다.

(1) 일반 Primitive Gate(AND,OR 등)

m 입력게이트가 있을 때, 해당 게이트의 입력에 인가하는 모든 경우의 수의 조합은 2m이 되고 이 모든 입력에 있어서 가능한 고장의 경우의 수를 표현하면 아래와 같다.

2m(mC1 + mC2 +mC3 + - - +mCm) : mC1 은 단일 고장이 발생하는 경우의 수

또한 m입력게이트 입력에 에러가 들어왔을 때 출력으로 에러가 전파되는 모든 경우의 수를 구하면 아래와 같다.

2mC1 + 2mC2 +2mC3 + - - +2mCm : 2mC1은 단일 고장이 발생할 때 이값을 전파시키는 경우

AND의 예를 들면 2입력의 경우 , 출력이 정상출력 0 --> 1로 바뀐 경우, 또는 정상출력 1 -->0으로 바뀐 경우 두가지 모두 전파하며, 각각의 경우는 입력중 하나만 1이 되면 되므로 2mC1 이 된다.

따라서 전파확률은 아래 수식으로 표현된다.

$$\frac{2_mC_1 + 2_mC_2 + 2_mC_3 + - + 2_mC_m}{2^m({}_mC_1 + {}_mC_2 + {}_mC_3 + - + {}_mC_m)}$$

여기서 간략화 하면 21-m 이 됨.

(2) XOR Gate

m 입력 XOR의 입력에 에러가 들어왔을 때 출력으로 에러가 전파되는 모든 경우의 수는 Primitive와는 조금 다르다. 왜냐하면 m 입력에서 에러가 2개가 발생되면 XOR의 특성상 정상출력으로 되돌아 오므로 짝수개의 에러인 경우에는 에러가 전파되지 않는다.

따라서 이경우의 전파확률은 아래 수식으로 표현된다.

$$\frac{2^m({}_mC_1 + {}_mC_3 + - + {}_mC_m)}{2^m({}_mC_1 + {}_mC_2 + {}_mC_3 + - + {}_mC_m)}$$

이 수식으로 부터 아래와 같은 게이트들의 에러 전파확률을 구할수 있다.

2입력 AND,OR 등

Prob(전파) = 21-2 = 1/2

2입력 XOR

Prob(전파) = 2C1/(2C1 + 2C2) = 3/4

3입력 AND,OR 등

Prob(전파) = 21-3 = 1/4

3입력 XOR

Prob(전파) = 4/7

지금까지 구한 것은 단순히 한 개의 게이트만 있다고 본경우이고 단(Stage)를 고려한 경우는 아래와 같이 표현할수 있다.

Stage 수는 아래와 같이 계산할수 있다.

* 2 입력으로만 구성된 경우

  단수 = log2n = 3.32log10n

  (여기서 n은 Space Compactor 입력 수)

* 3 입력으로만 구성된 경우

  단수 = log3n = 2.1log10n

위의 예로부터 Primitive Gate로만 구성된 경우 에러 전파확률은 다음과 같다.

* 2입력 Gate로 구성된 Compactor인 경우

$$\text{Prob(전파)} = \left(\frac{1}{2}\right)^{3.32\log_{10}n}$$

* 3 입력 Gate로 구성된 Compactor의 경우

$$\text{Prob(전파)} = \left(\frac{1}{2}\right)^{4.2\log_{10}n}$$

이 식으로 보면 3 입력 Gate로 구성된 Compactor가 오히려 전파확률이 적다는 것을 알 수 있다.

다음은 XOR Gate로만 구성된 경우 에러전파 확률은 다음과 같다.

* 2입력 XOR로 구성된 Compactor

$$\text{Prob(전파)} = \left(\frac{2}{3}\right)^{3.32\log_{10}n}$$

* 3입력 XOR로 구성된 Compactor

$$\text{Prob(전파)} = \left(\frac{4}{7}\right)^{2.1\log_{10}n}$$

이 수식은 지수의 Base가 다르기 때문에 계산이 하며 <표2>에 나타내었다.

〈표 2〉 에러전파확률

〈Table 2〉 Error Propagation Probability

	n = 40	n = 60	n = 90
2입력	0.088	0.059	0.059
3입력	0.107	0.107	0.061
4입력	0.152	0.152	0.081

<표2>에서 알수 있듯이 XOR로만 구성된 Compactor는 m이 클수록 에러 전파확률도 지금까지 AND, OR와 같은 Gate와 XOR Gate 각각의 경우에 대한 에러전파확률에 대해서 고려해 봤는데 AND, OR같은 Gate로만 구성된 경우는 에러전파확률이 입력개수가 크면 클수록 전파확률이 낮아지고, XOR의 경우는 반대로 전파확률이 커진다.

그러나 실제 구현된 경우를 보면 주로 XOR Gate로 많이 구성되는데 일반 AND,OR보다 전파확률이 크기 때문이다.

따라서 보편적으로 보면 입력의 개수가 크면 클수록 에러 전파확률이 커진다고 말할수 있다.

4.2 설계 알고리듬

일반적인 N-입력 논리 게이트를 사용하여 공간 응답 압축기를 설계하는 과정은 다음과 같다. 여기서 각 변수의 정의는 다음과 같음.

Wn(M) : n단에서 입력시퀀스에 있는 1의 개수가 많은 순서로 정렬했을 때 그순위가 M번째인 입력 스퀀스 예를 들면 W2(1) 은 1번째 단에서 입력스퀀스에 있는 1의 개수가 1위인 스퀀스를 말함

S1,S2,,,,SN : 단일에러 발생확률, 이중에러 발생확률 ,,,,, N중에러 발생확률

m : 단수

p : 각단의 입력 스퀀스 개수

N : 게이트의 입력갯수

설계 알고리듬

Set the values of S1, S2, S3, ···, and SN for the circuit

Do n=1, m

Do M=1, p, N ; within same stage

a) Find out Wn(M), Wn(M+1), Wn(M+2), ···, and Wn(M+N-1).

b) Compare the detectable error probability for Wn(M), Wn(M+1), Wn(M+2), ···, and Wn (M+N-1) with AND, OR, and XOR gates.

c) Take the logic function for Wn(M), Wn(M+1), Wn(M+2), ···, and Wn(M+N-1) with the highest value of detectable error probability to provide another sequence for next stage.

End do for M
End do for n

본 논문은 이러한 설계 알고리듬을 이용하여 에러 전파 확률이 가장 큰 공간 응답 압축기를 구현한다. 검출 가능한 에러 확률이 같은 경우에는 XOR 게이트를 선택한다.

4.3 N-입력으로의 확장

본 논문에서 제안하는 설계 알고리듬은 일반적인 N-입력 게이트까지로 확장이 가능하다. 먼저 i중 에러에 대한 확률을 구하면 다음 식과 같다.

$$S_i = \frac{{}_NC_i}{2^N - 1}, \quad i = 1, 2, 3, \cdots, N \tag{5}$$

여기서 ${}_NC_i = \binom{N}{i}$이다. 이로부터 테스트 벡터의 길이가 L인 대상 회로의 출력 시퀀스가 주어졌을 때의 N-입력 논리 게이트에 대한 검출 가능한 에러 확률 E의 일반식을 유도하면 다음과 같다.

$$E = S1\frac{R1}{{}_NC_1 L} + S2\frac{R2}{{}_NC_2 L} + \cdots + S_N\frac{R_N}{{}_NC_N L}$$

$$= \sum_{i=1}^{N} Si\frac{Ri}{{}_NC_i L} \tag{6}$$

여기서 Ri은 그 논리 게이트에 대한 입력 시퀀스에서 검출 가능한 i중 비트 에러의 개수이다.

N-입력 게이트를 사용하여 공간 응답 압축기를 구성하였을 때 필요한 게이트의 개수는 다음 식으로부터 구할 수 있다.

$$N_T = \left\lceil \sum_{k=1}^{m} \left(\frac{n}{N}\right)\left(\frac{1}{N}\right)^{k-1} \right\rceil$$

$$= \left\lceil \frac{1}{N-1}(n-1) \right\rceil \tag{7}$$

여기서 NT는 필요한 N-입력 게이트의 개수이고, n은 대상 회로의 출력 수이며, m은 구성되는 공간 응답 압축기의 단 수로써 $\log_N m$ 이 된다.

N 입력으로 확장하는 경우에는 지연시간 , 신호구동능력 등을 고려하여야 한다.

N-입력 확장시 이러한 문제점이 발생한다면 최적의 N을 구해야 한다.

5. 실험 및 결과

본 논문에서 제안하는 프로그램머블 공간 응답 압축기는 SUN SPARC Workstation 상에서 C 언어를 사용하여 구현하였으며, 성능 평가를 위하여 ISCAS'85 벤치마크 회로를 대상으로 stuck-at 고장에 대해 LFSRs에 의해 생성된 각각 480개와 960개의 패턴을 입력원으로 사용하여 시뮬레이션을 수행하였다.

<표 3>과 <표 4>는 기존의 방법[4]과 본 논문에서 제안한 설계 알고리듬을 적용하여 구현한 공간 응답 압축기의 하드웨어 오버헤드(hardware overheads)를 패턴에 따라 비교한 결과이다.

〈표 3〉 하드웨어 오버헤드 비교(480개의 패턴 사용)
〈Table 3〉 Comparison for H/W overhead(for 480 patterns)

대상 회로	기존의 방법[4]			제안한 방법					
				3-입력 게이트 압축회로			4-입력 게이트 압축회로		
	AND	OR	XOR	AND	OR	XOR	AND	OR	XOR
c432	1	0	5	1	0	2	0	0	2
c499	0	0	31	0	0	16	0	0	11
c880	6	6	13	2	3	8	1	1	7
c1355	0	0	31	0	0	16	0	0	11
c1908	0	0	24	0	0	12	0	0	8
c3540	1	2	18	0	1	10	0	0	7
c5315	11	25	86	4	13	44	2	7	32
c6288	0	1	30	0	0	16	0	0	11
c7552	16	2	88	6	0	47	4	0	32

〈표 4〉 하드웨어 오버헤드 비교(960개의 패턴 사용)
〈Table 4〉 Comparison for H/W overhead(for 960 patterns)

대상 회로	기존의 방법[4]			제안한 방법					
				3-입력 게이트 압축회로			4-입력 게이트 압축회로		
	AND	OR	XOR	AND	OR	XOR	AND	OR	XOR
c432	1	0	5	1	0	2	0	0	2
c499	0	0	31	0	0	16	0	0	11
c880	5	6	14	2	3	8	1	1	7
c1355	0	0	31	0	0	16	0	0	11
c1908	0	0	24	0	0	12	0	0	8
c3540	1	2	18	0	1	10	0	0	7
c5315	14	25	83	3	12	46	2	7	32
c6288	0	1	30	0	0	16	0	0	11
c7552	15	2	89	6	1	46	4	0	32

<표 3>과 <표 4>의 결과로부터 인가된 테스트 패턴의 길이에 따라 공간 응답 압축기가 다르게 구성되는 것을 알 수 있다. 즉, 대상 회로의 출력 시퀀스의 특성에 따라 가장 효율적인 공간 응답 압축기가 구성된다. 또한 본 논문에서 제안한 방법이 기존의 방법[4]에 비해 하드웨어 오버헤드가 약 30% 정도 감소함을 알 수 있다.

<표 5>는 LFSRs에 의해 생성된 각각의 패턴을 대상 회로에 인가했을 때의 고장 검출률(fault coverage)을 나타낸다. 이 고장 검출률은 공간 압축을 행하기 전의 값이다. 960개의 테스트 패턴을 인가했을 때가 각 대상 회로의 고장을 더 많이 검출할 수 있음을 알 수 있다.

고장 응답(faulty response)이 비고장 응답(fault-free response)으로 변환되는 현상인 에일리어싱(aliasing)이 발생할 확률에 대해 기존의 방법[4]과 비교한 결과를 <표 6>에 나타내었다. 벤치마크 회로에 대해 에일리어싱 확률이 증가하거나 감소된 경우가 각각 반 정도임을 표로부터 알 수 있다. 또한, 모든 벤치마크 회로에 대해 에일리어싱 확률의 증가비율이 감소비율보다 적음을 알 수 있다. c499 회로의 경우에는 한 고장의 효과가 동시에 영향을 미치는 주출력(PO)의 수가 많고, 이러한 주출력들을 공간 압축하는 과정에서 에러 마스킹(error masking) 현상의 발생으로 에일리어싱 확률이 다른 회로에 비해 약간 높음을 알 수 있다. 이와 같은 회로에 대해서는 고장의 효과가 동시에 영향을 미치는 주출력 신호선들을 분리하여 공간 압축을 수행함으로써 에일리어싱 확률을 줄일 수 있다.

에일리어싱 확률이 감소된 이유는 대상 회로에 대해 구성된 공간 응답 압축기의 단(stage) 수가 기존의 방법에 비해 줄어듦으로써 에러 마스킹 확률이 감소되기 때문이다.

〈표 5〉 입력으로 사용한 패턴에 대한 고장 검출률
〈Table 5〉 Fault Coverage for input patterns

대상 회로	고장수	기존의 방법[4] : 960패턴 고장검출률(%)	480 패턴		960 패턴	
			검출된 고장 수	고장 검출률(%)	검출된 고장 수	고 장 검출률(%)
c432	392	96.9	387	98.724	387	98.724
c499	486	98.0	480	98.765	484	99.588
c889	886	96.5	865	97.630	873	98.533
c1355	1174	96.3	1141	97.189	1158	98.637
c1908	1826	94.6	1665	91.183	1758	96.276
c3540	3438	95.0	3210	93.368	3280	95.404
c5315	4970	97.2	4936	99.316	4964	99.879
c6288	4896	98.9	4879	99.653	4879	99.653
c7552	7436	93.1	7010	94.271	7039	94.661

〈표 6〉 에일리어싱 확률의 비교(단위 %)
〈Table 6〉 Comparison of Aliasing Probability

대상 회로	기존의 방법[4]		제안한 방법			
			3-입력 게이트 압축회로		4-입력 게이트 압축회로	
	480 패턴	960 패턴	480 패턴	960 패턴	480 패턴	960 패턴
c432	0.775	0.775	0.775	0.775	**0.258**	**0.258**
c499	5.000	4.959	5.000	4.959	5.000	4.959
c880	1.387	1.489	**1.040**	1.604	**1.040**	**0.687**
c1355	2.103	2.073	2.103	2.073	2.103	2.073
c1908	1.441	0.853	1.441	0.853	1.441	0.853
c3540	2.087	1.829	2.368	2.104	2.150	**1.799**
c5315	2.289	1.873	**2.269**	**1.773**	**1.033**	**0.745**
c6288	0	0	0.041	0.041	0.041	0.041
c7552	3.909	1.762	**2.140**	2.131	**2.397**	1.861

〈표 7〉 패턴 수에 따른 c880 회로에 대한 공간 응답 압축회로 구성
〈Table 7〉 Space Response Compactor for c880 circuit

패턴 수	기존의 방법[4]			제안한 방법					
				3-입력 게이트 압축회로			4-입력 게이트 압축회로		
	AND	OR	XOR	AND	OR	XOR	AND	OR	XOR
160	5	8	12	2	3	8	1	2	6
320	5	7	13	2	3	8	1	2	6
480	6	6	13	2	3	8	1	2	6
640	6	6	13	2	3	8	1	2	6
800	5	6	14	2	3	8	1	2	6
960	5	6	14	2	3	8	1	1	7
1120	5	6	14	2	3	8	1	2	6
1280	5	6	14	2	3	8	1	2	6
1440	5	6	14	2	3	8	1	2	6
1600	5	6	14	2	3	8	1	2	6
1760	5	6	14	2	3	8	1	1	7
1920	5	6	14	2	3	8	1	1	7

〈표 8〉 패턴 수에 따른 c880 회로에 대한 고장 검출률과 에일리어싱 확률의 비교
〈Table 8〉 Comparision of fault coverage and aliasing probability for c880 circiut

패턴 수	고장 수	검출된 고장 수	고장 검출률(%)	에일리어싱 확률(%)		
				기존의 방법[4]	제안한 방법	
					3-입력 게이트 압축회로	4-입력 게이트 압축회로
160	886	850	95.937	2.118	**0.941**	2.235
320	886	857	96.727	1.050	**0.933**	1.050
480	886	865	97.630	1.387	**1.040**	**1.040**
640	886	865	97.630	0.694	0.925	**0.578**
800	886	865	97.630	0.578	0.925	0.578
960	886	873	98.533	1.489	1.604	**0.687**
1120	886	878	99.097	1.822	**1.708**	1.936
1280	886	878	99.097	1.822	**1.708**	1.936
1440	886	878	99.097	1.822	**1.708**	1.936
1600	886	878	99.097	1.822	**1.708**	1.936
1760	886	878	99.097	1.822	**1.708**	**0.797**
1920	886	878	99.097	1.822	**1.708**	**0.797**

<표 7>과 <표 8>은 패턴 수에 따른 c880 회로에 대한 공간 응답 압축회로의 구성과 고장 검출률 및 에일리어싱 확률을 보여준다. 인가된 패턴에 따른 대상 회로의 출력 시퀀스의 특성에 따라 가장 효율적인 공간 응답 압축기가 구성되고, 기존의 방법[4]에 비해 에일리어싱 확률이 많이 감소됨을 알 수 있다.

6. 결 론

본 논문에서는 내장 자체 테스트(Built-In Self-Test)를 위한 효율적인 공간 응답 압축기의 설계 방식을 제안하였다. 제안한 압축기의 설계 방식은 테스트 대상 회로의 구조와는 독립적으로 적용할 수 있다. 기존의 공간 응답 압축기는 하드웨어 오버헤드가 크고, 에일리어싱(aliasing)에 의해 고장 검출률을 감소시켰으나, 제안한 방식에 의한 공간 응답 압축기는 기존의 방법에 비해 하드웨어 오버헤드가 적고, 테스트 대상 회로의 출력 특성에 따라 가장 효율적으로 공간 압축기를 설계하므로써 많은 테스트 대상 회로에 대해 고장 검출률을 감소시키지 않았다.

제안한 방식은 일반적인 N-입력 논리 게이트로 확장이 가능하며, N-입력 논리 게이트를 사용할 경우 하드웨어 오버헤드를 기존의 방법에 비해 많이 줄일 수 있다. 3-입력 논리 게이트를 사용하여 공간 응답 압축기를 구성할 경우 하드웨어 오버헤드를 약 30% 정도 줄일 수 있었으며, 고장 검출률을 감소시키지 않았다.

제안한 설계 방식은 C 언어를 사용하여 구현하였으며, ISCAS'85 벤치마크 회로를 대상으로 LFSRs에 의해 생성된 의사 랜덤 패턴을 입력으로 사용하여 시뮬레이션을 수행하여 그 타당성과 효율성을 입증하였고, 일부 벤치마크 회로에 대해 고장 검출률의 향상을 보였다.

향후 과제로는 1의 개수만을 고려하지 않고 대응하는 비트 위치에 대해 동일한 값이 많은 출력 핀들을 고려함으로써, 좀 더 에일리어싱 확률을 줄일 수 있는 공간 응답 압축기를 설계하는 것이다. 또한 회로의 구조 분석을 통해서 N중 에러 확률 SN을 결정하고, 고장의 효과가 서로 독립적으로 영향을 미치는 출력 핀들을 공간 압축 대상으로 고려함으로써 좀 더 효율적인 공간 응답 압축기를 구현하는 것이다.

참 고 문 헌

[1] E. J. McClusky, "Built-In Self-Test Techniques," IEEE Design and Test of Computers, pp.21-28, April 1985.

[2] K. Chakrabatry and J. P. Hayes, "Efficient Test Response Compression for Multiple-Output Circuits," Proc. 1994 Int. Test Conf., pp.501-510, 1994.

[3] Y. Zorian and A. Ivanov, "Programmable Space Compaction for BIST," Proc. Int. Symp. on Fault-Tolerant Computing, pp.340-349, 1993.

[4] Y. K. Li and J. P. Robinson, "Space Compression Methods with Output Data Modification," IEEE Trans. on Computer-Aided-Design, Vol. C-6, pp.290-294, March 1987.

[5] R. U. Hogg and E. A. Tanis, "Probability and Statistical Inference," 2nd ed., New York : Macmillan, pp.46-49, 1983.

정 준 모

1987년 한양대학교 전자공학과(학사)
1989년 한양대학교 대학원 전자공학과(석사)
1989년~1996년 7월 삼성전자 기술총괄 ASIC 센타
1996년 8월~현재 김포대학 전자과 교수
관심분야 : CAD & TEST, Low Power Design, ASIC 설계