논문 2007-44SD-2-16

# 위상천이 네트워크를 사용한 X-마스크 기법

## ( An X-masking Scheme for Logic Built-In Self-Test Using a Phase-Shifting Network )

송 동 섭*, 강 성 호**

( Dong-Sup Song and Sungho Kang )

요 약

본 논문에서는 최대길이 의사무작위 이진 시퀀스(m-시퀀스)의 쉬프트-덧셈 특성에 근거한 위상천이를 이용하여 회로 출력에 나타나는 X-값을 효과적으로 마스크 함으로써 내장된 자체 테스트를 실현할 수 있는 기법을 제안한다. 이 기법은 패턴생성기인 LFSR의 출력을 적절하게 위상천이 하여 마스크 패턴을 생성할 수 있는 위상천이 네트워크를 이용한다. 테스트 절차 동안에 각 스캔 체인에 인가되는 마스크 패턴의 위상 천이 수는 재구성 가능하다. LFSR의 출력을 적절하게 위상 천이하여 모든 스캔 체인 마스크 패턴을 생성할 수 있는 위상천이 네트워크 합성 알고리즘을 제안한다. 본 논문에서 제안하는 X-마스크 회로는 각 스캔 체인 마스크 패턴을 생성할 수 있는 후보 위상천이 수가 많기 때문에 하드웨어 오버헤드를 효과적으로 감축할 수 있다. 실험을 통하여 제안된 위상천이를 이용한 X-마스크 회로는 기존의 연구 결과보다 훨씬 적은 저장공간과 하드웨어 오버헤드를 필요로 함을 증명한다.

Abstract

In this paper, we propose a new X-masking scheme for utilizing logic built-in self-test The new scheme exploits the phase-shifting network which is based on the shift-and-add property of maximum length pseudorandom binary sequences(m-sequences). The phase-shifting network generates mask-patterns to multiple scan chains by appropriately shifting the m-sequence of an LFSR. The number of shifts required to generate each scan chain mask pattern can be dynamically reconfigured during a test session. An iterative simulation procedure to synthesize the phase-shifting network is proposed. Because the number of candidates for phase-shifting that can generate a scan chain mask pattern are very large, the proposed X-masking scheme reduce the hardware overhead efficiently. Experimental results demonstrate that the proposed X-masking technique requires less storage and hardware overhead with the conventional methods.

**Keywords :** Logic BIST, response compaction, X-masking, phase-shifting

## I. 서 론

내장된 자체 테스트 기법은 저속, 저가의 외부 ATE(Automatic Test Equipment)를 이용하여 고집적, 고속의 칩 테스트를 가능하게 하는 효과적인 테스트 용이화(Design for Testability) 설계 기법으로 알려져 있다. 내장된 자체 테스트 절차는 패턴 생성, 패턴 인가, 그리고 응답 압축 과정으로 나눠진다. 패턴 생성과 패턴 인가는 자동 테스트 패턴 생성(Automatic Test Pattern Generation)을 통해 얻은 결정론적 테스트 패턴 혹은 일정 수의 의사무작위 테스트 패턴을 패턴 생성기를 이용하여 생산하고 회로로 인가하는 과정을 의미한다. 응답 압축은 응답 압축기를 통해 회로의 출력 응답을 압축하여 회로의 고장 유무를 결정할 수 있는 압축치(signature)를 생산하는 과정이다.

응답 압축 과정에서 부딪히는 가장 큰 문제 중의 하나는 테스트 패턴에 의한 회로의 출력 응답 중 특정 출력으로 X-값이 전파되어 나타나는 경우이다[1]. 로직

* 학생회원, ** 평생회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Graduate School, Yonsei University)
접수일자: 2006년11월13일, 수정완료일: 2007년2월6일

회로내 X-값 발생원으로는 삼상(tri-stated) 혹은 부유(floating) 버스 라인들, 초기화 되지 않은 플립플롭이나 래치들, 다중 클록 도메인을 갖는 회로에서 두 개의 클록 도메인을 걸치는 신호선들, 그리고 로직 회로에 내장된 아날로그 혹은 메모리 블록으로부터 전파되는 X-값들을 예로 들 수 있다[2]. 응답 압축 과정이 올바르게 동작하기 위해서는 무고장 회로의 압축치를 시뮬레이션에 의해서 예상할 수 있어야 한다. 만일 X-값이 응답 압축기로 전파된다면 무고장 회로에 대한 압축치는 올바르게 예상될 수 없다. 특히, 내장된 자체 테스트 기법에 가장 많이 사용되는 응답 압축기인 MISR(Multiple Input Signature Register)로 유입되는 단 하나의 X-값은 모든 비트의 값이 X-값인 신호치를 생산하기도 한다.

응답 압축기로 X-값이 전파되는 문제는 크게 두 가지 연구 방향으로 나뉘어져 해결책을 모색해 왔다. 첫째는 X-내성(X-tolerant) 콤팩터를 설계하는 방법이고 둘째는 X-마스크(X-mask) 회로를 설계하는 방법이다.

X-내성 콤팩터란 동시에 나타나는 제한된 수의 X-값이 존재하여도 회로 응답 중 임의의 고장 검출을 보장하는 응답 압축 기법이다. 선형 블록 코드(linear block code) 패리티 체크 매트릭스를 이용하여 에일리어싱 확률을 낮출 수 있는 스페이스 콤팩터 설계 방법이 [3]에서 제안되었다. 이 방법은 회로 응답에 X-값이 존재하지 않는 상황을 가정하였다. [4]에서는 패리티 체크 매트릭스를 기반으로 하여 X-값과 임의의 수의 고장영향이 포함한 테스트 응답도 콤팩션 할 수 있는 X-COMPACT가 제안되었다. 하지만 X-COMPACT는 내성 할 수 있는 X-값의 수가 제한된 단점이 있다. X-COMPACT는 XOR 게이트로 이루어진 조합회로로, 내성 할 수 있는 동시에 나타나는 X-값의 수가 콤팩터 디자인과 콤팩터 출력의 수를 결정한다. 물론 내성할 X의 수는 콤팩터 설계 이전에 결정되어야 하는 설계 파라미터이다. [5]에서는 X-COMPACT와 마찬가지로 제한된 수의 X-값과 임의의 수의 고장영향이 존재하는 회로 응답을 콤팩트 할 수 있는 convolutional compactor가 제안되었다. Convolutional compactor는 비-피드백(non-feedback) 쉬프트 레지스터와 회로의 스캔 체인과 쉬프트 레지스터를 연결하는 XOR 게이트 네트워크로 구성된다. XOR 네트워크의 효율적 배치로 동시에 2개 혹은 홀수개 존재하는 고장을 검출할 수 있으며, 100배 이상의 응답 압축률을 갖는다. 하지만, 콤팩터 설계시 상호 소멸(mutual cancellation) 현상 때문에 테스트 응답에 나타나는 어떤 고장들은 검출될 수 없게 되는 에러 마스크 현상이 단점으로 작용한다. X-내성 콤팩터는 스페이스 콤팩션 기법에 적합한 방법이다. 대부분의 X-내성 콤팩터는 설계 파라미터로 내성할 수 있는 X-값의 수가 제한되어 있다. 그러므로 설정된 수보다 많은 X-값이 회로 응답에 존재할 경우는 MISR로 대표되는 내장된 자체 테스트의 응답 압축기와 같이 사용할 때 큰 문제를 유발 할 수 있다.

회로 응답이 응답 압축기로 입력되기 전에 오직 규정값(known value: 0 또는 1)만이 응답 압축기로 전파될 수 있도록 부가적인 로직을 삽입하여 회로 출력에서 발생하는 X-값을 마스크 하는 기법을 통칭 X-마스크 기법이라 한다. [6][7]에서는 X-값 출력을 다루기 위해서 외부의 ATE에 의해서 조정되는 부가적인 마스크 회로를 제안하였다. 이 방법은 마스크 회로를 제어하여 X-값을 포함한 응답 벡터 전체를 마스크 하는 것이 가능하다. 하지만 고장검출율의 손실을 초래할 수 있다. [8]에서는 외부의 ATE로부터 로드된 시드(seed)에 따라서 X-값의 마스크에 필요한 패턴을 생산하는 LFSR을 제안하였다. 이 방법에서 고장검출율을 손상시키지 않는 X-값은 마스크 하지 않는다. 하지만, MISR을 응답압축기로 사용할 경우 유효한 압축기를 얻기 위해서는 모든 X-값이 마스크 되어야 하기 때문에 단점이 존재한다. 최근에는 결정론적 테스트 패턴 생성 기법으로 연구되었던 reseeding에 기반을 둔 LFSR[1][9], LFSR의 각 단에 서로 다른 가중치를 할당하는 가중치 선형 콤팩터(weighted linear compactor)[10], 비트-반전(bit-flipping) 기법을 활용한 XML[11] 등이 X-마스크를 위한 설계 기법으로 발전되었다. 이 방법들은 마스크 회로 구현에 소요되는 하드웨어 오버헤드가 중요한 제한요소로 작용한다. 마스크 회로를 LFSR reseeding으로 구현할 때에는 다수의 seed를 저장하기 위한 부가적인 하드웨어가 부담으로 작용할 수 있다. 그리고 비트-반전을 이용한 XML의 마스크 회로는 X-출력을 갖는 출력 비트 수 만큼의 OFF 세트, 최대 고장영향 비트 수 만큼의 ON 세트로 구현된 부가적인 조합회로로 구현되기 때문에 마스크 회로의 하드웨어 오버헤드를 적정 수준으로 낮추는 노력이 필요하다.

본 논문은 응답압축기로의 X-값 전파를 막기 위한 방법으로 새로운 X-마스크 기법을 제안한다. 본 논문에서 제안하는 X-마스크 회로는 위상천이 네트워크를 이

용한다. 위상천이기(phase-shifter)는 의사무작위 테스트 패턴이 스캔 체인으로 인가될 때 LFSR의 연속된 단(stage)에 의해서 생성되는 비트 시퀀스의 선형의존성을 줄이기 위해 사용되는 XOR 게이트 트리이다. 위상천이기는 최대길이 의사무작위 이진 시퀀스인 $m$-시퀀스의 윈도우 성질(window property)과 쉬프트-덧셈 성질(shift-and-add property)을 갖기 때문에, 주어진 임의의 스캔 체인 마스크 패턴이 쉬프트 이동된 $m$-시퀀스의 처음 $l$ 비트 동안에 생산되도록 XOR 네트워크를 구성할 수 있다($l$은 스캔 체인의 길이를 의미). 회로 응답 중 X-값 비트와 고장영향 비트를 제외한 비트를 U-값 비트라 하자. 여기서 고장영향 비트란 무고장 회로의 출력값과 고장 회로의 출력값이 서로 상이하여 고장의 유무를 판단하는데 사용할 수 있는 회로 출력 비트를 의미한다. 만일 $l$ 비트의 스캔 체인 마스크 패턴에 $d$ 개의 U-값 비트가 존재한다면 주어진 스캔 체인 마스크 패턴을 생성하기 위해 최대 $2^d$ 가지로 XOR 네트워크를 구현 가능하다. 많은 연구에서 하나의 고장은 다수의 테스트 응답을 통해 다수의 출력비트에서 고장영향이 나타난다는 것이 증명되었다. 이 점은 고장 시뮬레이션을 통해 결정되는 모든 고장영향을 응답압축기로 전파할 필요가 없음을 뜻하기 때문에 고장검출율 및 결함검출율이 영향 받지 않는 범위내에서 임의의 스캔 체인 마스크 패턴내에 U-값 비트의 수는 극대화 될 수 있다. 최대 개수의 스캔 체인 마스크 패턴이 생산 될 수 있는 XOR 네트워크 구성을 찾는다면 위상천이를 이용한 X-마스크 회로의 하드웨어는 최소화 될 수 있다.

본 논문은 다음의 성질을 만족하는 X-마스크 회로 설계를 목적으로 한다.

(1) MISR을 응답압축기로 사용할 수 있도록 회로의 모든 X-값 출력을 마스크 한다.
(2) 하드웨어 오버헤드를 최소화 할 수 있도록 모든 스캔 체인 마스크 패턴을 최소 개수의 위상천이를 이용하여 생성한다.

## Ⅱ. X-마스크 패턴과 위상천이를 이용한 마스크 회로

칩이 실제로 제조되기 이전에 X-값 발생원을 모두 규정 할 수 없기 때문에 테스트 포인트와 같은 테스트 용이화 설계를 통하여 모든 X-값 발생원을 제거 한다는 것은 실제로 불가능하다. 본 장에서는 구조적인 X-마스크 회로 설계가 가능하도록 X-마스크 패턴을 분석하고 본 논문에서 제안하는 위상천이에 기반을 둔 마스크 회로의 기본 아이디어를 기술한다.

### 1. X-마스크 패턴

테스트 대상 회로의 출력에서 생산되는 X-값들을 신호압축기로 들어가기 이전에 걸러내는 장치는 여러 가지 방법으로 실현이 가능하지만, 구현이 용이하고 적은 하드웨어 오버헤드만을 필요로 하는 장점 때문에 그림 1과 같이 AND(혹은 OR) 게이트를 이용하는 방법이 널리 사용된다.

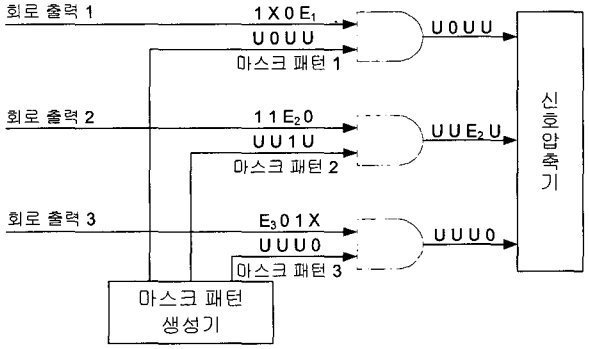


그림 1. 회로출력과 마스크 패턴
Fig. 1. Output bits of a circuit and mask patterns.

X-마스크 관점에서 테스트 대상 회로의 출력 비트들은 다음의 3가지로 분류가 가능하다.

(1) X-값
(2) 고장영향 (E)
(3) 규정값 (0 또는 1)

X-값은 회로 출력의 값이 X-값인 비트로서 신호압축기에서의 신호파괴 현상을 막기 위해 모든 X-값을 갖는 회로 출력 비트에 해당하는 마스크 패턴의 값은 0-값(AND 게이트가 마스크 게이트일 경우)을 가져야 한다. 고장영향(E)은 고장회로의 출력 비트 값과 무고장 회로의 출력 비트 값이 서로 상이하여 고장을 검출할 수 있는 정보를 갖고 있는 비트를 의미하며, 고장검출율을 유지하기 위해서 해당 비트 그대로 신호압축기로 전달되어야 한다. 그러므로 회로 출력의 고장영향에 상응하는 마스크 패턴의 비트는 1의 값으로 결정된다. 마지막으로 규정값은 회로 출력 중 X-값 비트와 고장

영향 비트를 제외한 모든 비트를 의미하며 0 또는 1의 규정값을 갖는다. 규정값 비트에 상응하는 마스크 패턴의 비트는 0 또는 1, 어떤 값도 사용할 수 있다는 의미에서 U로 표시하였다.

원칙적으로, 마스크 패턴은 회로 출력 비트들 중에 오직 X-값 비트만을 마스크하고, 다른 모든 비트(고장 영향 비트들과 규정값 비트)들은 회로 출력 비트 그대로 신호압축기로 전달되도록 생성가능하다. 하지만 이 경우는 마스크 패턴 생성기의 하드웨어가 커지기 때문에 높은 실리콘 면적 비용을 지불해야만 한다.

마스크 패턴 생성기의 하드웨어 오버헤드는 마스크 패턴에 포함된 U-값 비트의 수와 반비례한다. 회로 출력 비트의 규정값에 상응하는 마스크 패턴 비트는 모두 U-값 비트로 할당 할 수 있다. 회로 출력의 고장영향에 해당하는 마스크 패턴의 일부분은 U-값 비트로 할당할 수 있다. 이것은 하나의 고장이 서로 다른 다수의 테스트 패턴에 의해서 서로 다른 출력으로 고장영향을 나타내기 때문이다. 예를 들어 그림 1과 같이 하나의 고장 $f_1$이 서로 다른 회로 출력 비트 $E_1$, $E_2$, 그리고 $E_3$ 이렇게 3개의 비트에 고장영향을 나타낸다고 하자. 단일 고착 고장 검출의 경우 $E_1$, $E_2$, 그리고 $E_3$ 중 최소한 하나의 고장영향만이 신호압축기로 전달 될 수 있도록 마스크 패턴 생성기를 구현하면 충분하다. $E_2$의 고장영향만을 신호압축기로 전달한다고 가정할 경우 $E_2$에 해당하는 마스크 패턴 비트만 1로 결정되고, 나머지 $E_1$과 $E_3$에 해당하는 마스크 패턴 비트는 U로 할당할 수 있다. 본 논문의 X-마스크 회로는 단일 고착 고장 검출을 보장하는 마스크 패턴 생성을 목표로 한다.

## 2. 위상천이기를 이용한 마스크 패턴 생성기

본 논문에서 제안하는 마스크 패턴 생성기는 마스크 패턴을 생성하기 위해 LFSR과 위상천이 네트워크를 사용한다. 위상천이기는 XOR 게이트들로 구성되며 위상천이기의 출력은 $m$-시퀀스의 윈도우 특성(window property)과 쉬프트-덧셈 특성(shift-and-add property)을 갖는다[12]. $m$-시퀀스란 원시 특성 다항식(primitive characteristic polynomial)을 갖는 LFSR에 의해서 생산된 시퀀스를 의미한다.

위상천이 네트워크를 이용한 마스크 패턴 생성기는 다음의 2가지에 이론적 근거를 두고 있다.

(1) 모든 스캔 체인 마스크 패턴은 $m$-시퀀스의 서브시퀀스로 나타난다. 스캔 체인의 길이가 $l$ 일때 최악의 경우 모든 스캔 체인 마스크 패턴은 $l$차 원시 특성 다항식을 갖는 LFSR이 생산하는 $m$-시퀀스의 서브시퀀스로 표현될 수 있다.(윈도우특성)

(2) LFSR에 의해서 생산된 $m$-시퀀스를 $k$번 쉬프트 이동하여 얻을 수 있는 또 다른 $m$-시퀀스는 적절한 LFSR 출력단을 XOR 연산 하여 얻을 수 있다.(쉬프트-덧셈 특성)

이 두 가지 성질은 마스크 패턴 생성을 위해 다음과 같이 사용되어 진다. 그림 2는 위상천이를 이용한 마스크 패턴 생성 시스템의 개념을 보여준다. 설명의 편의상 단일 스캔 체인을 가정하였다.

LFSR은 $x^4+x+1$의 원시 특성 다항식을 갖는 4단 플립플롭으로 구성되었으며, *1000*의 초기값을 갖는다. 테스트 대상 회로는 길이 4의 스캔 체인을 갖는다. $D_0$ $D_1$은 위상선택기의 제어입력으로 정해진 개수의 스캔 체인 마스크 패턴이 생산될 동안 고정된다. LFSR로부터의 10개의 테스트 패턴이 테스트 대상 회로로 인가되어, 스캔 출력으로 4비트 길이 10개의 출력 시퀀스가 발생되는 상황을 생각해보자.

$M$은 10개의 마스크 패턴으로 구성되는 스캔 체인 마스크 패턴 세트이다($M$={$m_1$, $m_2$, ..., $m_{10}$}={UUU1, 0U1U, U0U1, 1UUU, UU10, UUU0, U1U1, 1UU0, UU1U, UUU1}). 위상선택기 입력이 $D_0$ = 1, $D_1$ = 0으로 인가되면 마스크 게이트로 LFSR의 첫 번째 단, 두 번째 단, 그리고 세 번째 단의 출력이 XOR 연산되어 가해진다. LFSR의 첫 번째 단의 출력 시퀀스인 $m$-시

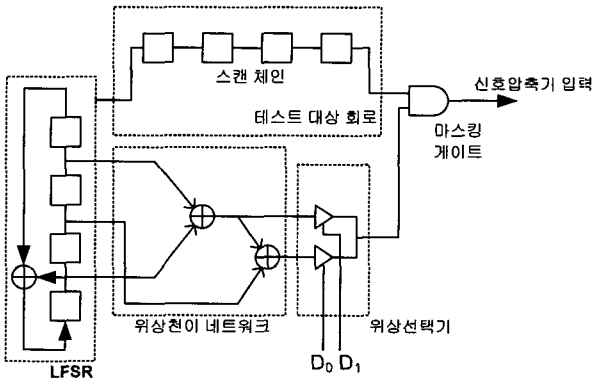


그림 2. 위상천이 네트워크를 이용한 마스크 패턴 생성기 개요

Fig. 2. The concept of mask-pattern generator based on phase-shifting network.

퀀스(111101011001000)를 5번 쉬프트 이동한 시퀀스가 마스크 게이트로 인가된다. LFSR이 초기화되고 처음 4클록 동안 첫 번째 단에서 생산되는 값은 1111이다. 이 때 마스크 패턴은 이를 m-시퀀스에서 5번 쉬프트 이동한 1011이 전달되고, 이는 $m_1$=UUU1을 커버한다. 두 번째 테스트 패턴을 인가하는 동안 LFSR의 첫 번째 단은 0101을 생산하나, 마스크 패턴은 0010가 생산되고 이는 $m_2$=0U1U를 커버 가능하다. $D_0$=1, $D_1$=0를 유지하고 다른 제어 과정 없이 마스크 패턴 $m_8$까지 생산 가능하다. 이제 LFSR에서 9번째 테스트 패턴이 인가된다고 하자. 이때 LFSR의 첫 번째 단의 4클록 동안의 출력은 1101이고, 마스크 패턴은 이를 5번 쉬프트 이동한 1100가 생산된다. 하지만, 1100는 $m_9$=UU1U를 커버 하지 못한다. 이제 8번째 테스트 패턴 생산이 끝나고 9번째 테스트 패턴을 생산하기 전에 $D_0$=0, $D_1$=1로 위상선택기로의 제어 입력이 변경된 경우를 생각해 보자. 그러면 마스크 게이트로 LFSR의 첫 번째, 세 번째 단의 출력이 XOR 연산되어 가해지고, 이는 LFSR의 첫 번째 출력 시퀀스를 $m$-시퀀스에서 7번 쉬프트 이동한 패턴이다. 9번째 테스트 패턴이 인가되는 동안 LFSR의 첫 번째 단 출력 1101를 7번 쉬프트 이동한 0010가 마스크 게이트로 전달되고 $m_9$=UU1U를 커버 가능하다. 마찬가지로 $m_{10}$= UUU1은 $D_0$=0, $D_1$=1로 유지하여 생산할 수 있다. 결론적으로 총 4비트 10개의 마스크 패턴은 LFSR의 출력 시퀀스를 1번의 위상 변경만으로 생산가능하다. 위상선택기의 제어 입력이 고정된 동안을 구성(configuration)이라 한다. 예에서는 2개의 구성을 이용하여 10개의 스캔 체인 마스크 패턴을 생산하였다.

본 논문에서 제안하는 위상천이를 이용한 X-마스크 패턴 생성기의 하드웨어 오버헤드는 전체 마스크 패턴 세트를 생산하는데 필요한 구성의 총 수에 비례한다. 하드웨어 오버헤드를 최소화하기 위해서는 각각의 구성이 가능한 많은 수의 마스크 패턴을 커버할 수 있어야 한다. 다음 장에서는 다중 스캔 체인을 갖는 테스트 대상 회로에 대해 주어진 마스크 패턴 세트를 최소의 구성수로 커버 할 수 있도록 위상천이 네트워크를 설계하는 알고리즘을 개발한다.

## III. X-마스크 하드웨어 합성 알고리즘

그림 3은 본 논문에서 제안하는 X-마스크 회로의 하드웨어 구조를 나타낸다. 위상선택기 블록은 멀티플렉서 스위치로 구성되며, 적당한 $D_0$, $D_1$, ..., $D_{C-1}$의 제어 비트 인가로 재구성 될 수 있다. 파라미터 $C$는 마스크 패턴 생산 동안 사용되는 위상선택기의 총 구성 횟수를 의미하며 시뮬레이션 절차에 의해서 구해진다. 위상천이 네트워크는 XOR 게이트들로 이루어지며, 스캔 체인 마스크 패턴을 생성하기 위해 LFSR 출력의 $m$-시퀀스를 적절하게 위상천이 하는데 사용되어 진다. 위상천이 네트워크의 XOR 게이트 구성은 시뮬레이션 절차를 통해 얻을 수 있다. 제어 입력 $D_0$, $D_1$, ..., $D_{C-1}$은 $d$ 입력 $C$ 출력($d$=log$_2C$) 디코더에 의해서 제공되어 진다. $d$ 비트 구성 카운터는 디코더에 대해 가능한 $2^d$개의 입력 조합을 모두 제공하기 위해서 업카운팅 동작을 하며 패턴 카운터의 값이 1이 될 때 트리거 된다. 패턴 메모리는 각 구성에서 생산할 마스크 패턴의 수가 이진값의 형태로 저장되며, 패턴 카운터는 매 구성 시작 시에 각 구성에서의 마스크 패턴 수로 초기화 되어 다운카운팅 동작을 한다.

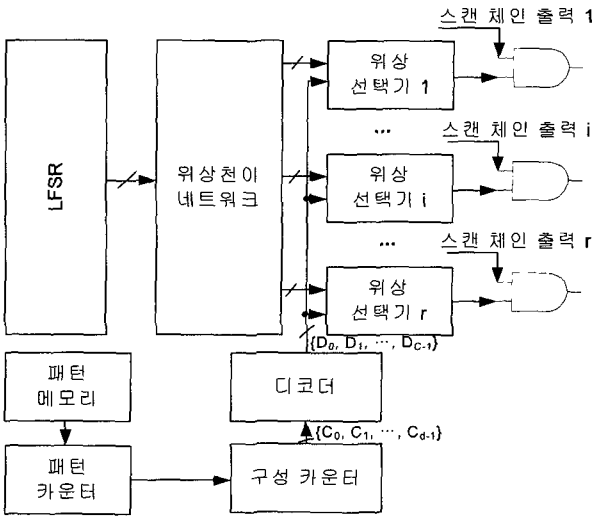


그림 3. 제안하는 X-마스크 회로 구성
Fig. 3. The proposed X-masking circuit architecture.

각각의 구성에서 가능한 많은 수의 마스크 패턴을 생산가능 하기 위해서는 각 스캔 체인의 마스크 게이트로 적절한 위상천이의 선택이 이루어져야 하며, 이것은 그림 4에 나타낸 반복적인 시뮬레이션 절차에 의해서 결정된다. 구성 $i$의 시뮬레이션 절차 동안 $j$번째 스캔 체인의 마스크 게이트로 연결 가능한 위상천이를 나타내기 위해 $P_j^{(i)}$의 기호를 사용한다. 또한 회로의 스캔 체인 총수와 스캔 체인 길이를 나타내기 위해 각각 $s$와 $l$의 기호를 사용한다. 시뮬레이션 절차는 다음과 같다.

```
Function phase_network_synthesis
Input : mask pattern sets M
Output : phase-shift value of each scan chain over all configurations
(a)   set i =1
(b)   initialize PP_table, TP_table ;
      while(M ≠ empty) begin
(c)       initialize P_j^(i) for each scan chain;
(d)       m_c := current mask pattern;
(e)       while(cover_mask_pattern(PP_table, P_j^(i), m_c)) begin
(f)           reduce P_j^(i);
              delete current mask pattern m_c from M;
(g)           update PP_table;
          end while
(h)       store current P_j^(i) to P_j^(i)reduce;
          i=i+1;
      end while
(i)   Select_phase_configuration(P_j^(i)reduce);
end phase_network_synthesis
```

그림 4. 위상천이 네트워크 합성 알고리즘

Fig. 4. Phase-shifting network synthesis algorithm.

(*a*) $i$=1로 설정한다.

(*b*) LFSR의 초기값을 이용하여 *PP_table*과 *TP_table*을 초기화한다. *PP_table*은 LFSR의 첫 번째 단 출력 시퀀스를 $k$번 위상천이 하여 얻을 수 있는 $l$ 비트 패턴이 $k$번째 엔트리인 테이블이다. 예를 들어 그림 5. (*D*)(*a*)의 *PP_table(1)*은 LFSR의 첫 번째 단 출력 시퀀스를 1번 위상천이 할 경우 1110의 4비트 시퀀스를, *PP_table(2)*는 2번 위상천이 할 경우 1101의 4비트 시퀀스를 얻을 수 있음을 뜻한다(*PP_table(k)*는 테이블의 $k$번째 엔트리를 나타냄). *TP_table*은 원하는 위상천이를 얻기 위해 XOR 연산되어야 할 LFSR의 단(stage)을 엔트리로 갖는 테이블이다. 예를 들어 그림 5. (*D*)의 *TP_table(2)*의 1011은 LFSR의 첫 번째 단의 출력을 $m$-시퀀스에서 2번 위상천이 한 $m$-시퀀스를 얻기 위해서는 LFSR의 첫 번째, 세 번째, 그리고 네 번째 단을 XOR 연산해야 함을 의미한다. $m$-시퀀스를 $k$번 위상천이 한 $m$-시퀀스를 얻기 위해 XOR 연산해야 할 LFSR 단을 구하는 방법은 [13]에 나타나있다. 본 논문에서의 *TP_table*은 [13]에 나타난 방법을 이용하여 생성한다.

(*c*) 각 스캔 체인의 $P_j^{(i)}$를 $P_j^{(i)}$ = {1, 2, ..., $L$}로 초기화 한다($j$=1, 2, ..., $s$, $L$=$2^l$-1). 이것은 각 구성의 초기에 각 스캔 체인의 마스크 게이트는 임의의 위상천이를 가질 수 있음을 뜻한다.

(*d*) $m_c$은 현재 테스트 스텝에서의 마스크 패턴이다.

(*e*) $m_c$를 현재의 $P_j^{(i)}$가 포함하는 위상천이로 커버 가능한지 조사한다. 여기서 커버 가능은 다음의 경우를 만족할 때이다. 마스크 패턴 $m_c$는 $m_{c1}$, $m_{c2}$, ...$m_{cs}$의 스캔 체인 마스크 패턴들로 구성된다.

커버가능: 모든 $j$=1,2, ..., $s$에 대해서 *PP_table(k)*와 $m_{cj}$가 상호양립(mutually compatible) 가능한 $k \in P_j^{(i)}$가 존재. 이 때 벡터 $u$={$u_1$, $u_2$, ... $u_z$}는 $v$={$v_1$, $v_2$, ...$v_z$}와 임의의 $i$에 대해서 ($1 \le i \le z$) 다음의 3가지 중 하나가 유효할 때 상호양립하다고 한다.

(1) $u_i$, $v_i$가 모두 규정값일 때 $u_i = v_i$

(2) $u_i$ 가 U-비트

(3) $v_i$ 가 U-비트

(*f*) $P_j^{(i)}$를 다음과 같은 방식으로 축소한다. 각 $j$=1, 2, ..., $s$에 대해서 $F$를 $F \subset P_j^{(i)}$이고, $k \in F$인 임의의 $k$에 대해서 *PP_table(k)*가 $m_{cj}$와 상호양립 하지 않은 서브 세트라고 하자. $P_j^{(i)} = P_j^{(i)} - F$를 연산하여 $P_j^{(i)}$를 축소한다.

(*g*) *PP_table*을 업데이트한다. (*e*)의 과정에서 $l$ 클록 동안 LFSR이 생산하는 패턴을 위상천이 하여 $m_c$를 커버 하였으므로 다음 $l$ 클록 동안 LFSR에서 생산되는 $l$비트 패턴을 기준으로 한 *PP_table*로 변경시켜야 한다. 이는 *PP_table*의 엔트리를 윗방향으로 $l$번 사이클릭 이동하여 얻을 수 있다.

(h) 현재의 구성 $i$에서 더 이상 커버 가능한 마스크 패턴이 존재하지 않으므로 축소된 $P_j^{(i)}$세트를 저장한다. 각 구성에서 최종적으로 축소된 $P_j^{(i)}$를 $P_j^{(i)}{}_{reduce}$라 하자. (*c*)에서 (*h*)의 과정을 마스크 패턴 세트 $M$이 전부 커버 될 때 까지 반복한다.

(*i*) 이 과정은 위상천이 네트워크의 하드웨어 오버헤드를 최소화하는 과정이다. 반복 알고리즘의 수행으로 임의의 구성 $i$에서 스캔 체인 $j$의 $P_j^{(i)}{}_{reduce}$는 하나 이상의 위상천이를 포함하고 있다. 위상천이 네트워크의 XOR 게이트 수를 최소화 하는 방향으로 각 $P_j^{(i)}{}_{reduce}$당 하나의 위상천이를 결정하는 과정이 (*i*)이고 이를 위해 다음을 반복 수행한다.

(1) $k \in P_j^{(i)}{}_{reduce}$($1 \le i \le C$, $1 \le j \le s$, $C$는 구성의 총 수)인 $k$중에 가장 많은 출현수를 갖는 $k_{max}$를 결정한다.

(2) $k_{max}$를 포함하는 $P_j^{(i)}{}_{reduce}$를 $P_j^{(i)}{}_{max}$라 하자. $k_{max}$를 $P_j^{(i)}{}_{max}$의 위상천이로 최종 결정한다. 그리고 $P_j^{(i)}{}_{reduce} = P_j^{(i)}{}_{reduce} - P_j^{(i)}{}_{max}$ 연산을 수행한다.

(3) $P_j^{(i)}{}_{reduce}$={ }이 될 때까지 (1), (2), 그리고 (3)을 반복 수행한다.

(*A*) Mask pattern set

| | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ |
|---|---|---|---|---|---|---|
| Scan chain 1 | 0UUU | UU1U | UUUU | U0UU | UU1U | 0UUU |
| Scan chain 2 | UU1U | UUU0 | U0UU | UUU1 | U1UU | UU1U |
| Scan chain 3 | 1UUU | U1UU | U1UU | UU0U | UUU0 | UU1U |

(*B*) Phase network synthesis

| | *Init.*) | (a) $m_1$ | (b) $m_2$ |
|---|---|---|---|
| $P_1^{(1)}$: | (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) | (4, 6, 9, 10, 12, 13, 14) | (9, 10, 12, 14) |
| $P_2^{(1)}$: | (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) | (1, 3, 5, 6, 9, 13, 14, 15) | (3, 5, 6, 14) |
| $P_3^{(1)}$: | (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) | (2, 3, 5, 7, 8, 11, 15) | (2, 3, 11, 15) |

| (c) $m_3$ | (d) $m_4$ | (e) None : End of confi. 1. |
|---|---|---|
| (9, 10, 12, 14) | (12, 14) | (12, 14) |
| (3, 5) | (3, 5) | (3, 5) |
| (2, 11) | (11) | ( ) |

| | *Init.*) | (f) $m_5$ | (g) $m_6$ |
|---|---|---|---|
| $P_1^{(2)}$: | (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) | (2, 4, 5, 8, 12, 13, 14, 15) | (4, 5, 8, 14) |
| $P_2^{(2)}$: | (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) | (1, 3, 5, 6, 9, 13, 14, 15) | (1, 9, 13, 15) |
| $P_3^{(2)}$: | (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) | (2, 5, 6, 8, 9, 10, 15) | (8, 9, 10, 15) |

(*C*) $P_j^{(i)}{}_{reduce}$

| | (a) *Init.* $P_3^{(1)}$ | (b) $P_1^{(1)}$, $P_1^{(2)}$, $P_2^{(2)}$, $P_3^{(2)}$ | (c) $P_2^{(1)}$ |
|---|---|---|---|
| $P_1^{(1)}$: | (12, 14) | (14) | (14) |
| $P_2^{(1)}$: | (3, 5) | (3, 5) | (3) |
| $P_3^{(1)}$: | (11) | (11) | (11) |
| $P_1^{(2)}$: | (4, 5, 8, 14) | (14) | (14) |
| $P_2^{(2)}$: | (1, 9, 13, 15) | (15) | (15) |
| $P_3^{(2)}$: | (8, 9, 10, 15) | (15) | (15) |

(D) *TP_table* and *PP_table*

| *TP_table* | *PP_table* (a) | (b) | (c) | (d) | (e) | (f) |
|---|---|---|---|---|---|---|
| 1001 | 1110 | 1011 | 0010 | 0011 | 1101 | 0110 |
| 1011 | 1101 | 0110 | 0100 | 0111 | 1010 | 1100 |
| 1111 | 1010 | 1100 | 1000 | 1111 | 0101 | 1001 |
| 0111 | 0101 | 1001 | 0001 | 1110 | 1011 | 0010 |
| 1110 | 1011 | 0010 | 0011 | 1101 | 0110 | 0100 |
| 0101 | 0110 | 0100 | 0111 | 1010 | 1100 | 1000 |
| 1010 | 1100 | 1000 | 1111 | 0101 | 1001 | 0001 |
| 1101 | 1001 | 0001 | 1110 | 1011 | 0010 | 0011 |
| 0011 | 0010 | 0011 | 1101 | 0110 | 0100 | 0111 |
| 0110 | 0100 | 0111 | 1010 | 1100 | 1000 | 1111 |
| 1100 | 1000 | 1111 | 0101 | 1001 | 0001 | 1110 |
| 0001 | 0001 | 1110 | 1011 | 0010 | 0011 | 1101 |
| 0010 | 0011 | 1101 | 0110 | 0100 | 0111 | 1010 |
| 0100 | 0111 | 1010 | 1100 | 1000 | 1111 | 0101 |
| 1000 | 1111 | 0101 | 1001 | 0001 | 1110 | 1011 |

그림 5. 위상천이 네트워크 합성 예

Fig. 5. Example of phase-shifting network synthesis.

예제의 위상천이 네트워크 합성 절차를 그림 5에 나타내었다.

테스트 대상 회로는 3개의 스캔 체인을 가지고 있고 각 스캔 체인은 4비트의 길이를 갖는다. LFSR은 $x^4+x+1$의 원시 특성 다항식과 1000의 초기값을 갖고 의사무작위패턴을 생성한다. 그림 5의 (*A*)는 모든 X-값을 마스크 하도록 0와 1, 그리고 U-값 비트가 할당된 6개의 마스크 패턴 세트 $M = \{m_1, m_2, ...m_6\}$를 나타낸다. 마스크 네트워크는 그림 5의 (*B*)와 같은 반복 절차에 의해서 구해진다.

*Init.*)에서 $P_j^{(1)}$는 각각 1부터 15까지의 원소를 갖도록 초기화된다. 각 원소는 위상천이 수를 의미한다. 예제의 LFSR은 서로 다른 15개의 위상천이를 가질 수 있다. (*B*)(*a*)에서 $P_j^{(1)}$는 (*D*)(*a*)의 *PP_table*을 이용하여 $PP_table(k)(k \in P_j^{(1)})$가 $m_1$와 상호양립 할 수 있는 $k$만을 유지하도록 축소되었다. $m_1$은 (*B*)(*a*)의 과정에서 커버 되고 *PP_table*은 (*D*)(*b*)와 같이 업데이트 된다. 마찬가지 방법으로 $P_j^{(1)}$ 세트의 축소를 진행하면서 (*B*)(*b*), (*B*)(*c*), (*B*)(*d*)에서 볼 수 있는 것과 같이 $m_2$, $m_3$, 그리고 $m_4$를 첫 번째 구성에서 커버한다. 이때 *PP_table*은 현재 테스트 스탭에서의 LFSR 출력을 반영하기 위해 (*D*)(*c*), (*D*)(*d*), (*D*)(*e*)와 같이 업데이트 된다. 이제 5번째 마스크 패턴 $m_5$를 커버하는 과정을 생각해보자. 현재의 *PP_table*은 (*D*)(*e*)의 테이블이다.

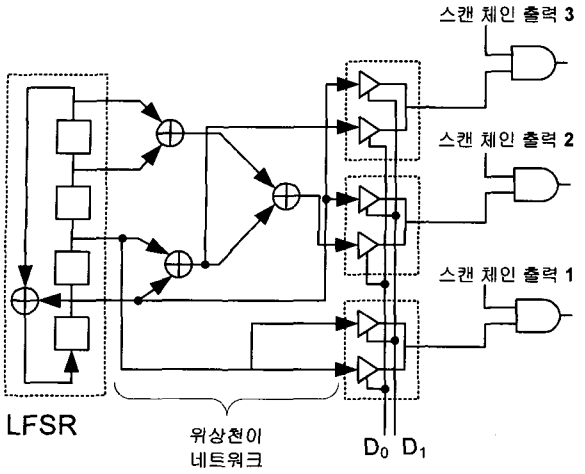


그림 6. 예제의 위상천이 네트워크 구성
Fig. 6. Example phase-shifting network.

(*B*)(*e*)에서 볼 수 있는 것과 같이 $m_{63}$과 상호양립 할 수 있는 $k \in P_3^{(1)}$인 $k$는 존재하지 않는다. 그러므로 $m_5$는 구성 1에서 커버 불가능하다. (*C*)와 같이 $P_j^{(1)}{}_{reduce}$를 저장하고 첫 번째 구성에 대한 시뮬레이션을 마친다. 이때 *PP_table*은 업데이트 하지 않는다. 2번째 구성의 시뮬레이션을 시작하기 위해 $P_j^{(2)}$를 다시 초기화하고 앞과 같이 반복적인 절차를 수행하여 $m_5$, $m_6$를 커버한다. (*B*)의 반복 루프는 각 구성에서 가능한 많은 수의 마스크 패턴을 생산하기 위한 과정 이였다. (*C*)는 위상천이 네트워크에 필요한 XOR 게이트의 수를 최소화하는 과정이다. 각 구성에서 각 스캔 체인의 마스크 게이트에 연결되는 위상천이는 하나만 필요하므로 각 $P_j^{(i)}{}_{reduce}$중 하나의 원소만을 선택한다. $P_3^{(1)}$은 이미 하나의 위상천이 만을 포함하고 있기 때문에 더 이상의 선택 과정은 필요 없다. 위상천이 14와 15는 $P_j^{(i)}$에서 각각 2번씩 가장 많이 발생하므로 $P_1^{(1)}$과 $P_1^{(2)}$는 위상천이 14 그리고 $P_2^{(2)}$와 $P_3^{(2)}$는 위상천이 15로 선택한다. 하나의 위상천이 $k$로 결정되지 않은 $P_2^{(1)}$은 위상천이 3과 5 중 어느 것을 선택하여도 좋다. 결정된 구성 $i$에서 스캔 체인 $j$로의 위상천이를 얻기 위해 XOR 연산해야 하는 LFSR 단들은 *TP_table*에 의해 알 수 있으며, 위상천이 네트워크는 다음 그림 6과 같이 구성된다.

## IV. 실험결과

이번 장에서는 ISCAS'89 벤치마크 회로에 대해 본 논문에서 제안한 위상천이를 기반으로 하는 X-마스크 기법의 효율성을 증명한다.

표 1. ISCAS'89 벤치마크 회로
Table 1. ISCAS'89 benchmark circuit.

| Circuit | Outputs | No. of Faults | No. of scan chain | Circuit Size(GE) |
|---|---|---|---|---|
| s298 | 20 | 610 | 2 | 156 |
| s344 | 26 | 662 | 3 | 210 |
| s400 | 27 | 778 | 3 | 213 |
| s444 | 27 | 754 | 3 | 232 |
| s526 | 27 | 932 | 3 | 245 |
| s713 | 42 | 908 | 5 | 489 |
| s5378 | 228 | 6536 | 32 | 3221 |
| s13207 | 790 | 15480 | 32 | 9441 |
| s15850 | 684 | 18130 | 32 | 11067 |
| s38584 | 1730 | 60350 | 32 | 22447 |

표 1은 실험에 사용된 벤치마크 회로에 관한 정보를 나타낸다. 2번째 열은 각 회로의 출력수를 나타내며 주 출력(primary input) 수와 스캔 셀을 이용한 유사출력(pseudo-primary output)의 수를 합한 수이다. 3번째 열은 회로내 존재하는 0-고착고장의 수와 1-고착고장의 수를 합한 고착 고장의 총수 이다. 5번째 열의 "Circuit size"는 게이트등가수치(gate equivalent value)로 환산한 벤치회로의 크기 이다. $n$-입력 NAND 또는 NOR 게이트는 0.5$n$, 그리고 인버터에 대해서는 0.5의 게이트등가수치를 사용하였다. 또한 트랜스미션 게이트는 0.5, 그리고 플립플롭에 대해서는 6의 게이트등가수치를 적용하였다(RS 회로에 대해 2 게이트, 멀티플렉서에 대해서 3 게이트, 그리고 에지 핸들링을 위해 1게이트). 실험에서 회로 입력으로의 테스트 패턴 생성은 고려하지 않았다. 정해진 테스트 패턴 시퀀스의 생성은 기존의 많은 연구들의 결과를 이용하여 가능하기 때문에, 본 논문의 실험에서 각 마스크 패턴의 생성 순서는 재배치 할 수 있는 것으로 가정하였다. 실험의 편의를 위해 실험에 사용된 벤치마크 회로들은 모든 스캔 체인의 길이가 동일한 균형(balanced) 스캔 체인을 갖는다고 가정한다.

실험은 2가지 형태로 진행하였다. 실험 1은 ATPG를 통해 얻은 패턴을 테스트 패턴으로 사용 하였고, 실험 2는 의사무작위로 생성된 패턴을 테스트 패턴으로 사용하였다.

표 2와 표 3은 각각 실험 1과 실험 2에 대해 본 논문에서 제안한 X-마스크 회로의 성능을 보여준다. 벤치회로들은 회로에 X-값 발생원을 가지고 있지 않기 때문에 회로 출력에 X-값이 나타나지 않는다. 실험을 위해 $p$=0.2%의 확률로 랜덤하게 X-값이 나타나는 회로 출력을 사용하였다.

표 2. ATPG 테스트 패턴에 대한 X-마스크 실험 결과($p$=0.2%)
Table 2. Experimental results for ATPG test patterns.

| Circuit | $P$ | $S$ | $S/P$ | *confi.* | XOR | storage requirement |
|---|---|---|---|---|---|---|
| s298 | 41 | 114 | 2.78 | 6 | 13 | 24 |
| s344 | 38 | 122 | 3.21 | 5 | 13 | 20 |
| s400 | 43 | 128 | 2.98 | 3 | 10 | 15 |
| s444 | 46 | 126 | 2.74 | 4 | 13 | 20 |
| s526 | 55 | 171 | 3.11 | 5 | 13 | 25 |
| s713 | 59 | 176 | 2.98 | 4 | 16 | 24 |
| s5378 | 284 | 1383 | 4.87 | 5 | 44 | 40 |
| s13207 | 429 | 4132 | 9.63 | 19 | 62 | 133 |
| s15850 | 513 | 3710 | 7.23 | 17 | 71 | 119 |
| s38584 | 824 | 13149 | 15.96 | 24 | 109 | 168 |

표 3. 유사무작위 테스트 패턴에 대한 X-마스크 실험 결과($p$=0.2%)
Table 3. Experimental results for pseudorandom test patterns.

| Circuit | $P$ | $S$ | $S/P$ | *confi.* | XOR | storage requirement |
|---|---|---|---|---|---|---|
| s298 | 600 | 103 | 0.17 | 1 | 5 | 10 |
| s344 | 300 | 101 | 0.34 | 1 | 7 | 9 |
| s400 | 500 | 97 | 0.19 | 1 | 6 | 9 |
| s444 | 600 | 108 | 0.18 | 1 | 6 | 9 |
| s526 | 2600 | 157 | 0.06 | 1 | 6 | 12 |
| s713 | 2500 | 164 | 0.07 | 1 | 7 | 12 |
| s5378 | 10000 | 1027 | 0.10 | 2 | 29 | 26 |
| s13207 | 42000 | 4085 | 0.10 | 2 | 35 | 30 |
| s15850 | 1200 | 3350 | 2.79 | 4 | 68 | 40 |
| s38584 | 30000 | 12752 | 0.43 | 3 | 61 | 45 |

표 4. 기존 연구결과와의 비교
Table 4. Result comparison to previous works.

| Circuit | Naruse et al., [9] | | | | Tang et al., [11] | | | Proposed | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $p$=0.2 | | | | | | | (a) $p$=0.2% | | | (b) $p$=0.2% | | | |
| | $P$ | $S$ | $D$ | H/W | $P$ | $p$ | H/W | $P$ | $S/P$ | H/W | $P$ | $S/P$ | H/W | Minimal H/W |
| s298 | 600 | 2 | 14 | 60% | 25 | 1.6% | 6% | 41 | 2.78 | 57% | 600 | 0.17 | 45% | 5% |
| s344 | 300 | 1 | 20 | 60% | 16 | 2.6% | 6% | 38 | 3.21 | 57% | 300 | 0.34 | 31% | 4% |
| s400 | 500 | 1 | 19 | 57% | 28 | 2.2% | 12% | 43 | 2.98 | 46% | 500 | 0.19 | 30% | 4% |
| s444 | 600 | 4 | 10 | 31% | 28 | 1.2% | 6% | 46 | 2.74 | 44% | 600 | 0.18 | 28% | 4% |
| s526 | 2600 | 4 | 30 | 87% | 55 | 1.5% | 13% | 55 | 3.11 | 49% | 2600 | 0.06 | 27% | 4% |
| s713 | 2500 | 9 | 26 | 44% | 31 | 1.2% | 4% | 59 | 2.98 | 23% | 2500 | 0.07 | 14% | 2% |
| s5378 | 10000 | 103 | 30 | 30% | 121 | 7.3% | 10% | 284 | 4.87 | 7% | 10000 | 0.10 | 4% | 2% |
| s13207 | 42000 | 462 | 31 | 40% | 276 | 1.3% | 14% | 429 | 9.63 | 6% | 42000 | 0.10 | 2% | 1% |
| s15850 | 1200 | 100 | 28 | 8% | 139 | 2.2% | 11% | 513 | 7.23 | 5% | 1200 | 2.79 | 3% | 1% |
| s38584 | 30000 | 1002 | 21 | 24% | 147 | 2.2% | 15% | 824 | 15.96 | 4% | 30000 | 0.43 | 2% | 1% |

표 2의 첫 번째 열과 두 번째 열은 벤치회로의 이름과 실험에 사용된 테스트 패턴의 수를 나타낸다. 마스크 패턴은 단일 고착 고장 검출을 보장하고, 모든 X-값 출력은 마스크 되어 0-값이 신호압축기로 전달되도록 생성되었다.

"$S$" 열은 마스크 패턴에 존재하는 규정값(0 또는 1) 비트의 수를 의미한다. 또 "$S/P$" 열은 S를 패턴 수($P$)로 나눈 값이며, 마스크 패턴 1개당 포함되는 평균 규정값 비트의 수를 의미한다. "confi."열은 테스트 과정 동안 모든 마스크 패턴 세트를 생성하기 위해 스캔 체인 마스크 게이트로의 위상 선택을 달리해야 하는 구성의 총수를 의미한다. "XOR" 열은 각 회로의 위상천이 네트워크의 구현에 필요한 XOR 게이트의 총수이다. 마지막으로 "storage requirement" 열은 패턴 메모리의 크기를 의미하며 다음의 수식을 이용하여 구해진다.

$$storage\ requirement = confi. \times < \log_2 P_i^{max} > \qquad (2)$$

여기서 $P_i^{max}$는 각 구성에서 인가되는 마스크 패턴의 수

중에서 최대값을 의미한다.

위상천이를 이용한 X-마스크 회로의 하드웨어 오버헤드는 전체 마스크 패턴을 생성하는데 필요한 구성의 총수가 좌우한다. 즉 구성의 총 수가 작을수록 하드웨어 오버헤드를 줄일 수 있다. 구성의 총수는 동일한 위상천이를 통해 최대한 많은 스캔 체인 마스크 패턴을 커버할 수 있을 때 최소가 되며, 이 가능성은 스캔 체인 마스크 패턴에 포함된 U-비트의 수가 많을수록 증가한다. 이 결과는 표 2와 표 3의 실험 결과를 통해 확인 할 수 있다. "*S/P*" 열은 각 마스크 패턴당 0 또는 1의 값을 갖는 비트의 수를 의미한다. 그러므로 스캔 체인의 총수를 *s*, 스캔 체인의 길이를 *l*이라 할 때 각 스캔 체인 마스크 패턴에는 평균적으로 $l-S/(s\times P)$개 만큼의 U-값 비트를 포함하게 되고, 이때 스캔 체인 마스크 패턴을 생성 가능한 $2^{1-\frac{S}{s\times P}}$개의 위상천이 후보를 갖는다. *S/P*의 값이 작을수록 마스크 패턴을 생성하기 위한 총 구성의 수가 작아지는 것을 실험 결과를 통해 관찰 할 수 있다.

표 4는 본 논문의 실험 결과와 기존 연구의 결과를 비교한 결과이다. [11]에서는 회로 입력에 $p$=0.2%의 확률로 X-값을 삽입하고 회로를 통해 X-값을 전달하는 실험 방법을 사용하였고, 그에 따른 회로 출력에서의 X-값의 비율을 "*p*" 열에 나타내었다. [9]의 실험결과에서 "*S*"열은 LFSR 시드(seed)의 총수, "*D*"열은 LFSR의 단(stage) 수를 의미한다. "*P*" 열은 실험에 사용된 패턴의 수이다. 표 4의 본 논문의 실험결과에서 (*a*)는 ATPG를 통한 패턴을 테스트 패턴으로 사용한 경우이고, (*b*)는 의사무작위 패턴을 테스트 패턴으로 사용한 경우이다. (*a*)의 경우 고장검출율은 모든 회로에 대해 100%이고, (*b*)의 경우는 [9]의 고장 검출율과 거의 유사하기 때문에 표 4에는 나타내지 않았다. 테스트 대상 회로에 대한 X-마스크 회로의 하드웨어 오버헤드는 "H/W" 열에 나타내었다. 제안된 X-마스크 회로의 하드웨어 오버헤드는 LFSR, 위상천이 네트워크, 패턴 메모리, 패턴 카운터, 구성카운터, 디코더, 그리고 위상선택기 전부를 고려한 수치이다. 여기서, 구성수가 1인 경우(표 3참고)는 모든 마스크 패턴이 하나의 위상천이를 유지하여 생산 가능한 경우로, 이때는 오직 LFSR과 위상천이 네트워크만을 필요로 하고 패턴 메모리, 패턴 카운터, 구성카운터, 디코더, 그리고 위상선택기는 필요하지 않다. LFSR과 패턴 메모리의 하드웨어 오버헤드 산출은 [11]에서 사용한 방법과 동일하게, $GE_{LFSR} = 6\cdot D+2$ 그리고 $GE_{메모리}$ = (storage requirement)/4의 식을 이용하였다. 여기서 $D$는 LFSR의 단수를 의미한다.

표 4를 통해 볼 수 있는 것과 같이 위상천이를 이용한 X-마스크 회로는 s5378, s13207, s15850, 그리고 s38584와 같이 크기가 큰 4개의 회로에 대해서는 기존의 연구에 비해 크게 35% 정도의 하드웨어 오버헤드 감축의 효과가 있다. 하지만, 크기가 작은 4개의 회로에 대해서는 [9] 보다는 작지만 [11] 보다는 높은 하드웨어 오버헤드를 갖는 것을 볼 수 있다. 이것은 s298, s344, s400, s444, s526, 그리고 s713의 벤치회로는 회로의 크기가 작아, LFSR 자체의 하드웨어가 상대적으로 크게 작용하기 때문이다. 일반적으로 내장된 자체 테스트는 LFSR을 패턴 생성기로 사용한다. 본 논문의 위상천이를 이용한 X-마스크 회로는 마스크 패턴을 생성하기 위해 패턴 생성기에서 생성되는 $m$-시퀀스를 적절히 위상천이 하여 사용하기 때문에 패턴 생성기와 LFSR을 공유할 수 있다. 패턴 생성기와 X-마스크 회로가 LFSR을 공유하는 경우 X-마스크 회로의 하드웨어 오버헤드는 LFSR 면적을 고려하지 않아도 되기 때문에 더욱 줄어들 수 있다. 표 4의 마지막 열인 "Minimal H/W" 열은 패턴 생성기와 LFSR을 공유할 경우 테스트 대상 회로 면적 대비 제안하는 X-마스크 회로의 하드웨어 오버헤드를 의미한다. 본 논문에서 제안하는 X-마스크 회로는 기존의 연구결과와 비교해 적은 하드웨어를 필요로 함을 확인할 수 있다.

## V. 결 론

출력으로 X-값을 생산하는 로직 블록들은 X-값에 의해서 신호압축치가 손상되기 때문에 내장된 자체 테스트의 구현시 큰 걸림돌로 작용한다. 이런 모듈의 출력에서 X-값을 마스크 하는 기술은 X-값에 의한 신호압축치의 손상 가능성을 원천적으로 봉쇄할 뿐만 아니라, 신호압축기로 스페이스콤팩터 또는 타임콤팩터 등 어떤 형태의 콤팩터 기술도 사용할 수 있기 때문에 높은 응답 패턴 압축율을 제공할 수 있다. 회로내 대부분의 고장들은 다수의 테스트 패턴에 의해서 검출가능하기 때문에, 회로 출력의 많은 규정비트들은 고장검출율의 손실 없이 X-마스크 회로에 의해서 마스크 될 수 있다.

본 논문에서는 $m$-시퀀스의 위상천이 성질을 이용한

X-마스크 회로를 제안하였다. 위상천이를 이용한 X-마스크 회로는 회로 출력의 모든 X-값을 마스크 가능하고, 고착 고장에 대한 단일 검출이 가능하기 때문에 고장검출에 영향을 미치지 않는다. 위상천이를 이용한 X-마스크 회로는 스캔 체인 마스크 패턴에 존재하는 규정 비트의 수가 작을수록 적은 하드웨어 오버헤드를 필요로 함이 실험을 통해 증명되었다. 본 논문에서는 최소의 위상천이 변경으로 전체 마스크 패턴을 생산할 수 있는 반복적인 시뮬레이션을 이용한 하드웨어 합성 알고리즘을 개발하였다. 제안된 X-마스크 기법은 부가적인 하드웨어 블록을 필요로 하지만, 마스크 패턴을 생성하기 위한 총 구성의 효과적인 감축으로 기존의 연구 결과보다 적은 하드웨어 오버헤드를 필요로 한다.

## 참 고 문 헌


[1] 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.

[2] 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.

[3] K. K. Saluja and M. Karpovsky, "Testing computer hardware through data compression in space and time," *Proc. of International Test Conference,* pp. 83-88, 1983.

[4] Mitra and K. S. Kim, "X-Compact: an efficient response compaction technique for test cost reduction," *Proc. of International Test Conference,* pp. 311-320, 2002.

[5] J. Rajski, C. Wang, J. Yuszer, and S. M. Reddy, "Convolutional compaction of test responses," *Proc. of International Test Conference,* pp. 745-754, 2003.

[6] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, and B. Koenemann, "OPMISR: The foundation for compressed ATPG Vectors," *Proc. of International Test Conference,* pp. 748-757, 2001.

[7] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko, B. Keller, D. Soctt, and B. Koenemann, "Extending OPMISR beyond 10x scan test efficiency," IEEE Design & Test of Computers, Vol. 19, no. 5, Oct., pp. 65-73, 2002.

[8] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K. H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded deterministic test for low cost manufacturing test," *Proc. of International Test Conference,* pp. 301-310, 2002.

[9] M. Naruse, I. Pomeranz, S. M. Reddy, and S. Kundu, "On-chip compression of output responses with unknown values using LFSR reseeding," *Proc. of International Test Conference,* pp. 1060-1068, 2003.

[10] T. Clouqueur, K. Zarrineh, K. K. Saluja, and H. Fujiwara, "Design and analysis of multiple weight linear compactors of responses containing unknown values," *Proc. of International Test Conference,* pp. 2005.

[11] Y. Tang, H. J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, P. Engelke, I. Polian, and B. Becker, "X-Masking During Logic BIST and Its Impact on Defect Coverage," *Proc. of International Test Conference,* pp. 441-451, 2004.

[12] P. H. Bardell, W. H. Mcanney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Technique,* New York: Wiley, 1987.

[13] J. Rajski and J. Tyszer, "Design of phase shifters for BIST applications," *Proc. of VLSI Test Symposium,* pp. 218-224, 1998.

## 저 자 소 개

송 동 섭(학생회원)
2000년 건국대학교 전기공학과 학사 졸업.
2002년 연세대학교 전기전자 공학과 석사 졸업.
2007년 현재 연세대학교 전기전자 공학과 박사과정.
<주관심분야 : DFT, SoC Testing, 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.
2007년 현재 연세대학교 전기전자공학과 교수.
<주관심분야 : SoC 설계 및 SoC 테스트>