論文 92-29B-9-1

CMOS 테스트를 위한 Built-In Self-Test 회로 설계

(A Built-In Self-Test Method for CMOS Circuits)

金 倫 弘,* 林 寅 七,*

(Yun Hong Kim and In Chil Lim)

要 約

본 논문에서는 CMOS 회로에 적합한 Built-in self-test 방법을 제안한다. CMOS회로에서 발생되는 stuck-open 고장은 그 특성상 2개의 연속적인 테스트 패턴이 인가되어야 검출되므로, 이러한 특성을 고려하여 테스트 패턴을 생성하는 회로를 설계하여야 한다. 본 논문에서 제안되는 패턴결합 및재배열 알고리듬에 따라 설계된 Built-in self-test 회로는 제어 신호에 따라 테스트 패턴 생성시에는 NLFSR(Nonlinear Feedback Shift Register)로 동작하고, 신호분석시에는 LFSR(Linear Feedback Shift Register)로 동작하게 된다. NLFSR에 의해 생성되는 테스트 패턴은 pseudo-exhaustive하게 생성되는 것이 아니라 필요한 패턴만이 지정된 순서대로 생성되도록 하므로써, CMOS 회로에 대한 테스트가 효과적으로 수행될 수 있다.

Abstract

This paper proposes a built-in self-test technique for CMOS circuits. To detect a stuck-open fault in CMOS circuits, two consequent test patterns is required. The ordered pairs of test patterns for stuck-open faults are generated by feedback shift registers of extended length. A nonlinear feedback shift register is designed by the merging method and reordering algorithms of test patterns proposed in this paper. And a new multifunctional BILBO (Built-In Logic Block Observer) is designed to perform both test pattern generation and signature analysis efficiently.

I. 서 론

VLSI기술의 발전으로 칩의 집적도가 증가하고 회로가 복잡하여짐에 따라 그 테스트는 심각한 문제로 대두되고 있다. 종래의 조합논리회로의 테스트는 제조과정 또는 사용 중에 발생할 수 있는 여러 형태의 물리적인 결함이나 고장 등을 모델화하고 각 고장모델에 대하여 테스트 패턴을 생성한 다음 이 테스트 패턴을 테스트하고자 하는 회로 (circuit under test : CUT)에 인가하여 결과적인 응답과 고장이 없는 응답을 비교함으로써 행해진다. 그러나 논리회로가 복잡하고 규모가 큰 VLSI에서는 기존의 고장모델 이외에 새로운 고장모델을 고려하여야 하며 이에 대하여 테스트 패턴을 생성하는 것과 이 테스트 패턴에 의하여 고장을 진단하는 과정이 매우 어렵다. 특히 테스트 패턴을 생성하기 위한 비용은 회로의 크기에 대하여 지수함수적으로 증가[1]하므로 테스트 과정에 많은 비용이 들게 된다. 그러므로 이와같은 대규모 회로에 있어서의 테스트에 대한 문제점을 해결하기 위하여 논리회로 설계시, 테스트를 용이하게 할 수 있도록 부가회로를 사용

*正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1991年 8月 16日

하는 논리 설계방식이 제시되었다. [2-3]

한편 VLSI의 등장으로 수많은 트랜지스터가 한개의 칩에 집적됨에 따라 과다한 전력소비는 VLSI사용의 근본적인 제한요소가 되었다. 그러므로 낮은 전력소비와 고집적도 등의 특성을 갖는 CMOS가 VLSI의 중요한 구성소자로 채택되고 있다.[4-6]

CMOS 회로에는 stuck-at 형태의 고전적인 고장 모델 외에도 회로의 특성에 따라 stuck-open(s-op)이라는 고장 모델[7]이 존재한다. CMOS 회로에 s-op고장이 발생할 경우, 회로의 출력은 전하 저장 기능으로 인하여 전 상태값을 그대로 유지하므로 조합 논리 회로가 순서 논리 회로와 같이 동작한다. 따라서 이와 같은 고장을 검출하기 위해서는 일정한 순서를 갖는 두 개의 연속적인 테스트 패턴들이 필요하게 된다.

기존의 built-in self-test 방식을 CMOS회로 테스트에 그대로 적용하면, LFSR(linear feedback shift register)에 의해 pseudo-exhaustive하게 테스트 패턴이 생성되므로 s-op고장을 검출할 수 없다.

본 논문에서는 CMOS회로 테스트에 적합한 built-in self-test 방법을 제안한다. S-op고장 검출에 필요한 초기화 패턴과 테스트 패턴을 연속적으로 생성할 수 있는 NLFSR(Nonlinear Feedback Shift Register)이 패턴 결합과 재배열 알고리듬에 의하여 효과적으로 설계된다.

Ⅱ. CMOS 회로의 고장모델

CMOS회로는 pMOS트랜지스터 블럭 (p블럭)과 nMOS트랜지스터 블럭 (n블럭)이 대칭적으로 연결되어 구성된 회로이다.(그림 1)

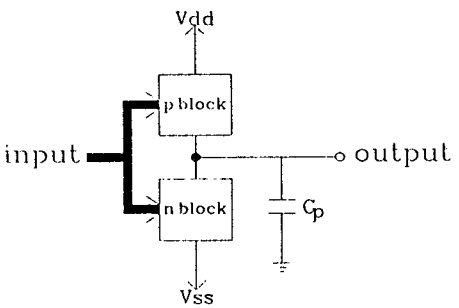

그림 1. CMOS 회로
Fig. 1. CMOS circuit.

그림 1과 같은 CMOS 회로에는 stuck-at-1(s-a-1), stuck-at-0(s-a-0)와 같은 고전적인 고장이외에 stuck-open(s-op) 고장이 발생할 수 있다. CMOS 회로의 stuck-open 고장은 p블럭이나 n블럭내의 트랜지스터 open고장 등으로 인하여 발생하는 것으로, 출력선은 high-impedance상태가 되어 C_p에 저장된 전상태값이 출력된다. 따라서 s-op고장이 발생하면 조합논리회로가 마치 순서논리회로처럼 동작하게 된다.

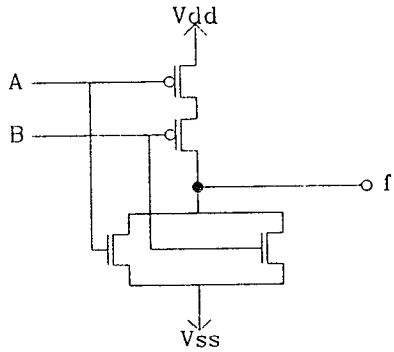

그림 2. 2입력 CMOS NOR 회로
Fig. 2. A 2-input CMOS NOR circuit.

그림 2는 2입력 CMOS NOR회로를 나타낸 것이다. 이 회로에서 s-op고장들은 각 입력 A, B와 연결된 nMOS트랜지스터의 open 또는 missing에 의해 발생하거나, VDD의 open에 의해 발생할 수 있다. 표 1은 2입력 CMOS NOR회로에서 고장 및 정상 상태에 대한 진리표이다.

표 1. CMOS NOR회로에 대한 진리표
Table 1. A truth table of the CMOS NOR circuit.

input	output	classical fault				non-classical fault		
A B	1	F_0	F_1	F_A	F_B	F_{ASOP}	F_{BSOP}	F_{VDDSOP}
1 0	0	0	1	1	1	1	1	U
0 1	0	0	1	0	1	0	U	0
1 0	0	0	1	1	0	U	0	0
1 1	0	0	1	0	0	0	0	0

여기서 $F_0(F_1)$은 출력단의 s-a-0 (s-a-1) 고장 발생 경우의 출력함수를 나타내고, FA(FB)는 입력 A(B)의 s-a-0 고장의 경우를 나타낸다. 또한 $F_{ASOP}(F_{BSOP})$는 입력 A(B)와 연결된 nMOS트랜지스터의 s-op고

장 발생 경우의 출력함수를 나타내고, F_{VDDSOP}는 VDD에서의 s-op고장의 경우를 나타낸다. U는 s-op고장이 발생했을 때 출력이 전 상태의 값을 유지함을 나타내는 것으로 s-op고장이 발생하면 조합논리회로가 순서논리회로와 같이 동작하는 것을 의미한다.

NOR회로에서 ASOP고장이 발생한 경우, 입력선 AB에 10을 인가하면 출력은 전 상태값인 0가 된다. 한편 고장이 없는 경우에도 10을 인가하면 0을 출력하므로, 고장과 정상 상태의 출력값이 동일하여 ASOP고장은 검출할 수 없게 된다. 그러므로 2입력 NOR회로의 고장을 검출하기 위하여 표 1의 입력을 순서대로 인가한다면 NOR회로에서의 ASOP과 VDDSOP고장을 검출할 수 없게 된다.

이와 같이 s-op고장을 검출하기 위해서는 일정한 순서를 갖는 두개의 연속적인 테스트 패턴, 즉 초기화 패턴 (T_0)과 테스트 패턴(T_1)이 인가되어야 한다. 초기화 패턴은 전상태값 U를 초기화하기위한 패턴으로서, 테스트 패턴 T1에 의한 출력값과 반대되는 값으로 U를 초기화해야 한다.

Ⅲ. CMOS회로의 Built-in self-test

Built-in self-test의 기본 형태는 그림 3과 같이 CUT의 입력측과 출력측에 테스트 패턴 생성과 테스트 응답 처리를 위한 회로를 각각 부가하는 것이다.

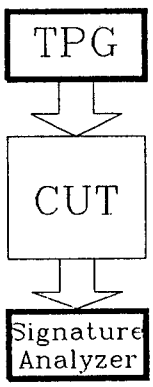

그림 3. Built-in self-test의 기본 구조
Fig. 3. Basic structure of Built-in self-test.

TPG(Test Pattern Generator)에서 테스트 패턴을 생성하여 CUT에 입력시키고 그 응답을 압축하고 저장하는 기능은 Signature Analyzer에서 하게 된다. Signature Analyzer에서 테스트 응답처리 방식은 CRC(Cyclic Redundancy Check) 코드에 의해서 이루어진다.[10] 기존의 TPG와 Signature Analyzer는 일반적으로 LFSR를 사용하여 설계된다. LFSR에 의한 테스트 패턴 생성은 CUT에 입력가능한 패턴이 pseudo-exhaustive하게 모두 발생되므로 CMOS의 s-op고장 검출을 위해 특정 패턴들이 순서적으로 인가되기에는 부적합하다.

CMOS회로의 s-op고장을 검출하기 위하여 초기화 패턴과 테스트 패턴이 연속적으로 인가되도록, 두 패턴을 동시에 저장하기 위하여 확장된 길이를 갖는 레지스터를 사용한다. 그림 4는 각각 m비트로 구성된 초기화 패턴과 테스트 패턴을 연속적으로 생성할 수 있는 NLFSR의 일반 구성도이다. 주어진 패턴에 따라서 feedback함수를 설계함으로써 원하는 패턴을 지정된 순서대로 생성할 수 있다.

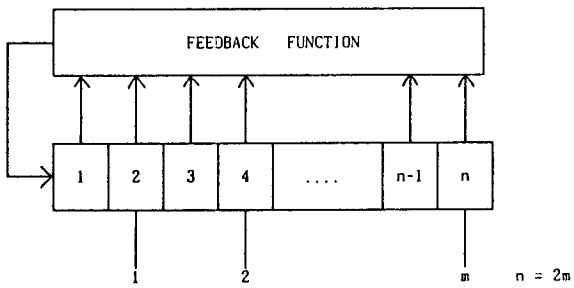

그림 4. CMOS 조합논리회로를 위한 일반적인 TPG
Fig. 4. A general TPG for CMOS combination circuits.

여기서 쉬프트 레지스터의 출력은 매 두번째 레지스터로부터 나오므로 m개의 입력을 가지는 CMOS회로에 대한 TPG를 구성하는 쉬프트 레지스터의 길이는 2m이 되어야 한다.

CMOS 조합 회로를 테스트하기 위하여 주어진 결합 시이퀀스(초기화 패턴과 테스트 패턴)를 발생시키는 TPG, 즉 NLFSR을 설계하기 위한 절차는 다음과 같다.

1. 초기화 패턴과 테스트 패턴들을 쉬프트 레지스터의 단일 벡터로 결합하여, 상태 벡터를 구성한다.
2. 쉬프트 레지스터의 상태 벡터들을 재배열한다.
3. 레지스터의 길이와 feedback 함수를 결정한다.

이와 같은 단계로 설계된 NLFSR은 쉬프트 레지스터가 매 클럭마다 초기화 패턴과 테스트 패턴을 연속적으로 생성할 것이다.

일반적인 조합 회로의 경우 테스트 패턴들을 임의의 순서로 인가하여 고장을 검출할 수 있으나, CMOS회로의 s-op고장의 경우 NLFSR로부터 초기화 패턴과

테스트 패턴을 순서대로 얻어야 하므로 다음과 같은 정의에 의하여 패턴을 결합한다.

[정의 1] : 테스트 패턴과 초기화 패턴들을 각각 1비트씩 교대로 배열하여 하나의 패턴으로 작성하는 것을 패턴 결합 이라 한다. 이때 패턴의 길이는 두배로 확장되며 배열 순서는 테스트 패턴의 비트를 먼저 배열하고 그 다음에 초기화 패턴의 비트를 배열한다.

그림 5는 결합 시이퀀스들의 패턴 결합 과정을 나타낸 것으로 T_0, T_1은 각각 초기화 패턴과 테스트 패턴이다. 이들 패턴을 연속적으로 생성하기 위하여 두 패턴들이 쉬프트 레지스터의 단일 상태로 결합된다. 쉬프트 레지스터에 상태 벡터가 저장되면, 초기화 패턴은 매 두번째 레지스터, 즉 NLFSR의 출력에 할당되고 한번 더 쉬프트하면 테스트 패턴이 NLFSR의 출력에 할당되어 고장 검출을 위한 결합 시이퀀스가 연속적으로 생성된다. 이것은 연속된 두 클럭내에 초기화 패턴 T_0와 테스트 패턴 T_1이 연속적으로 발생하여 s-op고장을 테스트할 수 있음을 나타내는 것이다.

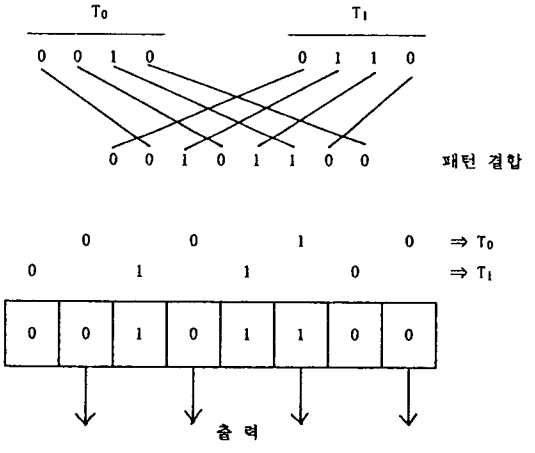

그림 5. 패턴 결합
Fig. 5. Pattern merging.

패턴 결합이 끝나면 쉬프트 레지스터의 단일 벡터들은 그 순서를 바꾸어도 CMOS회로의 s-op고장을 검출할 수 있다. 따라서 쉬프트 레지스터 상태 벡터들은 적절하게 그 순서를 바꾸어 NLFSR의 feedback 함수를 효과적으로 설계할 수 있도록 한다. 이것을 재배열이라고 한다. 즉, 쉬프트 레지스터의 feedback 함수에 의해서 레지스터의 상태 벡터가 바뀌므로 쉬프트 레지스터의 상태 벡터들의 집합 T_m(패턴 결합된 테스트 패턴들)을 재배열하는 것을 의미한다.

패턴 결합된 벡터들의 집합 T_m에 대한 재배열 알고리듬은 다음과 같다.(쉬프트 레지스터 상태 벡터의 재배열 알고리듬)

[단계 1] 상태벡터 집합 T_m에서 임의의 상태벡터를 선택하여 S라 한다. 더 이상 선택할 상태 벡터가 없으면 단계 8로 간다.

[단계 2] k를 1로 정한다.

[단계 3] 상태 벡터 S를 k번 shift하여 S_k를 구한다.

[단계 4] 만약 S_k가 T_m에 존재하면, T집합의 최종 상태 벡터와 S_k간의 연결 벡터를 작성하여 T집합에 저장한다. S_k가 T_m에 존재하지 않으면 단계 1로 간다.

[단계 5] 상태 벡터 S를 T_m에서 제거시킨다.

[단계 6] k를 1만큼 증가시킨다.

[단계 7] 단계 3으로 간다.

[단계 8] 종료

NLFSR의 최소 길이는 패턴 결합된 상태 벡터들의 길이에 의하여 결정된다. 이때 모든 상태 벡터들은 서로 다른 상태, 즉 상태 벡터들의 유일성을 전제로 하여야 한다. 동일한 재배열된 상태 벡터들이 두번 또는 그 이상 존재하는지 조사되어야 한다. 만약 두번 이상 동일한 상태 벡터들이 존재하게 되면 상태 벡터들은 그 다음 벡터의 첫 번째 비트를 사용하여 확장되어야 한다.

NLFSR의 설계를 위한 마지막 단계로서, NLFSR의 feedback 함수는 다음과 같은 방법에 의하여 결정된다. 재배열된 상태 벡터들은 쉬프트 레지스터의 쉬프트 동작에 의하여 생성되므로 현재의 상태 벡터가 다음 상태 벡터의 첫번째 비트를 결정하게 된다. 현재의 상태 벡터를 입력으로 하고 다음 상태 벡터의 첫번째 비트값을 출력값으로 하는 논리함수를 구성하고 논리를 간단화하여 feedback 함수를 구할 수 있다.

다음과 같이 주어진 초기화 패턴(T_0)과 테스트 패턴(T_1)을 생성하는 NLFSR을 설계하기 위하여 위에서 제시된 설계절차를 적용하여 본다.

초기화 패턴 T0				테스트 패턴 T1			
A1	B1	A2	B2	A1	B1	A2	B2
1	0	1	1	0	0	1	1
1	1	0	1	1	1	0	0
0	1	1	0	1	1	1	0
0	1	1	1	0	0	0	1
1	1	0	0	0	1	0	0
1	1	1	0	1	0	1	0
0	0	1	0	1	1	0	1
0	0	0	0	1	1	1	1

우선 초기화 패턴(T_0)과 테스트 패턴(T_1)들을 단일 상태 벡터들로 패턴 결합 한다. 그림 6은 패턴 결합 과정을 나타낸 것이다.

초기화 패턴(T_0)	테스트 패턴(T_1)	단일 상태 벡터(T_m)
1 0 1 1	0 0 1 1	0 1 0 0 1 1 1 1
1 1 0 1	1 1 0 0	1 1 1 1 0 0 0 1
0 1 1 0	1 1 1 0	1 0 1 1 1 1 0 0
0 1 1 1	0 0 0 1	0 0 0 1 0 1 1 1
1 1 0 0	0 1 0 0	0 1 1 1 0 0 0 0
1 1 1 0	1 0 1 0	1 1 0 1 1 1 0 0
0 0 1 0	1 1 0 1	1 0 1 0 0 1 1 0
0 0 0 0	1 1 1 1	1 0 1 0 1 0 1 0

그림 6. 패턴 결합에 의한 상태 벡터 생성
Fig. 6. State vector generation by pattern merging.

다음은 T_m의 상태 벡터들을 쉬프트 레지스터의 상태씨이퀀스의 일부분이 되도록 재배열된 상태 벡터 집합 T를 구한다. 그림 7에서 집합 T는 새롭게 재배열된 상태 벡터들이다. 실제 T_m에만 존재하는 상태 벡터들

단일 상태 벡터(T_m)	재배열된 상태 벡터(T)
0 1 0 0 1 1 1 1	0 1 0 0 1 1 1 1
1 1 1 1 0 0 0 1	* 0 0 1 0 0 1 1 1
1 0 1 1 1 1 0 0	* 0 0 0 1 0 0 1 1
0 0 0 1 0 1 1 1	* 1 0 0 0 1 0 0 1
0 1 1 1 0 0 0 0	* 1 1 0 0 0 1 0 0
1 1 0 1 1 1 0 0	* 1 1 1 0 0 0 1 0
1 0 1 0 0 1 1 0	1 1 1 1 0 0 0 1
1 0 1 0 1 0 1 0	* 0 1 1 1 1 0 0 0
	1 0 1 1 1 1 0 0
	* 0 1 0 1 1 1 1 0
	* 0 0 1 0 1 1 1 1
	0 0 0 1 0 1 1 1
	* 0 0 0 0 1 0 1 1
	* 1 0 0 0 0 1 0 1
	* 1 1 0 0 0 0 1 0
	* 1 1 1 0 0 0 0 1
	0 1 1 1 0 0 0 0
	* 1 0 1 1 1 0 0 0
	1 1 0 1 1 1 0 0
	* 0 1 1 0 1 1 1 0
	* 0 0 1 1 0 1 1 1
	* 1 0 0 1 1 0 1 1
	* 0 1 0 0 1 1 0 1
	1 0 1 0 0 1 1 0
	* 0 1 0 1 0 0 1 1
	* 1 0 1 0 1 0 0 1
	* 0 1 0 1 0 1 0 0
	1 0 1 0 1 0 1 0

그림 7. 재배열된 상태 벡터
Fig. 7. Reordered state vectors.

만으로 연속적인 씨이퀀스를 만들 경우 feedback함수가 복잡해지므로, 상태 벡터를 연결하는 연결 벡터를 삽입하여 쉬프트 동작과 간단한 feedback함수블럭으로서 필요한 상태 씨이퀀스가 이루어질 수 있도록 한다. 그림 7에서 *로 표시된 벡터들은 T_m에 존재하지 않는 연결 벡터들이다. 연결 벡터는 두 상태 벡터를 쉬프트시키므로써 얻을 수 있다.

입력 패턴의 비트수가 4이므로 NLFSR에서 필요한 쉬프트 레지스터의 길이는 8이 된다. feedback 함수는 현재 상태 벡터의 각 레지스터 값들로 부터 다음 상태 벡터의 첫번째 레지스터 값을 결정해야하므로 다음과 같은 논리함수값들을 얻을 수 있다.

f(0,1,0,0,1,1,1,1) = 0
f(0,0,1,0,0,1,1,1) = 0
f(0,0,0,1,0,0,1,1) = 1
f(1,0,0,0,1,0,0,1) = 1
f(1,1,0,0,0,1,0,0) = 1
f(1,1,1,0,0,0,1,0) = 1
f(1,1,1,1,0,0,0,1) = 0
f(0,1,1,1,1,0,0,0) = 1
f(1,0,1,1,1,1,0,0) = 0
f(0,1,0,1,1,1,1,0) = 0
f(0,0,1,0,1,1,1,1) = 0
f(0,0,0,1,0,1,1,1) = 0
f(0,0,0,0,1,0,1,1) = 1
f(1,0,0,0,0,1,0,1) = 1
f(1,1,0,0,0,0,1,0) = 1
f(1,1,1,0,0,0,0,1) = 0
f(0,1,1,1,1,0,0,0) = 1
f(1,0,1,1,1,1,0,0) = 0
f(0,1,0,1,1,1,1,0) = 0
f(0,0,1,0,1,1,1,1) = 1
f(1,0,0,1,1,0,1,1) = 0
f(0,1,0,0,1,1,0,1) = 1
f(1,0,1,0,0,1,1,0) = 0
f(0,1,0,1,0,0,1,1) = 1
f(1,0,1,0,1,0,0,1) = 0
f(0,1,0,1,0,1,0,0) = 1

위의 함수 출력중에서 1값을 갖는 입력(minterm)만을 선택하여, 이로부터 논리를 간단화하면 (논리최소화 CAD tool인 PLAMIN[11]을 사용), feedback함수 f는 다음과 같이 된다.

$$F(A,B,C,D,E,F,G,H) = \overline{A}\overline{B}\overline{C}\overline{D}\overline{E}\overline{F}\overline{G}H + \cdots + ABCDEFGH$$
$$= \overline{A}F + \overline{A}CD + \overline{F}H + FGH + \overline{C}DF + EFGH$$

feedback 함수 f로부터 NLFSR을 구성하면, TPG는 그림 8과 같다.

위와 같이 설계된 TPG에 그림 7의 T집합의 시작 상태 벡터인 01001111을 초기화한 후 클럭을 인가하게

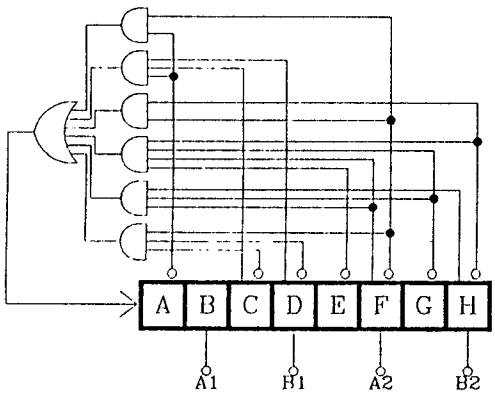

그림 8. 테스트 패턴 생성 회로
Fig. 8. An test pattern generator.

되면, T집합내에서 배열된 순서대로 상태 벡터가 발생하게 된다. 따라서 패턴 결합에 의해서 각 상태 벡터에 포함되어 있던 s-op 고장에 대한 초기화 패턴과 테스트 패턴이 TPG의 출력선을 통하여 CUT에 연속적으로 인가된다.

그림 9는 BILBO(Built-In Logic Block Observer)[10]내에 feedback 함수 블럭과 선택 회로를 부가하여 TPG시에 NLFSR로 동작하도록 구성된 회로로서, 두 제어신호선 (C_1과 C_2)에 의해 TPG와 Signature Analyzer 등의 여러 기능을 수행하게 된다. 따라서 그림 9의 회로를 사용하게 되면 CMOS 회로에 대한 Built-in self-test가 가능하다.

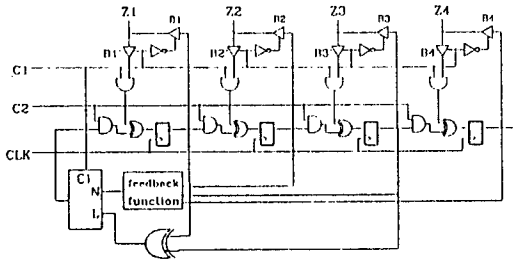

그림 9. 변경된 BILBO 회로
Fig. 9. A modified BILBO.

MUX로 구성된 feedback 선택회로는 제어신호 C_1에 의해서 feedback이 선택되는데, C_1이 0이면 테스트 패턴 생성을 위하여 NLFSR이 선택되고, C_1이 1이면 테스트 결과 신호의 분석을 위하여 LFSR이 선택된다. TPG로 동작할 경우의 출력선과 Signature Analyzer로 동작할 경우의 입력선을 같이 사용하기 위하여 tri-state 버퍼를 사용한다. tri-state 버퍼의 동작은 C_1=0, C_2=1 일 때 버퍼 B1, B2, B3, B4는 enable되고, B1', B2', B3', B4'는 disable된다. 따라서 NLFR에서 생성된 테스트 패턴은 Z1, Z2, Z3, Z4를 통해서 CUT에 인가된다. 한편 C_1=1,C_2=1이면 버퍼 B1, B2, B3, B4는 disable되고, B1', B2', B3', B4'는 enable되어 LFSR은 Z1, Z2, Z3, Z4를 통해 CUT로부터 응답신호를 받아 들여 Signature Analyzer로 동작한다.

표 2는 제어신호에 따른 회로의 기능을 나타낸 것이다.

표 2. 동작 모드
Table 2. Operation mode.

C_1	C_2	동 작 모 드	기 능
0	0	입출력 모드	Reset
0	1	출 력 모 드	테스트패턴 생성(NLFSR)
1	0	입 력 모 드	정 상 동 작
1	1	입 력 모 드	신 호 분 석

V. 결 론

본 논문에서는 CMOS 회로에 적합한 Built-in self-test 방법을 제안하였다. CMOS회로의 stuck-open 고장은 그 특성상 2개의 연속적인 테스트 패턴이 인가되어야 검출되므로, 이에 적합한 TPG를 설계할 수 있도록 2개의 패턴을 결합하고 재배열하는 알고리듬을 제안하였다. 제안된 알고리듬에 따라 설계된 Built-in self-test 회로는 제어 신호에 따라 테스트 패턴 생성시에는 NLFSR로 동작하고, 신호분석시에는 LFSR로 동작하게 된다. NLFSR에 의해 생성되는 테스트 패턴은 pseudo-exhaustive하게 생성되는 것이 아니라 필요한 패턴만이 지정된 순서대로 생성되므로, CMOS 회로 테스트가 효과적으로 수행된다.

參 考 文 獻

[1] P. Goel, "Test generation costs analysis and projections," *17th Design Automation Conf.*, pp. 77-84, June 1980.

[2] T. W. Williams and K. P. Parker, "Design for testability-a survey," *IEEE Trans. Comput.*, vol. C-31, no. 1, pp. 2-15, Jan.

1982.

[3] Donald Komonytsky, ""LSI self-test using level sensitive scan design and signature analysis," *IEEE Test Conf.*,pp. 414－424, 1982

[4] J. Fiebiger, "CMOS : a designer's dream with best yet come", *Electronics*, pp. 113－115, Apr. 1984.

[5] D. J. Myers and P. A. Ivey, "A design style for VLSI CMOS", *IEEE Journal of Solid State Circuits*, vol. Sc-20, no. 3, pp. 741－745, June 1985.

[6] R. D. Davies, "The case for CMOS", *IEEE Spectrum*, pp. 26－32, Oct. 1983.

[7] R. L. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits", *Bell System Tech. Journal* vol. 57, no. 4, pp. 1449－1474, May-June 1978.

[8] Chandramouli. R., "On testing stuck-open faults", *The 1983 Int. Symp. Fault Tolerant Comput.* pp. 28－30, Milano, Italy, June. 1983.

[9] P. H. Bardell and W. H. Mcanney, "Parallel pseudorandom sequences for built-in test", *IEEE Test Conf.*, pp. 302－308, 1984.

[10] Brend Konemann, et al., "Built-in logic block observation techniques", *IEEE Test Conf.*, pp. 37－41, Oct. 1979.

[11] 이재민, 임인칠, "PLA의 논리최소화를 위한 휴리스틱 알고리듬－PLA 논리최소화 프로그램 PLAMIN－", 대한전자공학회논문지, pp. 63－68, 1986년 5월.

著 者 紹 介

金 倫 弘 (正會員) 第27卷 第5號 參照
현재 한양대학교 전자공학과 박사과정 졸업

林 寅 七 (正會員) 第25卷 第8號 參照
현재 한양대학교 전자공학과 교수