특 집

내장된 자체검사 기법(BIST) 의 기술동향

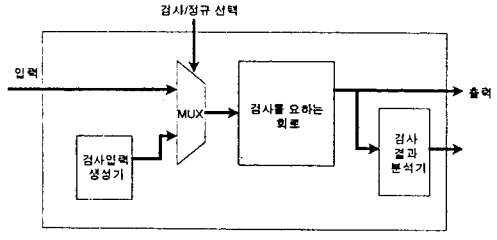
姜 成 昊

延世大學校 電氣工學科

I. 서 론

VLSI 회로 설계기술의 발달로 회로의 집적도가 증가됨에 따라 회로에 대한 검사는 점점 더 어려워지고 있으며 검사에 소요되는 비용도 커지고 있다. 따라서 효과적인 검사의 수행은 검사비용의 절감뿐만 아니라 전체회로의 개발시간을 절약하여 양질의 회로를 만들게 하므로 점점 중요하게 여겨지고 있다. 이에 따라 회로 설계 시 검사를 고려하여 설계하는 검사 용이화 설계(design for testability) 기법이 많이 사용되어지고 있다. 검사 용이화 기법이 사용된 회로는 원래 회로와 비교하여 검사 용이도가 높아지고 검사입력의 생성(test pattern generation) 이 쉬워져서 전체 검사 시간이 짧아지게 되며 양질의 검사를 수행하게 된다. 하지만 입출력 단자의 증가, 전체 회로 면적의 증가, 전력소모의 증가, 지연시간의 증가 등의 오버헤드를 갖게 된다. 따라서 설계하는 회로에 따라 여러 비용을 고려하여 적당한 검사 용이화 기법이 사용되어야 한다. 여기서는 여러 검사 용이화 설계기법 중 내장된 자체검사 기법(BIST : built-in self test)[1, 2, 3]에 대해 살펴보고자 한다.

내장된 자체 검사기법은 회로의 한 부분이 회로 자체를 검사하는데 쓰이는 것이다. 이의 특징은 검사입력이 자체 내에서 생성이 되고 검사의 결과도 자체 내에서 평가되어지는 것이다. 따라서 외부에서 필요한 동작은 검사를 시작시킨 후 검사에 성공했는지 실패했는지의 결과만을 확인하면 된다. 일반적인 내장된 자체 검사기법의 구조는 그림 1과 같다.



〈그림 1〉 내장된 자체검사의 구조

정규 동작 시에는 입력과 출력 선을 따라 회로가 정상동작을 하고 검사동작 시에는 검사입력 생성기에서 생성된 검사입력으로 회로를 검사한 후 결과를 검사입력 분석기에서 분석하여 회로의 검사 결함 유무를 나타내게 된다. 따라서 이 기법의 강점은 비싼 검사기(tester) 를 필요로 하지 않으며 정규동작 속도로 회로를 검사할 수 있다는 것이다. 이 방법을 사용하기 위해서는 면적을 많이 차지하게 되고 또 어떤 회로는 임의 입력(random pattern) 에 잘 검사되지 않는 회로가 있으며 검사 결과의 분석 시 결함이 있는 회로를 결함이 없다고 여기는 매스킹(masking) 문제가 생길 수 있다는 점 등을 잘 고려해야 한다.

II. 검사입력 생성

내장된 자체검사 기법을 이용하여 회로를 검사하기 위해서는 검사입력을 자체 생성해 내어야 한다.

한가지 방법은 회로내의 프로세서와 ROM이 있는 경우 ROM에 검사프로그램을 저장하여 검사 시 검사입력을 생성하도록 하는 것이다[2]. 이 때의 검사입력 기능적 검사(functional test) 를 생성하게 된다. 이의 단점은 ROM의 상당 부분을 차지하게 되며 검사입력 대신 생성 비용을 줄일 수 없다는 단점이 있다. 전수 검사입력을 사용하는 경우는 n입력을 갖는 조합 회로에 대해 2^n 검사입력이 필요하게 된다. 이는 알고리즘을 이용한 검사입력 생성과정이 필요없이 이진 계수기 등을 이용하여 생성할 수 있다. 또한 모든 검출 가능한 고장을 검출할 수가 있으므로 높은 검사적용범위(fault coverage) 를 갖게 된다. 그러나 이 방법은 n이 큰 경우에 너무 많은 검사입력 때문에 검사 시간이 길어지게 되어 사용할 수가 없다. 또한 순차화로에도 적합하지 않다. 한편 완전 LFSR(complete linear feedback shift register)[4]를 사용하여 LFSR에 모든 레지스터가 0인 상태를 포함하여 검사입력을 생성할 수 있다.

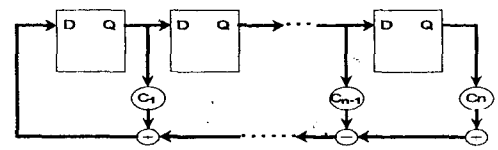
임의 검사입력을 사용하는 경우 같은 검사입력이 반복될 수 있으므로 효율적인 검사에는 맞지 않다. 따라서 모든 검사입력이 같은 확률로 생성되며 반복되지 않도록 하는 유사임의(pseudo random) 검사입력을 사용하게 된다. 이는 LFSR(linear feedback shift register) 를 사용하여 생성될 수 있다. LFSR은 플립플롭과 XOR 게이트로 구성되며 모든 유사임의 검사입력을 생성할 수 있다. LFSR은 XOR의 위치에 따라 2가지로 나타나는데 이는 그림 2와 같다. 여기서 Ci는 이진상수로 Ci =0은 끊어짐 상태를 1은 연결된 상태를 나타낸다. LFSR은 입력순서에 따라 출력순서와 출력부호를 결정하는 특성다항식(characteristic polynomial) 을 갖는다. 그림 2-(a)에서 보면

$$P(x)=1+C_1x+C_2x^2+\cdots+C_nx^n$$

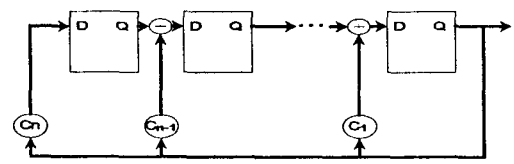
으로 정의되며 생성함수(generating function) 는

$$G(x)=a_0+a_1x+\cdots+a_nx^n$$

으로 나타내어진다.



(a) 외부 XOR를 이용한 LFSR



(b) 내부 XOR를 이용한 LFSR

〈그림 2〉 LFSR의 구조

이때 수열 $a_0\cdots a_m$은 LFSR에 의해 생성되는 출력 순서를 나타낸다.

$$G(x)=\sum_{i=1}^{n}a_mx^m$$

LFSR에서 현재상태가 $a_{m-i}(i=1,\cdots,n)$라면

$$a_m=\sum_{i=1}^{n}C_i a_{m-i}$$

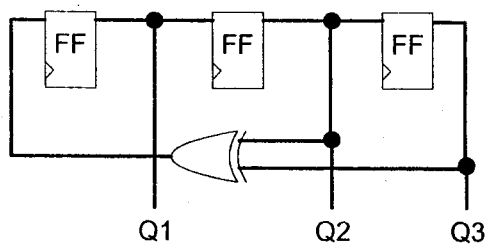
따라서 회로의 동작은

$$\begin{aligned}G(x)&=\sum_{m=0}^{\infty}a_m x^m\\&=\sum_{m=0}^{\infty}\sum_{i=1}^{n}C_i a_{m-i}x^m\\&=\sum_{i=1}^{n}C_i x^i\sum_{m=0}^{\infty}a_{m-i}x^{m-i}\\&=\sum_{i=1}^{n}C_i x^i[a_{-i}x^{-i}\\&\qquad+\cdots+a_{-1}x^{-1}+\sum_{m=0}^{n}a_m x^m]\\&=\sum_{i=1}^{n}C_i x^i[a_{-i}x^{-i}+\cdots+a_{-1}x^{-1}\\&\qquad+G(x)]\end{aligned}$$

따라서

$$G(x)=\frac{(a_{-i}x^{-i}+\cdots+a_{-1}x^{-1})}{1+\sum_{i=1}^{n}C_i x^i}$$

따라서 $G(x)$는 LFSR 초기상태 $a_{-1}\cdots a_{-n}$ 과 되먹임 계수 $C_1\cdots C_n$의 함수이다. n개의 레지스터가 있는 경우 $C_m=1$로 하고 $a_{-1}\cdots a_{1-n}=0$, $a_{-n}=1$이면 $G(x)=\frac{1}{P(x)}$로 된다. 즉 초기상태에 따른 특성다항식은 LFSR의 주기성을 나타내고 출력순서를 결정한다. LFSR은 주기성을 갖게 되는데 n개의 레지스터가 사용될 경우 최대 $2n-1$의 주기를 갖는다. 만일 LFSR의 초기상태가 $a_{-1}\cdots a_{1-n}=0$, $a_{-n}=1$이면 최대길이를 갖는 특성다항식을 근본다항식(primitive polynomial)이라 하는데 이를 위해서는 자기자신과 1이외에는 나누어지지 않는 형태를 가지고 있어야 한다. 예를 그림 3에 나타내었다. 그림 3에서 만일 초기상태가 000이면 이 상태에서 값이 바뀌지 않는다. 하지만 초기상태가 100이면 다음과 같은 2^n-1개의 유사임의 검사입력이 만들어진다.



〈그림 3〉 3 비트 LFSR

〈표 1〉 LFSR의 예

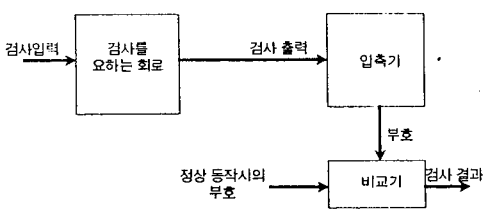
	Q1	Q2	Q3
초기상태	1	0	0
	0	1	0
	1	0	1
	1	1	0
	1	1	1
	0	1	1
	0	0	1
	1	0	0
	.	.	.
	.	.	.
	.	.	.

어떤 회로들은 임의 검사입력으로 잘 검출되지 않는 고장들을 가지고 있다. 이들을 효과적으로 검사하기 위해서는 많은 검사입력을 필요로 한다. 임의 검사입력을 잘 이용하였을 때 LFSR을 이용하여 검사입력 생성 시 입력 비트마다 0과 1이 같은 확률로 만들어진다. 어떤 회로들은 높은 검사적용범위를 갖기 위해서 0과 1의 분포가 한쪽으로 치우쳐야 한다. 검사입력의 분포에 따라 검사적용범위를 예측하는 연구[5, 6]들이 수행되어졌다. 따라서 0과 1의 분포가 균일하지 않도록 검사입력을 생성하는 가중입력 생성에 대해 연구[7, 8]가 진행되었다. 이는 가중치를 결정한 후 기존의 LFSR에 조합회로를 첨가하여 쉽게 구할 수 있다. 유사전수 검사입력을 이용하는 방법은 입력수가 큰 경우 전

수 검사입력 사용이 불가능한 단점을 개선하고자 하는 것이다. 이는 회로를 분할하여 전수 검사입력보다 작은 수의 검사입력을 사용하는 것이다. 분할하는 방법에 따라 크게 콘(cone) 분할과 기능블럭분할의 2가지로 구분된다[9]. 콘분할 방법은 각 출력에 관여하는 입력들로 콘을 구성하도록 분할하여 각 출력마다의 전수 검사입력을 만들어 사용하는 것이다. 예를 들면 24입력 6출력의 74LS630 회로의 경우 각 출력 당 10개의 입력을 가지므로 2^{24}대신 6×2^{10}의 검사입력이 필요하게 된다. 하지만 만일 어느 한 출력이 모든 입력에 관여한다면 콘분할 방법으로서 득이 없고 기능블럭분할을 고려하여야 한다. 기능적 분할은 회로 내에서 기능블럭 단위로 분할하여 각 블럭 당 전수 검사입력을 구하여 사용하는 방법이다. 14입력 8출력의 74LS181의 경우 콘분할 방법 사용 시 어떤 출력이 모든 14 입력과 관계하여 득이 없다. 이를 기능블럭으로 분할하면 356개의 검사입력이 필요하게 된다.

III. 검사 결과 분석기

회로를 검사하는 과정은 정확하게 동작할 때 기대되는 회로의 출력과 검사입력을 가했을 때의 출력을 비교하는 것이다. 내장된 자체검사를 수행하는 경우에 정상동작시의 출력값을 ROM 등에 저장해 두어야 하는데 많은 수의 입력이 생성되는 경우에 많은 기억용량을 필요로 한다. 이 오버헤드가 상당히 크기 때문에 검사결과를 압축해서 부호(signature) 로 나타내고 이를 비교하는 방법을 사용하게 된다. 이의 개념은 그림 4와 같다.



〈그림 4〉 압축을 이용한 검사결과 분석

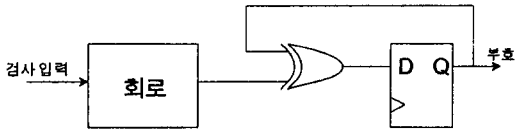
이와 같은 압축을 사용했을 때 오버헤드를 줄이는 장점이 있지만 고장 시 부호가 정상동작의 부호와 같아져서 고장이 있는 회로를 고장이 없다고 여길 수 있다. 이와 같은 현상을 매스킹 또는 에일리어싱(aliasing) 이라고 하는데 이것이 생기는 확률을 예측하고 이를 줄이도록 하는 것이 중요하다. 이 압축방법은 구현하기가 쉽고 높은 검사적용범위를 나타내므로 많이 사용된다. 여러 가지 압축방법 중 가장 간단한 방법은 검사 시 출력에 나타나는 1의 개수를 부호로 이용하는 방법이다. n개의 검사입력을 가한다면 부호는 0과 n사이의 값이 되고 따라서 이를 위한 계수기가 필요하게 된다. 만일 정상동작 시 1의 수가 p라면 이는 p개의 1과 $n-p$개의 0으로 출력이 나타나게 되는데 이 중 순서가 뒤바뀌는 경우 고장이 있지만 매스킹되게 된다. 따라서 정확한 p와 매스킹되는 시이퀀스는 ${}_nC_p-1$이다. 따라서 매스킹의 확률은 이를 전체 시이퀀스의 수로 나눈 것으로 다음과 같이 표시된다.

$$P_{masking}=\frac{{}_nC_p-1}{2^n-1}$$

신드롬(syndrome) 검사[10, 11]는 n개의 입력을 가진 조합회로에 2^n개의 전수입력을 가해서 그때의 1의 개수를 부호로 사용하게 된다. 따라서 신드롬 또는 부호는 k가 출력단에 나타나는 1의 개수일 때 $\frac{k}{2^n}$으로 주어진다. 이는 단순히 1의 개수를 세는 방법에 비해 좀더 일반적이고 신드롬 검사 용이도의 개념을 사용하였다. 전이계수(transition count)법은 출력순서에서 0에서 1 또는 1에서 0으로의 전이의 수를 사용하여 부호로 사용하는 것이다. 정상회로에서 전이계수의 부호가 p라면 n개의 검사입력을 가했을 때 부호는 최대 $m-1$까지 가능하고 이 중에서 순서가 바뀌는 경우는 ${}_{n-1}C_p$이고 0과 1이 바뀌어도 마찬가지이므로 모든 가능한 순서는 ${}_{2n-1}C_p$로 된다. 따라서 매스킹의 확률은 아래와 같다.

$$P_{masking}=\frac{{}_{2m-1}C_p-1}{2^n-1}$$

이를 위해서는 전이 감지기와 계수기가 필요하다. 다른 방법으로는 패리티 측정[12, 13]이 있다. m개의 출력순서를 홀수나 짝수의 패리티로 1비트로 압축하여 부호로 사용하는 것이다. 이는 XOR 게이트와 래치를 이용하여 $x+1$의 형태로 주어지는 LFSR을 그림 5처럼 구성하면 된다. 이는 모든 단일 고장이나 홀수개의 단일고장을 검출할 수 있다.

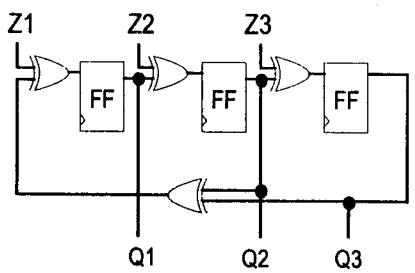


〈그림 5〉 패리티 측정기

매스킹이 생기는 확률은

$$P_{masking}=\frac{2^{m-1}-1}{2^m-1}$$

가장 널리 사용되는 방법인 부호분석기(signature analyzer)[14, 15]를 이용하는 방법은 LFSR을 이용하여 검사결과를 압축하는 것이다. 회로의 출력을 나눗셈의 입력으로 생각하고 LFSR을 나눗셈 회로로 사용하여 부호는 나눗셈의 몫으로 생각하여 출력을 압축시키게 된다. 예를 들면 그림 6에 부호분석기를 나타내었다.



〈그림 6〉 3 bit 부호분석기

만일 초기상태가 000이라 하자. 이에 대한 회로의 동작을 표 2에 나타내었다.

〈표 2〉 3 비트 부호분석기의 동작정상회로

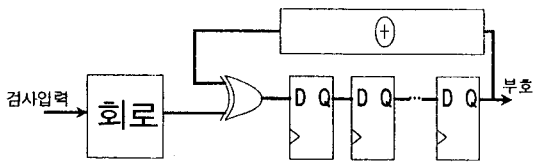
	정상회로	고장회로
clock	$Z_1Z_2Z_3\ Q_1Q_2Q_3$	$Z_1Z_2Z_3\ Q_1Q_2Q_3$
1	110110	110110
2	010101	011011

$Q_1Q_2Q_3$에 남아있는 값이 부호에 해당되며 정상회로의 부호와 고장회로를 비교해서 고장이 있음을 알 수 있다. 하지만 매스킹이 생기는 경우가 있는데 이는 표 3에 나타내었다.

〈표 3〉 매스킹이 생기는 예

	정상회로	고장회로
clock	$Z_1Z_2Z_3\ Q_1Q_2Q_3$	$Z_1Z_2Z_3\ Q_1Q_2Q_3$
1	110110	100011
2	010101	011101

이 경우 실제로 고장이 있지만 부호가 같아서 검출되지 않게 된다.



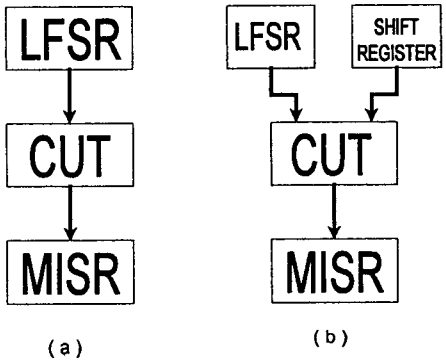
〈그림 7〉 MISR

n개의 레지스터가 사용되었을 때 매스킹이 생길 확률을 알아보자. m개의 검사입력이 사용된다면 $P_{masking}=\frac{2^{m-1}-1}{2^n-1}$로 주어지고 $m>>n$이면 $P_{masing}\frac{1}{2^n}$이 된다. 이 부호분석기는 여러 출력을 갖는 회로에 사용될 수 있다. 모든 출력마다 부호분석기를 붙이는 것은 오버헤드가 너무 크기 때문에 사용이 힘들다. 따라서 플립플롭 사이에 XOR를 첨가하고 이의 한 입력은 회로의 출력이, 다른 입력은 전단의 플립플롭의 출력이 입력되도록 하여 구성한다. 이 경우도 마찬가지로 매스킹 확률은 $\frac{1}{2^m}$이 된다. 예를 그림 7에 나타내었다. 이를 MISR(multiple input signature register)[16, 17]라

부른다. 매스킹의 확률을 줄이려면 LFSR에 사용되는 플립플롭의 수를 늘려도 되고 또는 LFSR의 특성다항식을 바꾸어 같은 검사입력을 가해도 된다.

IV. BIST 구조

그림 8에 나타낸 클럭 당 검사(test-per clock) 구조는 검사입력을 가하고 각 클럭주기 동안 결과를 얻는 구조로 되어있다.



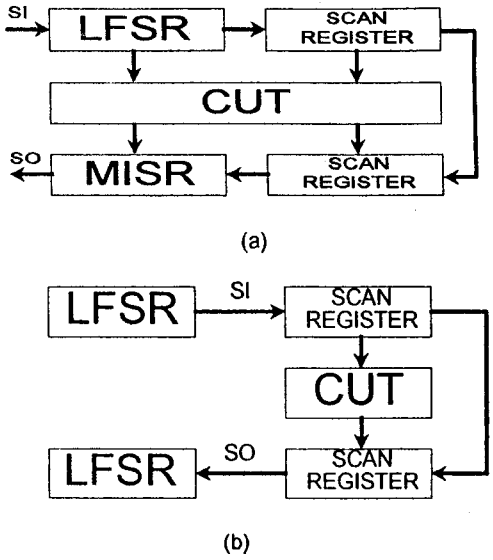
〈그림 8〉 클럭 당 검사

그림 8-(a)는 LFSR과 MISR을 사용한 간단한 구조로 전수검사나 유사임의 검사에 적당하며 그림 8-(b)는 유사전수 검사나 유사임의 검사에 적당하다.

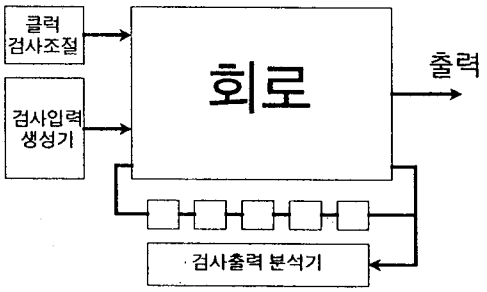
그림 9에 나타낸 스캔 당 검사(test per scan) 구조는 검사를 가하여 각 스캔 사이클 동안 결과를 얻는 것이다.

그림 9-(a)에 나타낸 경우 하드웨어를 줄이기 위해 MISR의 일부를 스캔 레지스터로 사용하여서 검사시간이 느리게 된다. 그림 9-(b)의 경우는 입력 하나의 스캔 체인으로 연결되어지고 LFSR로 결과를 얻는다.

BIST의 구조는 분리(separate) 와 내장(embedded) 으로 구분할 수 있다. 그림 10의 분리형 BIST의 경우는 검사입력 생성기와 검사출력 분석기는 외부에 있고 내부 스캔을 이용하는 것이다.



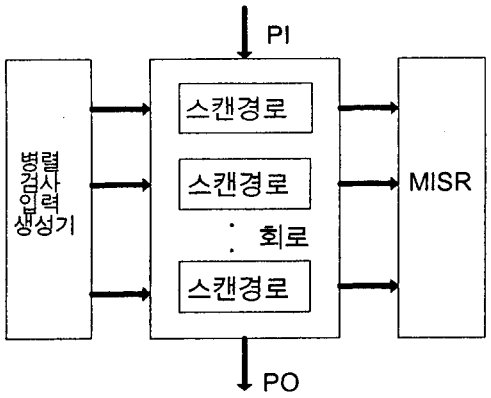
〈그림 9〉 스캔당 검사



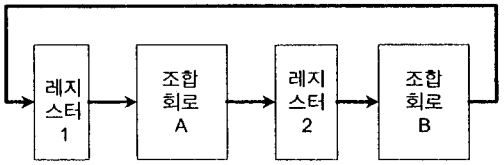
〈그림 10〉 분리형 BIST

이는 오버헤드가 적고 쉽게 제어할 수 있는 장점이 있는 반면 스캔을 이용하기 때문에 검사시간이 길어지게 된다. 분리형으로 구성되어 MISR와 병렬을 이용한 회로로 STUMP가 있다. STUMP[18, 19]의 구조는 그림 11과 같다.

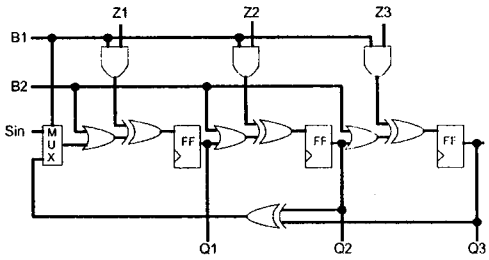
병렬 임의검사입력 생성기로부터 스캔 경로를 이용하여 검사를 입력하고 MISR을 이용하여 부호를 생성한다. 여러 스캔 경로를 이용하여 검사시간을 절약할 수 있다. 보드 상에서 검사 시에는 병렬 검사입력기와 MISR을 검사 칩으로 만들어서 사용할 수 있다. 그림 12의 내장된 BIST의 경우는 회로를 조합회로 블럭으로 분할하여 사용하는 것이다.



〈그림 11〉 STUMP의 구조



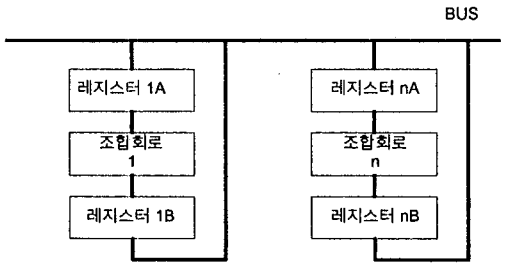
〈그림 12〉 내장된 BIST



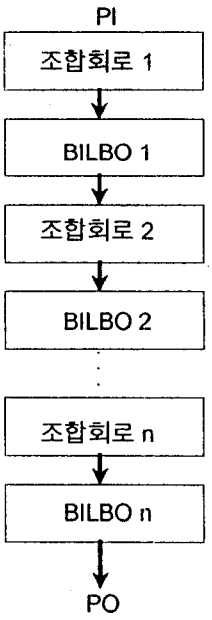
〈그림 13〉 BILBO의 구조

검사를 요하는 회로가 그림과 같이 분할되었으면 회로 A 검사 시 레지스터 1을 검사입력 생성기로 레지스터 2를 검사출력 분석기로 사용하고 조합회로 B를 검사 시 레지스터 2를 검사입력 생성기로 레지스터 1을 검사출력 분석기로 여기는 것이다.

내장된 BIST의 예로는 그림 13의 BILBO (Built-In Logic Block Observer)[20,21]를 들 수 있다. 이는 B1 B2의 제어신호에 따라 정상동작을 하거나 또는 부호 분석기 또는 검사입력 생성기로 동작하게 된다. 이를 표 4에 나타내었다. 회로내의 레지스터들이 많은 경우 이를 몇 개로 분할하여 BILBO 레지스터를 구성하고 이들 간의 조합회로에 대한 검사를 수행한다. 이는 그림 14처럼 BUS를 이용하거나 그림 15의 파이프라인(pipeline) 의 형태로도 구성되어질 수 있다.



〈그림 14〉 버스를 이용한 BILBO



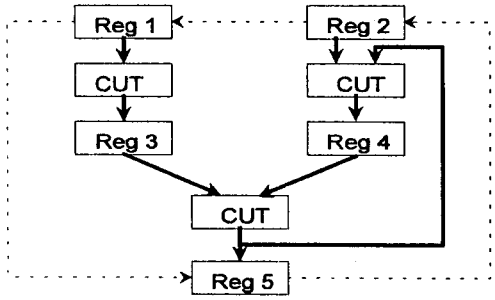
〈그림 15〉 파리프라인을 이용한 BILBO

CBIST (Circular BIST)[22,23,24]는 기존의 BIST의 비용을 줄이기 위해 레지스터를 BIST레지스터로 바꾸고 이를 연결하여 원형의 자체검사 경로를 만든 것이다. 이는 검사제어가 기존의 BIST보다 쉽고 BILBO보다 적은 수의 동작모드가 있으며 검사가 한번에 끝날 수 있다. 또한 자동합성이 쉽게 된다는 장점이 있지만 검사입력이 진정한 의미의

유사임의 형태가 아니며 매스킹의 확률이 커질 수 있다.

〈표 4〉 BILBO의 동작

동작제어신호		
B1	B2	동작
1	1	정규동작
0	1	리셋
0	0	이동 레지스터
1	0	부호분석기



〈그림 16〉 CBIST

그림에서 점선은 자체검사 경로를 나타낸다.

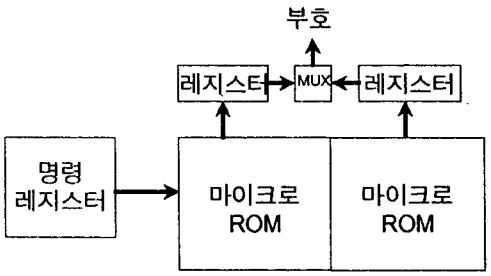
이외에 PLA[25, 26], ROM[27, 28], RAM[29, 30, 31, 32] 등에도 BIST가 많이 사용되고 있으며 기존의 고착고장 이외에 다른 여러 고장을 고려하여 검사입력을 생성하게 된다.

V. BIST 응용

검사의 중요성에 따라서 많은 회로들이 스캔, 경계스캔, BIST등의 D FT기법을 많이 사용하고 있다. 실제로 거의 모든 마이크로프로세서에서나 컨트롤러에는 BIST가 사용되고 있다[33, 34]. 몇 가지 예를 알아보자.

Intel 486 cpu의 경우 386의 BIST 기법을 확장시켰다[35, 36]. 모든 PLA은 LFSR을 이용하여 전수검사입력을 생성하고 이를 압축하였다. 마이크로코드 ROM을 자체검사입력으로 모든 워드를 검사하고 LFSR로 결과를 압축한다. 명령어 캐쉬(instruction cache) 와 데이터 캐쉬, TLB도 마이크로코드 검사를 사용하였다. BIST로 386의 경우 약 52%의 트랜지스터를 검사하게 하였다. 486의 경우 약 68%의 트랜지스터를 검사하게 하였다.

MC68881[37]의 경우에는 그림 17에 나타낸 형태로 마이크로ROM을 검사한다.



〈그림 17〉 MC68881의 마이크로ROM BIST

마이크로ROM은 2개로 분할되어서 검사되는데 명령 레지스터로 주소를 결정하고 2개의 16 비트 부호 분석기를 이용한다. $x^{15}+x^{12}+x^5+1$의 다항식을 사용한다.

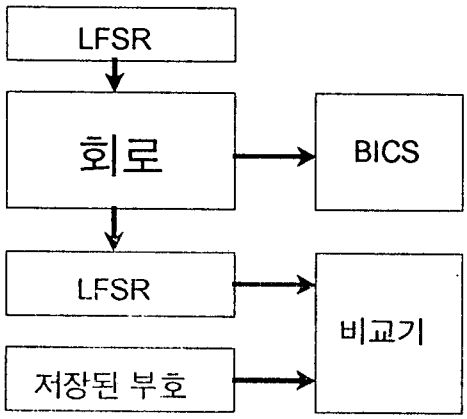
IBM ES/9121 processor[38]의 경우 STUMP 방법을 이용하였고 이의 오버헤드는 약 3-5% 정도를 차지했다. 검사입력 생성기로 $1+x+x^3+x^4+x^{64}$의 LFSR을 사용하였다.

SunSPARC[39]의 경우 2개의 동작을 갖는 BIST를 사용하였다. 17비트 LFSR로 검사입력기를 만들었고 31비트 검사출력 분석기를 사용하여서 간단한 검사 시는 임의 검사입력을 생성해서 각 스캔 체인에 보내고 복잡한 검사 시는 64K 임의 검사입력을 생성하였다. 이의 수행 시간은 복잡한 검사 시 40MHz로 27초 정도 걸렸다.

Motorola 68060[40]의 경우 스캔 BIST와 메모리 BIST를 사용한다. 12 bit 검사입력 생성기를 이용하여 7개의 스캔 체인을 병렬로 구동하고 결과를 압축한다. 메모리 BIST는 모든 내장된 캐쉬를 검사한다. BIST의 오버헤드는 전체 약 1.9%를 차지한다 (전체 DFT 오버헤드는 9.1%).

BIST구조를 자동으로 합성해주는 연구[41, 42, 43, 44]도 진행되고 있다. 주어진 회로의 HDL에서 다양한 BIST구조를 포함한 설계를 만들어 주는 연구이다.

또 BIST를 이용하여 Iddq 검사를 구현하는 방법[45, 46]도 연구되고 있다. 그림 18에 이의 구조를 나타내었다.



〈그림 18〉 Iddq BIST 구조

BICS는 내장된 전류센스로서 Iddq를 관찰하여 전류가 기준치이상 흐를 때 고장임을 나타내게 된다. 회로가 큰 경우 이를 분할하여 위의 방법을 사용한다.

한편 아날로그 회로에 대해 BIST를 이용하는 방법[47, 48, 49]도 활발히 연구되어지고 있다.

VI. 결 론

양질의 회로를 빠른 시간 내에 개발하려는 요구는 검사의 중요성을 부각시켰고 따라서 효율적인 검사를 설계초기부터 고려하기 시작하였다. BIST에 대한 연구는 매우 활발하게 진행되고 있으며 새로운 구조가 계속 개발되어지고 있다. 최근 수년간 회로설계에서는 회로의 높은 신뢰도의 요구에 따른 검사의 중요성과 더불어 BIST의 사용이 급증되고 있으며 이런 추세는 계속되어질 것이다. BIST 사용 시에 범용을 고려하여 회로에 맞게 사용되어져야 하므로 고부가가치 회로에서는 거의 사용되어지고 있다. BIST의 사용 시에 따르는 오버헤드를 줄이고 매스킹을 줄이는 연구가 계속되어지고 있으며 BIST의 응용은 더욱 광범위해질 것이다.

참 고 문 헌

[1] Thomas W. Williams and Kenneth P. Parker, "Design for Testability-A Survey", *Proceedings of IEEE*, Vol. 71, No. 1, pp. 98-112, January, 1983.

[2] Edward J. McCluskey, "Built-In Self-Test Techniques", *IEEE Design & Test*, pp. 21-28, April, 1985.

[3] Edward J. McCluskey, "Built-In Self-Test Structures", *IEEE Design & Test*, pp. 29-36, April, 1985.

[4] L. Wang and E. McCluskey, "Complete Feedback Shift Register Design for Built-In Self Test", *Proc. of Int'l Conf. on Computer Aided Design*, pp. 56-59, 1986.

[5] R. Linsanke. et al., "Testability Driven Random Test Pattern Generation", *IEEE Trans. on Computer Aided Design*, pp. 1082-1087, Nov., 1987.

[6] H. Wunderlich, "Multiple Distributions for Biased Random Test Patterns", *Proc. of Int'l Test Conference*, pp. 236-244, 1988.

[7] H. Schnurmann, E. Lindbloom, and R. Carpenter, "The Weighted Random Test Pattern Generator", *IEEE Trans. on Computers*, pp. 695-700, July, 1975.

[8] H. Wunderlich, "Self Test Using Unequiprabable Random Patterns", *Proc. of Int'l Symp. on fault Tolerant Computing*, pp. 258-263, 1987.

[9] E. McCluskey, "Verification Testing-A Pseudo exhaustive Test Technique", *IEEE Trans. on Computers*, pp. 541-546, June,

1984.

[10] J. Savir, "Syndrome-Testable Design of Combinational Circuits", *IEEE Trans. on Computers*, pp. 442–451, June, 1980.

[11] J. Savir, "Syndrome-Testing of Syndrome-Untestable Combinational Circuits", *IEEE Trans. on Computers*, pp. 606–608, August, 1981.

[12] A. Tzidon, I. Berger, and M. Yoeli, "A Practical Approach to fault Detection in Combinational Networks", *IEEE Trans. on Computers*, pp. 968–971, October, 1978.

[13] J. Carter, "Signature Testing with Guaranteed Bounds for fault Coverage", *Proc. of Int'l Test Conf.*, pp. 75–82, 1982.

[14] J. Smith, "Measures of the Effectiveness of Fault Signature Analysis", *IEEE Trans. on Computers*, pp. 510–514, June, 1980.

[15] W. McAnney and J. Savir, "Built-in Checking of the Correct Self-Test Signature", *IEEE Trans. on Computers*, pp. 1142–1145, September, 1988.

[16] S. Hassan, D. Lu, and E. McCluskey, "Parallel Signature Analyzers–Detection Capability and Extensions", *Proc. of COMPCON*, pp. 440–445, 1983.

[17] R. David, "Signature Analysis of Multiple Output Circuits", *Proc. of Int'l Symp. on Fault Tolerant Computing*, pp. 366–371, 1984.

[18] P. Bardell W. McAnney, "Self-Testing of Multichip Logic Modules", *Proc. of Int'l Test Conf.*, pp. 200–204, 1982.

[19] P. Bardell W. McAnney, "Parallel Pseudorandom Sequences for Built-In Test", *Proc. of Int'l Test Conf.*, pp. 302–308, 1984.

[20] B. Koremann, J. Mucha, G. Zwiehoff, "Built-In Logic Block Observation Technique", *Proc. of Int'l Test Conf.*, pp. 37–41, 1979.

[21] L. Wang, E. McCluskey, "Concurrent Built-In Logic Block Observer (BILBO)", *Proc. of Int'l Symp. on Circuilt and Systems*, pp. 1054–1057, 1986.

[22] J. Carletta and C. Papachristou, "Structural Constraints for Circular Self-Test Paths", *Proc. of Int'l VLSI Test Symp.*, pp. 87–92, 1994.

[23] R. Gage, "Structured CBIST in ASICs", *Proc. of Int'l Test Conf.*, pp. 332–338, 1993.

[24] A. Krasniewski and S. Pilaiski, "Circular Self-Test Path : A Low Cost, BIST Technique for VLSI Circuits", *IEEE Trans. on Computer Aided Design*, pp. 46–55, January, 1989.

[25] S. Hassan and E. McCluskey, "Testing PLAs Using Multiple Parallel Signature Analyzers", *Proc. of Int'l Symp. on Fault Tolerant Computing*, pp. 422–425, 1983.

[26] W. Daehn and J. Mucha, "A Hardware Approach to Self Testing of Large Programmable Logic Arrays", *IEEE Trans. on Computers*, pp. 829–833, Nov., 1981.

[27] Y. Zorian and A. Ivanov, "EEODM : An Effective BIST scheme for ROMs", *Proc. of Int'l Test Conf.*, pp. 871–879, 1990.

[28] P. Nagvajara and M. Karpovsky, "Built-in Self diagnostic Read Only Memories", *Proc. of Int'l Test Conf.*, pp. 695–703, 1991.

[29] S. Jain and C. Stroud, "Built-In Self Testing of Embedded Memories", *IEEE Design and Test*, pp. 27–37, Octover, 1986.

[30] M. Nicolaidis, "Transparent BIST for RAMs", *Proc. of Int'l Test Conf.*, pp. 598–607.

[31] A. Castro, M. Nicolaidis, P. Lestart, and B. Courtois, "Built-In Self Test for

Multiport RAMs", *Proc. of Int'l Conf. on Computer Aided Design*, 1991.

[32] P. Bardell and W. McAnney, "Built-In Test for RAMs", *IEEE Design and Test*, pp. 29-36, August, 1988.

[33] M. Gallup, W. Ledbetler Jr., R. McGarity, and S. McMahan, "Testability Features of the 68040", *Proc. of Int'l Test Conf.*, pp. 749-757, 1990.

[34] R. Daniels and W. Bruce, "Built-In Self Test Trends in motolora Microprocessors", *IEEE Design and Test*, pp. 64-71, 1985.

[35] P. Gelsinger, "Design and Test of the 80386", *IEEE Design and Test*, pp. 42-50, June, 1987.

[36] P. Gelsinger, S. Iyengar, J. Kranskopf, and J. Nadir, "Computer Aided Design and Built In Self Test on the i486 CPU", *Proc. of Int'l Conf. on Computer Design*, pp. 199-202, 1989.

[37] L. Basto and J. Kuban, "Test Features of the MC68881 Floating Point Processor", *Proc. of Int'l Test Conf.*, pp. 752-757, 1985.

[38] S. Sarma, "Built In Self Test Considerations in a High Performance General Purpose Processor", *Proc. of Int'l Test Conf.*, pp. 21-27, 1991.

[39] R. Patel and K. Yarlagadda, "Testability Features of the SuperSPARC Microprocessor", *Proc. of Int'l Test Conf.* pp. 773-781, 1993.

[40] A. Crouch, M. Pressly, and J. Circello, "Testability Features of the MC68060 Microprocessor", *Proc. of Int'l Test Conf.*, pp. 60-69, 1994.

[41] M. Vilas and C. Closter Jr., "Automatic Built-in Self Test Insertion for High Level Circuit Descriptions", *Proc. of Test Conf.*, pp. 222-226, 1995.

[42] L. Avra, "Allocation and Assignment in High Level Synthesis for Self Testable Data Paths", *Proc. of Int'l Test Conf.*, pp. 463-472, 1991.

[43] K. Kim, J. Tront and D. Ha, "Automatic Insertion of BIST Hardware using VHDL", *Proc. of Design Automation Conf.*, pp. 9-15, 1988.

[44] C. Stroud, "An Automated BIST Approach for General Sequential Logic Synthesis", *Proc. of Design Automation Conf.*, pp. 3-7, 1988.

[45] A. Singh and J. Hurst, "Incorporating IDDQ Testing in BIST Improved Coverage through Test Diversity", *Proc. of ASIC Conf.*, pp. 374-379, 1994.

[46] W. Maly and M. Patyra, "Built In Current Testing", *IEEE Journal of Solid State Circuits*, pp. 425-428, March, 1992.

[47] S. Mir, V. Lolarik, M. Lubaszewski, C. Nielsen and B. Courtois, "Built-in Self-Test and Fault Diagnosis of Fully Differential Analogue Circuits", *ICCAD*, pp. 486-490, 1994.

[48] E. Teraoka, T. Kengaku, I. Yasui, K. Ishikawa, T. Matsuo, H. Wakada, N. Sakashita, Y. Shimazu, and T. Tokuda, "A Built-In Self Test for ADC and DAC in a Single-Chip Speech CODEC", *Proc of Int'l Test Conf.*, pp. 791-796, 1993.

[49] M. S. Nejad, L. Sebaa, A. Ladick, and H. Kuo, "Analog Built-In Self-Test", *ASIC*, pp. 407-411, 1994.

저 자 소 개

姜 成 昊

1963年 4月 13日生
1986年 2月 서울대학교 제어계측공학과 학사
1988年 5月 The University of Texas at Austin
전기 및 컴퓨터공학과 석사
1992年 5月 The of Texas at Austin
전기 및 컴퓨터공학과 공학박사

1989年 11月～1992年 8月 Schlumberger Inc., Research Scientist
1992年 9月～1992年 10月 The Univ. of Texas at Austin, Post Doctoral Fellow
1992年 8月～1994년 6月 Motorola Inc., Senior Staff Engineer
1994年 9月～현재 연세대학교 전기공학과 조교수

주관심분야 : VLSI 설계, VLSI testing, CAD for VLSI