論文2003-40SD-7-9

내장형 자체 테스트 패턴 생성을 위한 하드웨어 오버헤드 축소

((Reduction of Hardware Overhead for Test Pattern Generation in BIST))

金 玄 炖 *, 愼 鏞 升 **, 金 容 準 **, 姜 成 昊 ***

(Hyundon Kim, Yongseung Shin, Yongjoon Kim, and Sungho Kang)

요 약

최근 들어, 테스트 시간과 하드웨어의 축소를 위한 많은 내장형 자체 테스트 구조가 연구되고 있다. 대부분의 패턴 생성에 대한 내장형 자체 테스트 구조는 결정 패턴 생성을 위한 것이다. 본 논문에서는 테스트 시간과 하드웨어 오버헤드를 줄일 수 있는 새로운 의사 임의 패턴 내장형 자체 테스트 기법을 제안한다. 본 문에서는 의사 임의 패턴 내장형 자체 테스트 기법의 하드웨어 오버헤드의 축소 가능성에 대한 이론을 간단한 예제와 함께 설명하고 실험 결과를 통해 기존의 방법에 비하여 제안하는 방식을 이용할 경우 하드웨어 오버헤드가 줄어드는 것을 알 수 있으며, 기존의 방법과 제안한 방법의 테스트 시간 비교를 보여 준다.

Abstract

Recently, many BIST(Built-in Self Test) schemes have been researched to reduce test time and hardware. But, most BIST schemes about pattern generation are for deterministic pattern generation. In this paper a new pseudo-random BIST scheme is provided to reduce the existing test hardware and keep a reasonable length of test time. Theoretical study demonstrates the possibility of the reduction of the hardware for pseudo-random test with some explanations and examples. Also the experimental results show that in the proposed test scheme the hardware for the pseudo-random test is much less than in the previous scheme and provide comparison of test time between the proposed scheme and the current one.

Keyword : BIST, test pattern generation, hardware overhead, LFSR

Ⅰ. 서 론

내장형 자체 테스트 기법(BIST : Built in Self Test)은 점점 더 거대해지고 있는 회로의 테스트를 위한 효과적인 방법으로 주목받고 있는 방법이다[1]. 테스트를 위해 고가의 테스터를 사용하는 기존의 방법은 그 비용이 만만찮을 뿐 아니라, 점점 더 빨라지는 테스트 대상 회로를 동작 속도에서 테스트할 수 없기 때문에, 이제는 회로를 자체적으로 테스트할 수 있는 테스트 회로의 삽입이 절대적으로 필요하다.

내장형 자체 테스트 기법은 기본적으로 의사 무작위 패턴 생성 방법(pseudo-random pattern generation)을 통한 테스트 벡터를 이용한 테스트를 한다. 의사 무작위 패턴 생성 방법은 LFSR이라는 구조를 가장 널리 사용하는데, 테스트 패턴의 생성을 위해 최적화된 LFSR 구조 이외에 별도의 하드웨어가 거의 필요없다

* 正會員, 延世大學校 電氣電子工學科
(Dept. of Electrical Eng., Yonsei Univ.)

※ 이 연구는 2000년도 한국과학재단 연구비 지원에 의한 결과임.(과제번호 : 2000-1-30200-002-3).
接受日字:2002年10月21日, 수정완료일:2003年7月4日

는 장점이 있는 반면, 고장 검출율(fault coverage)을 원하는 만큼 끌어올릴지 보장할 수 없다는 단점이 있다. 대다수의 큰 회로는 의사 무작위 패턴 생성으로 100%의 고장 검출율을 얻지 못하는데, 여기까지 의사 무작위 패턴으로 검출되지 못한 고장을 검출난해고장(hard-to-detect fault)이라 한다. 이러한 검출난해고장의 수를 줄이기 위해, 의사 무작위 패턴 생성 방법에 약간의 변형을 가하는 재초기화(reseeding) 방법과 가중치 의사 무작위 패턴 생성(weighted pseudo-random pattern generation) 방법 등이 있다. 하지만, 그러한 방법들 역시 의사 무작위 테스트를 기본으로 한 약간의 변형이기 때문에, 완전 고장 검출율을 보장할 수는 없다. 따라서, 매우 직접적인 방법으로 테스트 패턴을 생성해내는 것이 불가피한데, 이러한 패턴 생성 방법을 결정 패턴 생성 방법이라 한다. 결정 패턴 생성 방법은 원하는 테스트 벡터를 구성하는 하나하나의 비트(bit)를 디코딩 로직(decoding logic)에서 생성하거나[2, 3], ROM에 미리 그 값들을 저장해놓고 하나씩 꺼내 쓰는 방법[1, 5]인데, 이는 상당한 양의 하드웨어를 소모하므로, 의사 무작위 패턴으로 고장검출율이 어느정도 포화상태에 이른 후에 검출난해고장에 대해서만 적용함으로써 추가적인 하드웨어의 양을 최소화한다.

본 논문은 기존의 패턴 생성을 위한 하드웨어를 효과적으로 활용하여, 의사 무작위 패턴 생성을 하는 하드웨어를 더욱 줄이면서도, 기존의 의사 무작위 테스트 방법에서의 고장검출율을 유지하는 방법을 제시한다. 이 방법은 테스트 시간도 더 큰 하드웨어를 필요로 하는 기존의 방법보다 더 걸리지 않으므로, 하드웨어 축소로 인한 결점도 없다는 것을 알 수 있다.

Ⅱ. 제안하는 패턴 생성 방법

기존의 의사 무작위 패턴 생성 방법은 단일 스캔체인을 사용한다고 가정할 경우, LFSR로 패턴을 생성해서 그 중 하나의 값을 계속해서 스캔체인으로 보내는 형태이다. 따라서, LFSR의 stage 수가 n이라 할 때, 스캔체인의 길이에 상관없이 2^n-1개 만큼의 패턴을 생성할 수 있다. 이를 위해 기본적인 하드웨어로는 비트 카운터(쉬프트 카운터)와 패턴 카운터를 들 수 있다. 비트 카운터는 스캔체인의 길이에 의해 결정되는데, 그 길이가 m이라 하면, $\lceil \log_2 m \rceil$ 만큼의 길이를 갖는 비트 카운터를 사용한다. 비트 카운터는 스캔 당 테스트를 하는 경우, 스캔체인에 쉬프트 과정이 끝나고 테스트 대상 회로로부터 응답을 받는 시점을 알려주는 역할을 한다. 따라서, 필수적인 구성요소이며, LFSR과 함께 스캔 당 테스트 방법에서 의사 무작위 패턴 생성을 위한 기본 요소라고 할 수 있다. 이 외에, 의사 무작위 테스트가 끝나는 시점을 알려주고, 결정 패턴 생성시 컨트롤을 위해 필요한 것이 패턴 카운터인데, 현재 몇 번째 패턴이 회로에 가해지고 있는지를 알려주는 인덱스의 역할을 한다고 할 수 있다. 즉, 스캔체인에 한 번 값이 다 채워져서 테스트 대상 회로에 패턴이 가해질 때마다 값이 하나씩 커지는 형태의 카운터로서, 모든 테스트 패턴이 회로에 다 가해지고 응답이 확인되었을 때 테스트를 종료하라는 신호를 생성하고 결정 패턴 생성시 언제 비트 플립핑(bit-fliping)을 할지 결정하는데 기준이 되는 역할을 하는 부분이다.

<그림 1>에 패턴 카운터와 비트 카운터의 일반적인 쓰임을 나타내는 그림이 있다. <그림 1>은 의사 무작위 패턴 뿐 아니라 결정 패턴도 생성할 수 있는 구조이고 디코딩 로직과 멀티플렉서가 없으면 의사 무작위 패턴을 생성하는 구조이다.

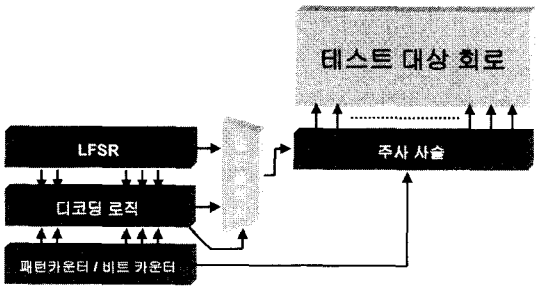


그림 1. 패턴 카운터와 비트 카운터의 쓰임
Fig. 1. Usage of Pattern counter and Bit counter.

대부분의 경우 큰 회로를 테스트하기 위해 32비트의 LFSR을 사용한다면, 패턴 카운터 32비트 이하, 비트 카운터는 위에서 언급한 크기로 사용해야 하는데, 제안하는 구조와 비교하기 위해 대조구로서 LFSR 32비트 혹은 12비트 (LFSR 12비트를 사용하는 것은 제안하는 방법에서 사용하는 하드웨어와 동일한 크기의 하드웨어를 쓰는 구조로서 비교하는 것이다.), 패턴 카운터와 비트 카운터를 합쳐서 32비트를 갖는 패턴 생성기 구조를 이용하였다.

제안하는 구조는 LFSR의 한 비트가 스캔체인으로 들어가는 형태와 달리, 원래 사용해야 하는 구조인 패

턴 카운터와 비트 카운터의 여러 비트 중 한 비트를 골라서 스캔체인으로 보내는 구조여서, 한 자리에서만 스캔체인으로 계속 값을 보내는 것이 아니며, 스캔체인으로 들어갈 값을 선택하는 역할은 작은 LFSR이 맡는다. 여기서는 5비트의 LFSR을 이용하였다.

새로운 구조의 의사 무작위 패턴 생성 방법은 <그림 2>와 같다. <그림 2>에서 보이는 구조는 5비트의 LFSR이 32비트의 [패턴 카운터 / 비트 카운터]에서 총 32비트중 한 비트를 제외한 31비트중 한 비트만 스캔체인으로 보낼 수 있도록 패턴을 생성한다. 32비트 모두가 아닌 31비트만이 멀티플렉서로 들어가는 이유는 5비트의 LFSR이 생성할 수 있는 패턴의 가지 수가 모두 0인 패턴(혹은 모두 1인 패턴)을 제외한 2^5-1가지이기 때문이다. <그림 2>에서 패턴 카운터와 비트 카운터는 각각 일반 카운터 혹은 LFSR 형태로 구성할 수 있으며, 비트 카운터는 일반 카운터의 형태를 유지하기로 한다. 뒤에 나오는 결과 테이블에서는 비트 카운터를 일반 카운터의 형태뿐 아니라 LFSR 형태인 경우의 결과도 포함하였다. 패턴 카운터의 경우는 비트 카운터와 함께 일반 카운터를 쓰는 경우 LFSR의 형태로 쓰는 경우보다 의사 무작위 테스트에서 고장검출율이 조금 낮아서 LFSR의 형태로 사용했고, XOR 게이트가 각 단의 외부에 있는 타입의 LFSR이 아닌 내부에 있는 타입의 LFSR의 형태를 사용하였다. 5비트의 LFSR에서 패턴 카운터와 비트 카운터의 값을 선택할 때, 0 또는 1의 값중 한 쪽의 값이 너무 많고 적을 경우, LFSR에서 선택하는 값이 0 또는 1쪽으로 몰릴 것이고, 그러한 패턴의 특성이 전체 고장 검출율을 빠른 시간 안에 올리는데 제한이 될 것이기 때문에, 한 클럭 주기에 0 또는 1의 수가 하나씩만 달라지는 타입의 LFSR(out-tapped LFSR)을 쓰기보다는 그 수가 여러 개씩 바뀔 수 있는 타입의 LFSR(in-tapped LFSR)을 사용하는 것이다.

이 구조가 기존의 구조보다 보다 적은 수의 비트를 사용하고도 같은 수준 혹은 그 이상의 고장 검출율을 유지할 수 있는 것은 의사 무작위 패턴 테스트를 하는 동안의 패턴 카운터와 비트 카운터의 역할을 증대시켰기 때문이다. 패턴 카운터에 패턴 생성의 역할을 맡겼지만, 패턴 카운터는 결정 패턴 테스트를 할 때, 컨트롤러의 역할도 수행할 수 있어야 하기 때문에, 결정 패턴을 생성할 때, 패턴 카운터의 값을 직접 바꾸는 방법은 사용하지 않는 것이 좋고, 이러한 것은 기존의 구조도 마찬가지 이므로, 새로운 구조에 별다른 제한을 가하는 것도 아니라고 할 수 있다.

<그림 1>과 <그림 2>에서 패턴 카운터와 비트 카운터의 총 비트 수를 같게 놓으면 <그림 2>의 LFSR 길이가 더 짧다. <그림 2>에서의 LFSR 길이를 L이라 하면, <그림 1>에 보이는 기존의 구조는 LFSR이 2^L의 길이를 가지므로 $2^L=m$이라 하면, 2^m-1의 패턴 생성 길이를 갖고, <그림 2>에 보이는 구조는 $(m-1)\times 2^m$정도의 패턴 생성 길이를 갖는다. 따라서, 이론적으로도 <그림 2>의 구조가 <그림 1>보다 좀 더 많은 의사 무작위 패턴을 생성할 수 있음을 알 수 있다.

이를 위한 간단한 예는 다음의 <그림 3>와 같다.

<그림 3>과 <표 1>에서 보이는 패턴 생성기는 간단한 예로서, 외부 XOR 타입의 2비트 LFSR, 패턴 카운터로서 내부 XOR 타입의 2비트 LFSR, 비트 카운터로서 2비트의 카운터를 이용하고, 스캔체인의 길이가 4인 경우이다. 실제의 경우에는 스캔체인의 길이를 h라 했을 때, 비트 카운터가 $\lceil \log 2h \rceil$비트만큼의 길이로서 스캔체인보다는 훨씬 짧은 길이가 되는데, 여기서의 간단한 예는, 패턴 카운터와 비트 카운터의 총 비트 수

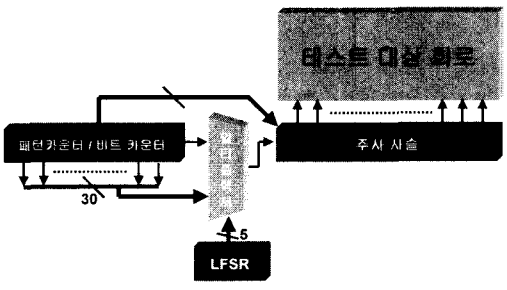


그림 2. 제안하는 의사 무작위 패턴 생성 구조
Fig. 2. The proposed structure for pseudo-random pattern generation.

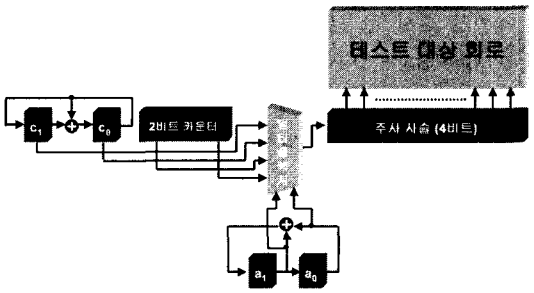


그림 3. 제안하는 방법의 패턴 생성의 간단한 예
Fig. 3. A simple example of the proposed pattern generation.

표 1. <그림 3>의 패턴 생성기로부터의 패턴
Table 1. Patterns from the pattern generator in Fig. 3.

a_1	a_0	c_1	c_0	2비트 카운터		주사사슬			
0	1	1	1	0	0	x	x	x	x
1	0	1	0	0	1	1	x	x	x
1	1	0	1	1	0	0	1	x	x
0	1	1	1	1	1	0	0	1	x
1	0	1	0	0	0	1	0	0	1
1	1	0	1	0	1	0	1	0	0
0	1	1	1	1	0	1	0	1	0
1	0	1	0	1	1	1	1	0	1
1	1	0	1	0	0	1	1	1	0
0	1	1	1	0	1	0	1	1	1
1	0	1	0	1	0	1	0	1	1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

와 스캔체인의 길이가 같게 구성되어서, 실제처럼 스캔 체인에서 다양한 패턴이 생성되지 않을 수도 있다. <표 1>에서 보여주는 패턴의 생성 주기는 시작 벡터 ($a_1a_0c_1c_0$[2비트카운터]) = 011100로부터 그것이 다시 반복되는 시점 사이의 간격인 $(4-1)(2^2-1)\times 2^2 = 36$이다.

Ⅲ. 실험결과

이번 실험은 ISCAS85 와 ISCAS89의 조합회로 및 순차회로를 대상으로 하였고, 그 결과는 아래의 <표 2>, <표 3>과 같다.

<표 2>는 4개의 섹션으로 나뉘어 있는데, 첫 번째와 두 번째 섹션은 대조구로서, 첫 번째 섹션은 기존의 32비트 LFSR을 사용한 결과이고, 두 번째 섹션은 세 번째, 네 번째와 동일한 하드웨어를 소모하는 크기의 패턴 생성기를 사용한 결과이다. 세 번째 섹션은 비트 카운터를 LFSR 형태로 구현한 것이고, 네 번째 섹션은 비트 카운터를 카운터 형태로 구현한 것이며, 세 번째, 네 번째 모두 5비트 LFSR을 사용한 결과이다. 각각의 섹션은 검출고장수와 고장 검출율로 이루어져 있는데, 이는 각각 모든 고장 중 검출된 고장의 수와 의사 무작위 테스트의 고장 검출율을 나타낸다. <표 2>에서도 알 수 있듯이, 5비트의 LFSR을 사용한 것이 고장검출율이 평균적으로 비슷하거나 약간 우위에 있는 것을 알 수 있다. 또한, 12비트 LFSR을 사용한 기존 방법의 경우, 큰 회로에 대해서는 고장 검출율이 현저하게 떨어지는 것을 볼 수 있다. 여기서 기존 방법의 12비트 LFSR을 사용한 경우를 비교한 이유는 이를 위한 하드웨어 오버헤드가 새로 제안한 방법의 5비트 LFSR과 멀티플렉서를 사용하는 경우의 하드웨어 오버헤드와 동일하다는 가정을 했기 때문이다. 따라서, 기존 방법을 사용한 32비트 LFSR의 경우는 새로 제안한 방법보다 약 20개의 stage를 더 갖는 하드웨어의 부담을 갖는다고 할 수 있다. 이와같이, 5비트의 LFSR을 사용한 새로운 구조는 훨씬 큰 하드웨어를 사용하는 기존의 방법에 비해 하드웨어를 덜 쓰면서, 의사 무작위 패턴 테스트에서 비슷하거나 더 높은 고장 검출율을 보인다는

표 2. 작은 크기의 LFSR로 구한 고장 검출율
Table 2. Fault coverages with smaller LFSR.

회로	32비트 LFSR (기존)		12비트 LFSR (기존)		5비트 LFSR (LFSR)		5비트LFSR (카운터)	
	남은 고장수	고장 검출율	남은 고장수	고장 검출율	남은 고장수	고장 검출율	남은 고장수	고장 검출율
c432	4	99.24	4	99.24	4	99.24	4	99.24
c499	8	98.94	8	98.94	8	98.94	8	98.94
c880	23	97.56	34	96.39	13	98.62	21	97.77
c1355	16	98.98	13	99.17	13	99.17	10	99.36
c1908	19	98.99	25	98.67	16	99.15	18	99.04
c2670	429	84.38	435	84.16	432	84.27	432	84.27
c3540	146	95.74	157	95.42	158	95.39	148	95.68
c5315	63	98.82	68	98.73	66	98.77	64	98.80
c6288	34	99.56	34	99.56	34	99.56	34	99.56
c7552	491	93.50	543	92.81	460	93.91	465	93.84
s208	5	97.67	5	97.67	4	98.14	3	98.60
s344	0	100.00	0	100.00	0	100.00	0	100.00
s349	2	99.43	2	99.43	2	99.43	2	99.43
s382	0	100.00	0	100.00	0	100.00	0	100.00
s386	1	99.74	36	90.62	2	99.48	12	96.88
s400	6	98.58	6	98.58	6	98.58	6	98.58
s420	44	89.77	67	84.42	33	92.33	35	91.86
s444	14	97.05	14	97.05	14	97.05	14	97.05
s510	0	100.00	0	100.00	0	100.00	0	100.00
s526	16	97.12	13	97.66	15	97.30	18	96.76
s641	12	97.43	19	95.93	22	95.29	20	95.72
s1196	57	95.41	52	95.81	72	94.20	51	95.89
s1238	134	90.11	126	90.70	132	90.26	129	90.48
s1423	50	96.70	162	89.31	26	98.28	26	98.28
s1488	14	99.06	28	98.12	27	98.18	24	98.38
s1494	26	98.27	41	97.28	33	97.81	36	97.61
s5378	158	96.57	146	96.83	131	97.15	134	97.09
s13207	857	91.27	2562	73.90	444	95.48	602	93.87
s15850	1025	91.26	1958	83.30	1034	91.18	1121	90.44
s38417	2156	93.09	4526	85.48	2424	92.23	2043	93.45
s38584	2028	94.41	3613	92.80	1946	94.64	1922	94.71

표 3. 작은 크기의 LFSR로 테스트하는데 걸린 CPU time

Table 3. CPU-times taken by tests with smaller LFSR.

circuits	32비트 LFSR (기존)			5비트 LFSR (LFSR)			5비트 LFSR (카운터)		
	남은 고장수	고장 검출율	총 패턴수	남은 고장수	고장 검출율	총 패턴수	남은 고장수	고장 검출율	총 패턴수
c432	4	99.24	42624	4	99.24	43776	4	99.24	47232
c499	8	98.94	48544	8	98.94	52480	8	98.94	51168
c880	23	97.56	90240	13	98.62	140160	21	97.77	90240
c1355	16	98.98	82656	13	99.17	101024	10	99.36	86592
c1908	19	98.99	88704	16	99.15	103488	18	99.04	106656
c2670	429	84.38	350432	432	84.27	812704	432	84.27	350432
c3540	146	95.74	163200	158	95.39	131200	148	95.68	179200
c5315	63	98.82	324672	66	98.77	364544	64	98.80	273408
c6288	34	99.56	35840	34	99.56	35840	34	99.56	35840
c7552	491	93.50	874368	460	93.91	854496	465	93.84	1033344
s208	5	97.67	24320	4	98.14	29792	3	98.60	21888
s344	0	100.00	6144	0	100.00	8448	0	100.00	3072
s349	2	99.43	26112	2	99.43	26880	2	99.43	26880
s382	0	100.00	10752	0	100.00	10752	0	100.00	10752
s386	1	99.74	42848	2	99.48	20384	12	96.88	20800
s400	6	98.58	30720	6	98.58	28416	6	98.58	26880
s420	44	89.77	77280	33	92.33	43680	35	91.86	84000
s444	14	97.05	37632	14	97.05	33024	14	97.05	28416
s510	0	100.00	16800	0	100.00	19200	0	100.00	17600
s526	16	97.12	39936	15	97.30	36864	18	96.76	34560
s641	12	97.43	86400	22	95.29	74304	20	95.72	69120
s1196	57	95.41	98304	72	94.20	89088	51	95.89	94208
s1238	134	90.11	98304	132	90.26	125952	129	90.48	97280
s1423	50	96.70	131040	26	98.28	224224	26	98.28	206752
s1488	14	99.06	30016	27	98.18	26880	24	98.38	26432
s1494	26	98.27	30016	33	97.81	27776	36	97.61	26432
s5378	158	96.57	540992	131	97.15	979264	134	97.09	862848
s13207	857	91.27	6496000	444	95.48	10841600	602	93.87	8220800
s15850	1025	91.36	5474560	1034	91.18	3284736	1121	90.44	2502656
s38417	2156	93.09	19701760	2424	92.23	28434432	2043	93.45	21512192
s38584	2028	94.41	13773312	1946	94.64	18270720	1922	94.71	14241792

것을 알 수 있다. 의사 무작위 테스트라는 것은 이론적으로는 이미 성능면에서 최적화되어 있는 LFSR을 사용하지만, 이와 동반하여 패턴 카운터와 비트 카운터를 함께 사용하므로, 그 카운터들을 패턴 생성에 이용함으로써 더 작은 하드웨어를 사용하고도 같은 성능의 패턴 생성을 할 수 있게 하였다.

<표 3>은 고장 검출율 외에 또 다른 성능평가 항목인 테스트 시간을 검증한 것으로 기존의 방법에 비해 더 많은 패턴이 필요하지 않음을 알 수 있으므로, 새로운 구조의 효율성을 입증하고 있다. <표 3>에서는 <표 2>에 있는 기존의 방법의 12비트 LFSR을 제외하였는데, 이는 이미 <표 2>에서 새로 제안한 방법이 기존의 12비트 LFSR을 사용한 방법보다 우수함은 보였으므로, <표 3>에서는 그보다 훨씬 큰 하드웨어를 사용하는 기존 방법의 32비트 LFSR과 새로 제안하는 구조의 고장 검출율을 구하는데까지의 가한 패턴 수를 비교하여, 새로 제안한 방법에서 생성되는 의사 무작위 패턴의 다양함을 보이려 한 것이다.

Ⅳ. 결 론

의사 무작위 패턴 생성시 스캔체인의 컨트롤만 하는 비트 카운터와 결정 패턴 테스트에서의 비트 플립핑을 위한 컨트롤 정도의 역할만 하는 패턴 카운터는 그 하드웨어의 크기에 비해 의사 무작위 패턴 생성시 별 역할을 하지 못하므로, 이를 의사 무작위 테스트에서도 충분히 활용하여 그 효율성을 높일 필요가 있다.

본 연구에서는 두 하드웨어를 의사 무작위 패턴 생성시 다른 역할도 수행하게 하여, 기존의 의사 무작위 패턴 생성에 필요한 하드웨어를 줄이고도, 높은 고장 검출율을 얻고, 테스트 시간도 증가되지 않는 결과를 내었다. 또, 작은 하드웨어를 사용하기 때문에 생겨날 수 있는 문제들을 세부적인 구조 변화를 통해 보완함으로써, 보다 효율적인 테스트를 할 수 있게 하였다. ISCAS85, ISCAS89 회로의 실험을 통하여 새롭게 제안한 구조의 효율성을 고장 검출율과 테스트 시간의 측면에서 입증하였다.

참 고 문 헌

[1] M. Abramovici, M. A. Breuer and A. D. Fiedman, Digital Systems Testing and Testable Design, Computer Science Press, 1990.

[2] N. A. Touba and E. J. McCluskey, "Bit-fixing in pseudorandom sequences for scan BIST", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, pp. 545~555, April 2001.

[3] G. Kiefer and H. J. Wunderlich, "Using BIST Control for Pattern Generation", Proc. of IEEE International Test Conference, pp. 347~355, 1997.

[4] K. Chakrabarty, B. T. Murray and V. Iyengar,

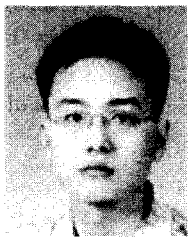
"Deterministic Built-In Self Test Pattern Generation for High-Performance Circuits using Twisted-Ring Counters", IEEE Trans. on VLSI Systems, vol. 8, no. 5, pp. 633~636, October 2000.

[5] S. Hellebrand, H. G. Liang and H. J. Wunderlich, "A Mixed Mode BIST Scheme Based On Reseeding of Folding Counters", Proc. of IEEE International Conference, pp. 778~784, 2000.

저 자 소 개

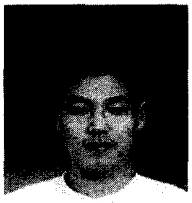
金 玄 炖(正會員)

2001년 2월 : 연세대학교 전기공학과 졸업. 2003년 2월 : 연세대학교 전기전자공학과 졸업(석사). 현재 : 삼성전자 시스템 LSI 사업부 SOC연구소



愼 鏞 升(正會員)

2002년 2월 : 연세대학교 전기공학과 졸업. 현재 : 연세대학교 전기전자공학과 석사과정



金 容 準(正會員)

2002년 2월 : 연세대학교 전기공학과 졸업. 현재 : 연세대학교 전기전자공학과 석사과정

姜 成 昊(正會員)

1986년 2월 : 서울대 공대 제어계측공학과 졸업. 1988년 5월 : The University of Texas at Austin 전기 및 컴퓨터공학과 졸업(석사). 1992년 5월 : The University of Texas at Austin 전기 및 컴퓨터공학과 졸업(공박). 미국 Schlumberger 연구원. Motorola 선임 연구원. 현재 : 연세대학교 공과대학 전기전자공학과 부교수