논문 2005-42SD-12-11

결정론적 테스트 세트의 신호확률에 기반을 둔 clustered reconfigurable interconnection network 내장된 자체 테스트 기법

(A Clustered Reconfigurable Interconnection Network BIST Based on Signal Probabilities of Deterministic Test Sets)

송 동 섭*, 강 성 호**

(Dong-Sup Song and Sungho Kang)

요 약

본 논문에서는 의사무작위패턴만으로는 생산하기 힘든 결정론적 테스트 큐브의 생산확률을 높일 수 있는 새로운 clustered reconfigurable interconnect network (CRIN) 내장된 자체 테스트 기법을 제안한다. 제안된 방법은 주어진 테스트 큐브들의 신호확률에 기반을 둔 스캔 셀 재배치 기술과 규정 비트(care-bit: 0 또는 1)가 집중된 스캔 체인 테스트 큐브의 생산확률을 높이기 위한 전용의 하드웨어 블록을 사용한다. 테스트 큐브의 생산확률을 최대로 할 수 있는 시뮬레이티드 어닐링(simulated annealing) 기반 알고리듬이 스캔 셀 재배치를 위해 개발되었으며, CRIN 하드웨어 합성을 위한 반복 알고리듬 또한 개발되었다. 실험을 통하여 제안된 CRIN 내장된 자체 테스트 기법은 기존의 연구 결과보다 훨씬 적은 저장 공간과 짧은 테스트 시간으로 100%의 고장검출율을 달성할 수 있음을 증명한다.

Abstract

In this paper, we propose a new clustered reconfigurable interconnect network (CRIN) BIST to improve the embedding probabilities of random-pattern-resistant-patterns. The proposed method uses a scan-cell reordering technique based on the signal probabilities of given test cubes and specific hardware blocks that increases the embedding probabilities of care bit clustered scan chain test cubes. We have developed a simulated annealing based algorithm that maximizes the embedding probabilities of scan chain test cubes to reorder scan cells, and an iterative algorithm for synthesizing the CRIN hardware. Experimental results demonstrate that the proposed CRIN BIST technique achieves complete fault coverage with lower storage requirement and shorter testing time in comparison with the conventional methods.

Keywords : Deterministic Logic BIST, embedded core testing

I. 서 론

로직 회로에 대한 내장된 자체 테스트 (BIST: Built-In Self Test)는 외부 테스터 (ATE: Automatic Test Equipment)의 대부분의 기능을 칩에 내장하고, 시스템온칩 (SoC: System-on-a-Chip)에 내장된 복잡한 코어에 대한 테스트 접근을 적절하게 획득하기 어려운 문제를 효과적으로 해결한다. 내장된 자체 테스트의 효율성은 충분히 높은 혹은 100%의 고장검출율을 달성하는데 필요한 테스트 시간과 하드웨어 오버헤드로 특징지워진다[1]-[3]. LFSR (Linear Feedback Shift Register) 은 적은 하드웨어만으로 구현 가능한 특징 때문에 내장된 자체 테스트의 의사무작위테스트 패턴 (pseudo-random test pattern) 생성기로 널리 사용된다. 하지만 무작위패턴저항고장 (RPRF: Random Pattern Resistive

* 학생회원 ** 평생회원 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Graduate School, Yonsei University)
※ 본 논문은 IDEC(IC Design Education Center)의 CAD 프로그램 지원에 의한 것임
접수일자: 2005년5월16일, 수정완료일: 2005년11월23

Fault) 때문에 내장된 자체 테스트의 효율은 떨어지게 되고, 높은 고장검출율을 달성하기 위해서는 많은 수의 무작위 패턴을 필요로 하게 되는 단점이 있다. 이런 무작위패턴저항고장의 문제를 극복하기 위해, 테스트포인트 삽입(test point insertion)[4][5], 가중치 무작위 패턴(weighted random pattern)[6][7], 그리고 혼합모드 테스트가 제안되었고, 이들 다양한 테스트 기술은 고장검출율, 하드웨어 오버헤드, 그리고 테스트 시간간의 다양한 상호절충(trade-off)을 제공한다.

혼합 모드 테스트 기술은 쉽게 검출 가능한 고장을 제거하기 위해서 제한된 수의 의사무작위패턴을 먼저 인가하고, 나머지 무작위패턴저항고장의 검출을 위해서 결정론적 테스트 패턴(deterministic test pattern)을 인가하는 방법을 사용한다. 테스트포인트 삽입 방법과는 달리 혼합 모드 테스트 기술은 테스트 대상 회로의 수정 또는 기능 저하 없이 높은 고장검출율을 달성할 수 있는 장점을 가진다. 무엇보다도, 혼합 모드 테스트 기술은 결정론적 테스트 패턴과 의사무작위패턴의 상대적인 수를 조절하여 테스트 데이터를 저장하기 위한 메모리의 양과 테스트 시간 간의 다양한 상호절충을 이끌 수 있다는 장점이 있다. 그러므로 혼합 모드 테스트 기술은 적은 하드웨어 오버헤드와 메모리의 사용으로 높은 혹은 100%의 고장검출율을 달성하는데 있어 문제해결의 훌륭한 방법을 제공할 수 있다. 지금까지 혼합모드 내장된 자체 테스트에 관한 다양한 연구가 진행되어 왔다[8]-[13]. LFSR reseeding[8]-[11]과 bit-flipping [12][13]이 혼합 모드 내장된 자체 테스트의 대표적 예라 할 수 있다. LFSR reseeding은 결정론적 테스트 패턴을 LFSR의 씨드(seed)로 인코딩하고, 결정론적 테스트 패턴의 수에 따라 다수개의 LFSR 씨드를 사용하는 기술이다. 하지만 다수의 무작위패턴저항고장을 포함하는 회로에 대해 이 방법은 LFSR로 씨드를 공급하는데 필요한 테스트 시간 오버헤드를 필요로 할 뿐만 아니라, 각 결정론적 테스트 패턴에 포함된 규정비트(care bit: 0 또는 1)의 수에 따라 그 사이즈가 결정되는 LFSR의 하드웨어 오버헤드가 커질 수 있다는 단점을 갖는다. Bit-flipping 기술은 결정론적 테스트 패턴을 생산하기 위해서 의사무작위패턴의 특정 비트를 반전시키는 방법을 사용한다. 이 방법은 실용적인 테스트 시간동안 높은 고장검출율을 달성할 수 있다는 장점이 존재한다. 하지만 반전되어야 할 비트의 수가 많은 회로에 대해서는 하드웨어 오버헤드가 너무 커진다는 단점이 있다.

최근에, LFSR과 각 스캔 체인 사이의 연결선을 테스트 중 변경할 수 있는 재구성 가능한 연결선 네트워크(RIN: Reconfigurable Interconnection Network) 내장된 자체 테스트 기술이 제안되었다[14][15]. 그림 1은 s9234 벤치마크 회로의 테스트 큐브를 생산하는데 있어 RIN의 특성을 보여준다. 그림 1(b)의 x축에 표시된 'The order of patterns'는 테스트 큐브가 포함한 규정비트의 수가 많은 순서로 정렬한 각 테스트 큐브의 순서를 의미한다. 그림 1이 나타내는 것과 같이 RIN은 총 1557개의 테스트 큐브 중 3분의 2인 1000개의 테스트 큐브를 총 517733개의 인가한 테스트 패턴 중 5분의 1인 100000개의 테스트 패턴 안에 생산한다. 또한 인가된 테스트 패턴의 5분의 1안에 생산되지 못한 테스트 큐브 중 대부분은 정렬된 순서로 총 테스트 큐브의 3분의 1에 속한다. 결과적으로 RIN 기법에서는 많은 수의 규정비트를 포함하고 있는 테스트 큐브를 무작위패턴저항패턴(RPRP: Random Pattern Resistant Pattern)으로 규정할 수 있고, 이 무작위패턴저항패턴이 RIN 기법에서는 테스트 시간과 하드웨어 오버헤드를 증가시키는 원인으로 작용한다.

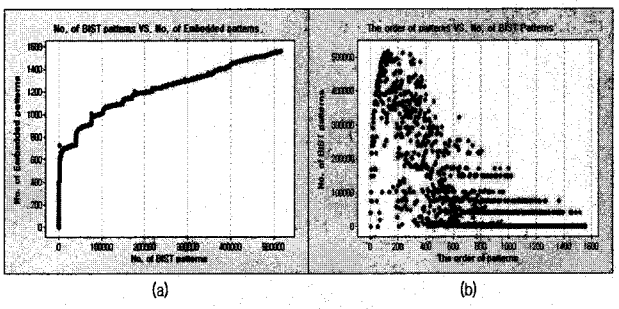


그림 1. RIN의 결정론적 테스트 큐브 생성 특징
Fig. 1. Characteristics of RIN to embed deterministic test cube.

본 논문에서는 무작위패턴저항패턴의 생산확률을 높이기 위한 새로운 CRIN(Clustered RIN) 내장된 자체 테스트 기법을 제안한다. 제안된 CRIN 기법은 결정론적 테스트 큐브 세트의 신호확률에 기반을 둔 스캔 셀 재배치 기술과, 재배열된 테스트 큐브의 생산확률을 향상시키기 위한 전용의 하드웨어 블록을 사용한다. ISCAS 벤치마크 회로에 대한 실험 결과는 제안된 방법이 적은 메모리와 하드웨어 오버헤드의 사용으로 100%의 고장검출율과 짧은 테스트 시간을 달성하는 문제에 있어 효과적인 해결책을 제시 할 수 있다는 것을 증명한다.

Ⅱ. 제안하는 내장된 자체 테스트 기법

1. 기본 원리

기존의 RIN 기법에서 LFSR로부터 발생한 테스트 패턴에 의한 무작위패턴저항패턴의 생산확률은 각 스캔 체인 테스트 큐브에 포함된 X의 수에 비례한다. 스캔 체인 테스트 큐브에 포함된 평균 X의 수를 증가시키기 위해, 기존의 RIN 내장된 자체 테스트 기법은 규정비트를 모든 스캔 체인에 걸쳐 골고루 분포하도록 분산시키는 스캔 셀 재배치 방법을 사용하였다[14][15]. 하지만, 만일 0-규정비트와 1-규정비트가 각각 별도의 스캔 체인으로 집중 된다면 무작위패턴저항패턴의 생산확률을 높일 수 있다.

LFSR로부터의 테스트 패턴 P_1, P_2, , P_{10}으로 결정론적 테스트 큐브 C:XX01000001110XXX의 생산확률을 높이는 예를 그림 2에 나타내었다. 그림 2에서, (a)는 기존 [14]에서 제안된 스캔 셀 재배치 알고리즘에 따라 테스트 큐브 C의 규정비트를 4개의 스캔 체인 테스트 큐브인 t_1, , t_4로 골고루 분산하도록 하는 경우를 보여주고 있으며, 재배치된 테스트 큐브 C'은 LFSR이 10번째 생산한 테스트 패턴 P_{10}에서 생산 가능하다. 이 때, 괄호 안의 숫자 (2, 4)는 만일 LFSR의 두 번째 스테이지 출력 또는 4번째 스테이지 출력 중 어느 하나를 첫 번째 스캔 체인에 연결한다면, 스캔 체인 테스트 큐브 t_1은 생산가능 하다는 것을 의미한다. 반면 그림 2(b)에서는 테스트 큐브 C가 다음 절에서 설명할 본 논문에서 제안하는 새로운 스캔 셀 재배치 알고리즘에 따라 0-규정비트는 스캔 체인 테스트 큐브 t_1으로, 그리고 1-규정비트는 스캔 체인 테스트 큐브 t_4로 각각 집중되어 재배치된 테스트 큐브 C''을 형성하였다. 설명의 편의를 위해서, 앞으로는 0(1)-규정비트가 집중된 스캔 체인을 AND(OR)-Chain으로, 그리고 AND-Chain과 OR-Chain을 제외한 나머지 스캔 체인을 LFSR-Chain으로 명명하겠다. 테스트 큐브 C의 규정비트가 AND-Chain과 OR-Chain으로 집중되었기 때문에, LFSR로부터 테스트 패턴을 공급받는 LFSR-Chain의 스캔 체인 테스트 큐브는 기존 RIN 기법의 경우보다 많은 수의 X를 가지게 되고, 결과적으로 LFSR-Chain에 대한 스캔 체인 테스트 큐브의 생산확률을 향상시킬 수 있다. 제안하는 CRIN 기법에서 AND(OR)-Chain에 대한 테스트 패턴은 LFSR의 출력을 입력으로 하는 AND(OR) 게이트들로 이루어진 AND(OR) 블록에 의해 공급받는다. AND(OR) 블록의 출력은 0(1)의 출력 비트로 가중되기 때문에, 0(1)-규정비트가 집중된 AND(OR)-Chain에 대한 스캔 체인 테스트 큐브의 생산확률 또한 향상시킬 수 있다. 그림 2(b)에서 *1&2(1\|2)*는 LFSR의 첫 번째와 두 번째 스테이지 출력을 입력으로 하는 AND(OR) 게이트에 의해 생산된 비트 시퀀스를 의미한다. 위의 경우 C''은 LFSR이 생산한 테스트 패턴 P_2, P_3, P_6, 그리고 P_{10}에 의해서 생산 가능하고 패턴 생성기와 각 스캔 체인간의 가능한 연결선의 수에 있어서도 기존 RIN 기법보다 많은 경우의 수를 갖는다. 결과적으로 본 논문에서 제안하는 CRIN 기법은 비교적 적은 하드웨어 오버헤드와 테스트 시간으로 100% 고장 검출율을 달성하는 문제에 있어서 효과적인 해결책을 제시할 수 있다.

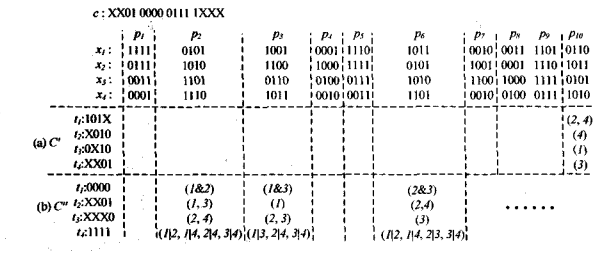


그림 2. 테스트 큐브 발생 확률 향상의 예
Fig. 2. An example of enhancing embedding probability.

2. 새로운 스캔 셀 재배치 알고리즘

이번 절에서는 주어진 결정론적 테스트 큐브 세트의 신호확률에 기반을 둔 새로운 스캔 셀 재배치 알고리즘 과정에 대해 설명한다. 제안하는 알고리즘은 AND-Chain, OR-Chain, 그리고 LFSR-Chain의 수를 계산하고, 스캔 체인 테스트 큐브의 생산확률을 극대화하기 위해서 스캔 셀을 재배치한다.

새로운 스캔 셀 재배치 알고리즘은 시뮬레이티드 어닐링(simulated annealing) 절차에 의해 개발되었으며, 그림 3에 나타내었다. 본 논문에서는 표 1이 나타내는 기호를 사용한다.

표 1. 기호의 정의
Table 1. Definitions of notations.

기호	정의
m	총 스캔 체인의 수
L	LFSR의 스테이지 수
l	스캔 체인의 길이
S_i	$0 \le i \le m \times l$, 각 스캔 셀의 순서
T_0	결정론적 테스트 큐브 세트
N_{TV}	총 테스트 큐브의 수
N_{AND}	총 AND-Chain의 수
N_{OR}	총 OR-Chain의 수
N_{LFSR}	총 LFSR-Chain의 수

```
INPUT : Deterministic pattern set T_D, T_init, T_low, IPT, K_T, Initial scan cell order S_i
OUTPUT : Reordered scan cell S_i', Reordered deterministic pattern set T_D'

Calculate signal probability of each scan cell S_i                    ..... (a)
Reorder scan cells and group AND-chains, OR-chains, and LFSR-chains   ..... (b)
For each group {
   T = T_init ;
   sv = Deterministic pattern set T_D corresponding current S_i ;
   Compute cost c(sv) ;                                               ..... (c)
   while (T > T_low) {
      if (no cost reduction in the last IPT iterations) break ;
      for (i=0; i<IPT; i++) {
         (max, min ) = Find_swap_target(sv) ;                         ..... (d)
         (s_i', sv') = movement(max, min, sv, s_i) ;                  ..... (e)
         Compute cost C(sv') ;
         ΔC = C(sv') - C(sv) ;
         if (ΔC < 0) {
            sv = sv' ; s_i = s_i' ;
         }
         else {
            p = e^(-ΔC/T) ;
            if (random(0,1) < p) sv = sv' ;
            else movement(max, min, sv', s_i') ;
         }
      }
      T = K_T × T ;
   }
}
T_D' = sv' ;
```

그림 3. 새로운 스캔 셀 재배치 알고리즘
Fig. 3. New scan cell reordering algorithm.

(a) 전체 결정론적 테스트 큐브 중 규정비트가 많은 순서로 3분의 1의 테스트 큐브를 이용하여 각 스캔 셀 S_i의 신호확률 P_i를 계산한다. 하나의 결정론적 테스트 큐브와 결정론적 테스트 큐브 세트를 각각 C_j와 $T_D = \{C_1, C_2, , C_{NTD}\}$라 하자. 결정론적 테스트 큐브 C_j의 i번째 비트와 비트 포지션 i의 신호확률은 각각 C_{ji}와 P_i로 정의되고 P_i는 다음의 식에 의해서 계산된다.

$$P_i = \frac{\{C_j \in T_D | C_{ji} = 1\}}{\{C_j \in T_D | C_{ji} \neq X\}} \quad \text{for } 1 \le i \le m \times l \text{ and } 1 \le j \le \frac{N_{TD}}{3} \tag{3}$$

총 테스트 큐브의 3분의 1만을 이용하여 P_i를 계산하는 이유는 대부분의 무작위패턴저항패턴이 규정비트가 많은 순서로 처음 3분의 1의 테스트 큐브에 집중 된다는 실험결과 때문이다.

(b) P_i가 낮은 순서대로 스캔 셀을 재배치한다. 또한 0.25 이하의 P_i를 갖는 스캔 셀을 이용하여 AND-Chain을 형성하고, 0.75 이상의 P_i를 갖는 스캔 셀을 이용하여 OR-Chain을 형성한다. LFSR-Chain은 나머지 스캔 셀을 이용하여 구성한다. 과정 (b)의 영향으로 0(1)-규정비트는 AND(OR)-Chain에 집중될 수 있다.

(c) 과정 (b)를 마친 후의 재배치된 AND(OR)-Chain의 스캔 체인 테스트 큐브에는 많은 수의 0(1)-규정비트와 X가 존재한다. 하지만, AND-Chain, LFSR-Chain, 그리고 OR-Chain의 각 스캔 체인 그룹 내에서 스캔 셀 S_i의 정확한 위치는 아직 정하지 않았기 때문에, 재배치된 테스트 큐브의 생산확률을 높일 수 있는 가능성은 아직 존재한다. 제안된 CRIN 내장된 자체 테스트에서 최상의 패턴 생산확률은 스캔 체인 테스트 큐브가 가능한 많은 X를 포함할 때 달성된다. 다음의 'for-loop'는 X를 모든 스캔 체인에 분산 시키는 역할을 한다. 주의해야 할 것은 loop는 각 스캔 체인 그룹(AND-Chain, OR-Chain, LFSR-Chain)에 대해서 적용된다는 것이다. 그러므로, 두 개의 스캔 셀에 대한 자리바꿈은 하나의 그룹에서만 발생하고 각 그룹의 총 신호확률은 변하지 않는다. i번째 테스트 큐브의 j번째 스캔 체인 테스트 큐브에 대해 총 규정비트의 수를 sp_{ij}라 하자. 가치함수(cost function) $c(sv)$는 다음의 식에 의해서 계산된다.

$$ave_i = \frac{\sum_{j=1}^{m} sp_{ij}}{m} \quad \text{for } 1 \le i \le N_{TD} \tag{2}$$

$$dis_i = \frac{\sum_{j=1}^{m} (sp_{ij} - ave_i)^2}{m-1} \quad \text{for } 1 \le i \le N_{TD} \tag{3}$$

$$c(sv) = \frac{\sum_{i=1}^{N_{TD}} \sqrt{dis_i}}{N_{TD}} \tag{4}$$

(d) 가치함수 $c(sv)$의 값을 줄이기 위해, 위치가 서로 바뀔 두 개의 스캔 셀을 찾는다. sp_{ij}-ave_i의 값이 최대인 스캔 체인 테스트 큐브를 $B_{max}(1 \le i \le N_{TD}, 1 \le j \le a)$, 그리고 B_{max}와 동일한 테스트 큐브에서 sp_{ij}-ave_i의 값이 최소인 스캔 체인 테스트 큐브를 B_{min}이라 하자. 여기서 a는 AND-Chain 그룹, LFSR-Chain 그룹, 그리고 OR-Chain 그룹 각각에 대해서 N_{AND}, N_{LFSR}, 그리고 N_{OR}를 의미한다. *maximum*은 C_{ik}가 B_{max}의 임의의 규정비트일 때, C_{ik}에 해당하는 스캔 셀 S_k이다. 또한 *minimum*은 C_{ik}가 B_{mim} 내의 임의의 X 일 때, C_{ik}에 해당하는 스캔 셀 S_k를 의미한다.

(e) 과정 (d)에서 찾은 *maximum*과 *minimum* 2개의 스캔 셀의 위치를 교환하고, 재배치된 스캔 셀 구

성 S_i'과 재구성된 결정론적 테스트 큐브 세트 T_D'을 얻는다.

3. 내장된 자체 테스트 하드웨어

그림 4는 제안된 내장된 자체 테스트의 하드웨어 구조를 나타낸다. RIN 블록은 멀티플렉서 스위치로 구성되며, 적당한 D_0, D_1, , D_{g-1}의 제어비트 인가로 재구성될 수 있다. 파라미터 g는 내장된 자체 테스트 동안 사용되는 RIN의 총 구성(configuration) 횟수를 의미하며 시뮬레이션 절차에 의해서 구해진다.

AND(OR)-Chain에 대한 테스트 패턴은 AND(OR) 블록에 의해서 제공되며, AND(OR) 블록은 LFSR의 출력을 입력으로 하는 2입력 AND(OR) 게이트들로 구성된다. LFSR-Chain의 테스트 패턴은 LFSR에 의해서 제공된다. AND(OR) 블록의 출력은 0(1)로 가중되기 때문에, 0(1)-규정비트가 집중된 AND(OR)-Chain의 스캔 체인 테스트 큐브에 대한 생산확률을 증가시킬 수 있다. 제어 입력 D_0, D_1, , D_{g-1}은 d 입력 g 출력($d = \log_2 g$) 디코더에 의해서 제공되어 진다. d 비트 구성(configuration) 카운터는 디코더에 대해 가능한 2^d개의 입력 조합을 모두 제공하기 위해서 업카운팅 동작을 한다. 구성 카운터는 매 구성 시작 시에 각 구성에서 인가할 테스트 패턴의 수에 해당하는 이진 값으로 초기화되는 패턴 카운터(pattern counter)에 의해서 트리거된다.

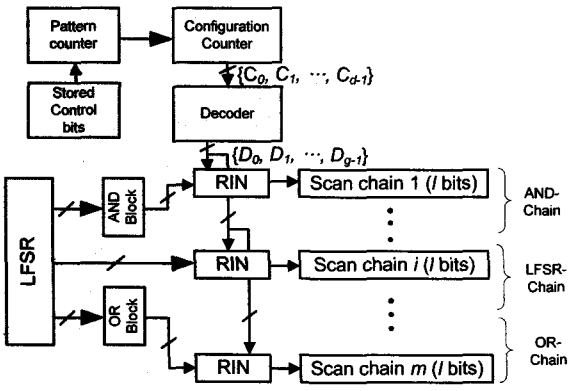


그림 4. 제안하는 내장된 자체 테스트 아키텍쳐
Fig. 4. Proposed logic BIST architecture.

Ⅲ. 하드웨어 합성 알고리즘

각각의 구성에서 가능한 많은 수의 테스트 큐브를 생산하기 위해서, RIN 블록은 AND 블록, OR 블록, 그리고 LFSR의 출력들 중에 가장 적당한 출력을 AND-Chain, OR-Chain, 그리고 LFSR-Chain에 각각 연결해야 하며, 이것은 그림 5에 나타낸 반복적인 시뮬레이션 절차에 의해서 결정된다. AND 블록에 사용된 AND 게이트의 수와 OR 블록에 사용된 OR 게이트의 수는 M_{AND}와 M_{OR}의 파라미터로 나타내기로 한다. 또한 구성 i의 시뮬레이션 절차동안 스캔 체인 $x(y, z)$와 연결가능한 AND(LFSR, OR) 블록의 출력을 나타내기 위해 $ConnAND_x(i)(ConnLFSR_y(i), ConnOR_z(i))$의 기호를 사용한다. 시뮬레이션 절차는 다음과 같다.

```
Set i = 1, j = 0;                                                    ·····(a)
while (T_D ? empty){
  Initialize connection configuration for each scan chain;           ·····(b)
  while (j < MaxSkipPattern+1 ) {
    Obtain the outputs of pattern generator for next l clock cycle;  ·····(c)
    Find matched test cube with current connection configurations;   ·····(d)
    if (match_success) {
      remove matched test cube; reduce connection configuration;     ·····(e)
      j = 0;
    }
    else j = j +1;
  }
  pattern simulation;                                                ·····(g)
  i = i + 1;                                                         ·····(h)
}
```

그림 5. CRIN 합성 알고리즘
Fig. 5. CRIN synthesis algorithm.

(a) i=1로 설정한다.

(b) $ConnAND_x(i)$ = {1, 2, , M_{AND}}, $ConnLFSR_y(i)$ = {1, 2, , L}, 그리고 $ConnOR_z(i)$ = {1, 2, , M_{OR}} ($0 \le x \le N_{AND}$, $N_{AND} < y < N_{AND}+N_{LFSR}$, $N_{AND}+N_{LFSR} \le z \le N_{AND}+N_{LFSR}+N_{OR}$)로 설정한다. 이것은 AND-Chian, LFSR-Chain, 그리고 OR-Chain의 각 스캔 체인이 각각 AND 블록, LFSR, 그리고 OR 블록의 임의의 출력과 연결될 수 있음을 의미한다.

(c) LFSR을 l 클록동안 구동하여, M_{AND} l-비트 벡터 $\{O_{(AND)a} \mid a = 1, 2, , M_{AND}\}$, M_{OR} l-비트 벡터 $\{O_{(OR)b} \mid b = 1, 2, M_{OR}\}$, and L l-비트 벡터 $\{O_{(LFSR)c} \mid c = 1, 2, , L\}$를 얻는다. $O_{(AND)a}$, $O_{(LFSR)b}$, 그리고 $O_{(OR)c}$ 는 AND 블록, LFSR, 그리고 OR 블록의 각각 a번째, b번째, 그리고 c번째 출력단에 대한 l 클록 동안의 출력 스트림을 의미한다.

(d) 현재의 연결가능 구성 $ConnAND_x(i)$, $ConnLFSR_y(i)$, 그리고 $ConnOR_z(i)$의 조건하에서, 각각 $O_{(AND)a}$, $O_{(LFSR)b}$, 그리고 $O_{(OR)c}$의 비트 시퀀스로 생산가능한 테스트 큐브 C^*을 찾는다.

C^*은 m개의 스캔 체인에 대해서 스캔 체인 테스트 큐브 $\{t_1^*, t_2^*, , t_m^*\}$의 형태로 표현된 테스트 큐브이다. 이때 '생산가능'은 다음의 3가지 조건이 동시에 만족할 때를 의미한다.

1) $1 \le j \le N_{AND}$일 때: 모든 j에 대해서 $O_{(AND)a}$에 의해 t_j^*가 커버되는 $a \in ConnAND_i(i)$가 존재
2) $N_{AND} < j < N_{AND}+N_{LFSR}$ 일 때: 모든 j에 대해서, $O_{(LFSR)b}$에 의해 t_j^*가 커버되는 $b \in ConnLFSR_i(i)$가 존재
3) $N_{AND}+N_{LFSR} \le j \le N_{AND}+N_{LFSR}+N_{OR}$ 일 때: 모든 j에 대해서, $O_{(OR)c}$ 에 의해 t_j^* 가 커버되는 $c \in ConnOR_i(i)$이 존재

(e) 만일 과정 (d)에서 3가지 조건을 동시에 만족하는 어떤 테스트 큐브도 발견하지 못했다면, 과정 (f)로 바로 이동한다. 만일 과정 (d)에서 3가지 조건을 동시에 만족하는 테스트 큐브가 존재한다면, C^*을 T_D에서 제거하고, 모든 스캔 체인에 대해 t_j^*를 커버하지 못하는 현재 연결가능 구성 $ConnAND_i(i)$, $ConnLFSR_i(i)$, $ConnOR_i(i)$의 원소를 삭제한다.

(f) 만일 이전 $MaxSkipPattern$+1의 과정 동안, 과정 (d)에서 최소한 하나의 테스트 큐브를 발견했다면, 과정 (c)로 이동한다. $MaxSkipPattern$은 전체 테스트 시간을 제한하기 위해서, 두 개의 결정론적 테스트 큐브의 생성 사이에 허락되는 의사무작위패턴의 최대수를 나타내는 파라미터이다.

(g) 현재의 구성 i에 대한 시뮬레이션 과정을 종료한다. 현재의 구성 i동안 생산된 테스트 패턴을 이용하여, T_D안의 생산 가능한 테스트 큐브 C^*을 제거한다.

(h) i를 1 증가시키고 과정 (b)로 이동한다. 이 반복적인 과정은 모든 테스트 큐브가 생산되어 T_D가 아무런 원소도 포함하지 않을 때까지 지속된다.

예제의 하드웨어 합성 절차를 그림 6에 나타내었다. 테스트 대상 회로는 6개의 스캔 체인을 가지고 있고 각 스캔 체인은 4비트의 길이를 갖는다. 의사무작위패턴은 x^4+x+1의 특성다항식을 갖는 LFSR에 의해서 생성된다. $MaxSkipPattern$ 파라미터는 0으로 설정되었다. AND 블록과 OR 블록은 각각 4개의 AND 게이트와 OR 게이트로 구성되었으며, 각 블록의 입력들은 (1, 4), (1, 3), (2, 4), (1, 2)의 구성으로 LFSR의 출력과 연결되었다. 여기서 괄호안의 숫자는 LFSR의 스테이지를 나타낸다. 즉 (1, 4)는 AND 게이트의 하나의 입력이 LFSR의 첫 번째 스테이지와 또 하나의 입력은 LFSR의 4번째 스테이지와 연결되었음을 의미한다.

초기의 테스트 큐브 세트 T_D와 본 논문에서 제안된 스캔 셀 재배치 알고리즘에 의해서 얻은 재구성된 테스트 큐브 세트 T_D'를 그림 6.(A)에 나타내었다. 그림 6.(B)는 T_D'가 6개의 테스트 큐브 C_1, C_2, , C_6를 포함하고 있으며, 각 테스트 큐브 C_i는 t_1,..., t_4의 스캔 체인 테스트 큐브로 표현됨을 나타낸다. 이 예에서 6개의 스캔 체인은 각각 2개의 AND-Chain, 2개의 LFSR-Chain, 그리고 4개의 OR-Chain으로 구성된다. 그림 6.(C)에서와 같이 패턴생성기의 출력은 테스트 패턴 p_i, i=1, 2... 를 형성한다. 각 테스트 패턴은 12개의 4-비트 벡터로 구성되며, 처음 4개의 4-비트 벡터는 LFSR의 출력, 그 다음 4개의 4-비트 벡터는 AND 블록의 출력, 그리고 나머지 4-비트 벡터들은 OR 블록의 출력을 나타낸다. 패턴생성기와 각 스캔 체인간의 최적 연결선을 선택하는 과정을 그림 6.(D)에 나타내었다.

$Init$은 모든 연결가능 구성 $ConnAND_i(1)$, $ConnLFSR_i(1)$, $ConnOR_i(1)$이 (1, 2, 3, 4)로 설정되는 초기화 과정이다. 여기서 $ConnAND_i(j)$의 (1, 2, 3, 4)와 $ConnLFSR_i(j)$의 (1, 2, 3, 4)는 서로 다른 의미를 갖고 있음을 주의해야 한다. AND-Chain의 테스트 패턴은 AND 블록으로부터 공급받기 때문에 $ConnAND_i(j)$의 (1, 2, 3, 4)는 AND 블록의 스테이지를 의미한다. 하지만 LFSR-Chain의 테스트 패턴은 LFSR로부터 공급받기 때문에, $ConnLFSR_i(j)$의 (1, 2, 3, 4)는 LFSR의 스테이지를 나타낸다. 이와 유사하게 $ConnOR_i(j)$의 (1, 2, 3, 4)는 OR 블록의 스테이지 정보를 표시한다. 과정 (a)에서 테스트 패턴 p_1으로 테스트 큐브 C_1을 생산가능하고, 각 스캔 체인에 대해 연결가능 구성을 나타내었다. 과정 (c)에서 어떤 테스트 큐브도 p_3로는 생산 불가능하다. $MaxSkipPattern$이 0으로 설정되었기 때문에, 현재의 구성에 대한 시뮬레이션 절차는 종료된다. 두 번째 구성의 처음 절차인 과정 (d)에서 각 스캔 체인에 대한 연결가능 구성은 (1, 2, 3, 4)로 초기화되고, 테스트 큐브 C_2가 테스트 패턴 p_4를 이용하여 생산가능하다. 과정 (e)에서 테스트 패턴 p_5로는 남아있는 어떤 테스트 큐브도 생산 불가능하기 때문에 두 번째 구성에 대한 시뮬레이션 절차를 종료한다. 두 번째 구성 동안 생산된 테스트 패턴 p_4와 p_5를 이용하여 T_D'에 남아 있는 테스트 큐브를 생산가능한지 조사한다. 이 경우 C_5와 C_6가 생

(A) Scan cell reordering :

T_D

C_1 : 11XXXXX01X0X0XXXXX10000X
C_2 : XX11X01X10XXXXXXXX0XXX01
C_3 : 01011XXXXXX1XXXXXXX00XXX
C_4 : 1XX0X01XXXXXX10X1XXXXXXX
C_5 : XXXXX01X1XX10XXXXXXXX01
C_6 : XXXXX01XX0X1XX1XXXXXXXX1

→

T_D'

C_1 : 00X00X0010XXX1XX1XXXXXX1
C_2 : 0XXXXXXXXX01X0101XXX11XX
C_3 : XXX00XXX0XX0XX1XXXX1XX11
C_4 : XXXXXXXX1X0X0X0XX11XX1XX
C_5 : 0XXXXX0XXX0XXXX1XX111XX
C_6 : XXXXXXXXXX0X1XX0XXX111XX

(B) Scan chain test cubes :

		C_1	C_2	C_3	C_4	C_5	C_6
AND - Chains	t_1 :	00X0	0XXX	XXX0	XXXX	0XXX	XXXX
	t_2 :	0X00	XXXX	0XXX	XXXX	XX0X	XXXX
LFSR - Chains	t_3 :	10XX	XX01	0XX0	1X0X	XX0X	XX0X
	t_4 :	X1XX	X010	XX1X	0X0X	XXXX	1XX0
OR - Chains	t_5 :	1XXX	1XXX	XXX1	X11X	1XX1	XXX1
	t_6 :	XXX1	11XX	XX11	X1XX	11XX	11XX

(C) Outputs of pattern generator :

			p_1	p_2	p_3	p_4	p_5	p_6
LFSR	1	x_1	0110	0100	0111	1010	1100	1000
	2	x_2	1011	0010	0011	1101	0110	0100
	3	x_3	0101	1001	0001	1110	1011	0010
	4	x_4	1010	1100	1000	1111	0101	1001
AND-Block	1	$x1$ & x_4	0010	0100	0000	1010	0100	1000
	2	$x1$ & x_3	0100	0000	0001	1010	1000	0000
	3	$x2$ & x_4	1010	0000	0000	1101	0100	0000
	4	$x1$ & x_2	0010	0000	0011	1000	0100	0000
OR-Block	1	$x_1 \mid x_4$	1110	1100	1111	1111	1101	1001
	2	$x_1 \mid x_3$	0111	1101	0111	1110	1111	1010
	3	$x_2 \mid x_4$	1011	1110	1011	1111	0111	1101
	4	$x_1 \mid x_2$	1111	0110	0111	1111	1110	1100

(D) Pattern matching procedure

	Init.)	*(a) p_1: C_1*	*(b) p_2 : C_4*	*(c) p_3 : None* *End of confi. 1* *Pattern simulation : None* *Init.*
$ConnAND_1(1)$:	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
$ConnAND_1(2)$:	(1, 2, 3, 4)	(2, 3, 4)	(2, 3, 4)	(1, 2, 3, 4)
$ConnLFSR_1(1)$:	(1, 2, 3, 4)	(2, 3)	(4)	(1, 2, 3, 4)
$ConnLFSR_1(2)$:	(1, 2, 3, 4)	(1, 3)	(1)	(1, 2, 3, 4)
$ConnOR_1(1)$:	(1, 2, 3, 4)	(1, 2)	(1)	(1, 2, 3, 4)
$ConnOR_1(2)$:	(1, 2, 3, 4)	(1, 2)	(1, 2)	(1, 2, 3, 4)

(d) p_4 : C_2	*(e) p_5 : None* *End of confi. 2* *Pattern simulation : C_5, C_6* *Init.*	*(f) p_6 : C_3*
(4)	(1, 2, 3, 4)	(1, 2, 3, 4)
(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
(2)	(1, 2, 3, 4)	(2, 3)
(1)	(1, 2, 3, 4)	(3)
(1, 2, 3, 4)	(1, 2, 3, 4)	(2)
(1, 2, 3)	(1, 2, 3, 4)	(2)

그림 6 CRIN 합성 절차의 예
Fig. 6. An example of CRIN synthesis.

산가능 하다. 마지막 테스트 큐브 C_3가 3번째 구성에서 p_6를 이용하여 생산가능하다. 결국 이 예제에서 6개의 테스트 큐브를 생산하기 위한 총 구성 수는 3개이며 인가해야 할 총 테스트 패턴의 수는 6개이다. 이 예제에 대한 연결 구성은 {$ConnAND_1(1)$, $ConnAND_1(2)$, $ConnAND_1(3)$} = {1, 4, 1}, {$ConnAND_2(1)$, $ConnAND_2(2)$, $ConnAND_2(3)$} = {2, 1, 1}, {$ConnLFSR_1(1)$, $ConnLFSR_1(2)$, $ConnLFSR_1(3)$} = {4, 2, 2}, {$ConnLFSR_2(1)$, $ConnLFSR_2(2)$, $ConnLFSR_2(3)$} = {1, 1, 3}, {$ConnOR_1(1)$, $ConnOR_1(2)$, $ConnOR_1(3)$} = {1, 1, 2}, 그리고 {$ConnLFSR_2(1)$, $ConnLFSR_2(2)$, $ConnLFSR_2(3)$} = {1, 1, 2}이며 패턴 생성기와 각 스캔

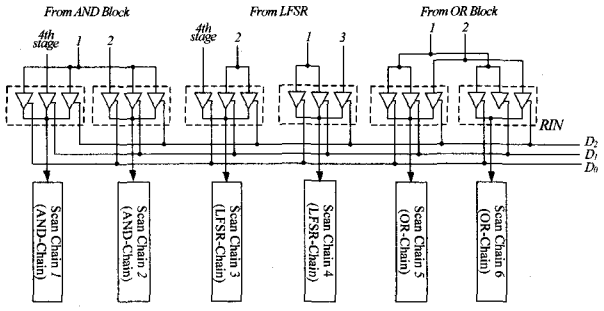


그림 7. CRIN 구성의 예
Fig. 7. An example of CRIN configuration.

체인과의 하드웨어 구성을 그림 7에 나타내었다.

Ⅳ. 실험 결과

이번 장에서는 ISCAS'89 벤치마크 회로에 대해 본 논문에서 제안한 CRIN 내장된 자체 테스트 기법의 효율성을 증명한다. 2장에서 제안한 스캔 셀 재배치 과정 동안 T_{init} = 5.0, T_{low} = 0.1, K_t = 0.97, 그리고 IPT = 500의 파라미터 값으로 설정하였다. 또한 본 실험에 사용된 테스트 큐브는 회로내 모든 고착고장의 검출을 목적으로 다이나믹 패턴 압축(dynamic compaction) 과정없이 Synopsys사의 TetraMax ATPG 프로그램을 이용하여 산출하였다. LFSR은 $x^{64}+x^4+x^3+x+1$의 특성다항식을 가지며, LFSR의 씨드는 랜덤한 값으로 설정하였다. AND 블록과 OR 블록은 각각 64개의 2-입력 AND 게이트들과 64개의 2-입력 OR 게이트들로 구성되었으며, 각 블록 내 게이트와 연결될 LFSR의 스테이지의 선정은 스테이지 사이의 거리가 가장 먼 한 쌍의 스테이지를 먼저 선출하는 방식을 사용하였다. 즉 괄호 안의 숫자가 LFSR의 스테이지를 의미할 때 (1, 64), (1, 63), (2, 64), (1, 62), (2, 63), (3, 64) ...의 구성을 갖는다. 하드웨어 오버헤드의 산출을 위해서는 [14]에서 사용한 방법과 동일하게 게이트등가수치(gate equivalent value)를 사용하였다. n-입력 NAND 또는 NOR 게이트에 대해서는 0.5n, 그리고 인버터에 대해서는 0.5의 게이트등가수치를 사용하였다. 또한 트랜스미션 게이트에 대해서는 0.5 그리고 플립플롭에 대해서는 4의 게이트등가수치를 적용하였다. 실험의 편의를 위해서 실험에 사용된 벤치마크 회로들은 모든 스캔 체인의 길이가 동일한 균형(balanced) 스캔 체인을 갖는다고 가정한다.

테스트 대상 회로가 32개의 스캔 체인을 갖고 *MaxSkipPattern* 파라미터가 5000으로 설정되었을 때, 표 2는 기존의 RIN 방법을 사용한 실험 결과를 의미하며, 표 3은 본 논문에서 제안한 CRIN 기법을 사용한 실험 결과를 의미한다. 'No. of configurations' 열은 테스트 큐브 세트 T_D를 생산하는데 필요한 총 구성 수를 나타낸다. 테스트 대상 회로에 인가한 총 패턴의 수는 'No. of BIST patterns' 열에 표시되어 있으며, 이는 100% 고장검출율을 달성하는데 필요한 테스트 시간의 판단 근거로 사용될 수 있다. 또한 'Scan chain groups' 열에 나타낸 괄호 안의 숫자들은 본 논문에서 제안한 새로운 스캔 셀 재배치 알고리즘에 의해서 얻어진 AND-Chain, LFSR-Chain, 그리고 OR-Chain의 수를 각각 나타낸다. 'Storage requirement' 열의 숫자는 각

표 2. 기존의 RIN[14][15]에 의한 실험 결과 (32개 스캔 체인, *MaxSkipPattern* = 5000)
Table 2. Experimental results using the previous RIN[14][15] (32 scan chains, *MaxSkipPattern* = 5000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Storage requirement (bits)	Hardware overhead (percentage)	Encoding efficiency
s5378	1285	7	214	16	257188	272	11.24%	44.54
s9234	1557	8	247	32	517733	544	10.95%	35.55
s13207	3221	22	700	10	205859	170	2.30%	140.20
s15850	3257	20	611	36	588317	612	6.57%	45.71
s35932	7477	56	1763	4	27055	60	0.34%	523.20
s38417	7691	52	1664	243	2065818	4374	17.52%	19.96
s38584	12216	46	1464	20	518498	360	1.50%	247.47
Average	□	□	□	52	597210	913	7.20%	150.95

표 3. 제안된 CRIN에 의한 실험 결과 (32개 스캔 체인, MaxSkipPattern = 5000)
Table 3. Experimental results using the proposed CRIN (32 scan chains, MaxSkipPattern = 5000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Scan chain groups	Storage requirement (bits)	Hardware overhead (percentage)	Encoding efficiency	Test time Reduction (percentage)	Storage requirement reduction (percentage)
s5378	1285	7	214	11	186503	(11, 18, 3)	187	15.05%	64.78	27.48%	31.25%
s9234	1557	8	247	16	296770	(12, 12, 7)	272	9.20%	71.10	42.68%	50.00%
s13207	3221	22	700	6	157895	(18, 11, 3)	108	3.59%	220.69	23.30%	36.47%
s15850	3257	20	611	25	356231	(16, 12, 4)	450	6.50%	62.17	39.45%	26.47%
s35932	7477	56	1763	3	19983	(17, 13, 2)	45	1.06%	697.60	26.14%	25.00%
s38417	7691	52	1664	133	1225964	(12, 17, 3)	2394	10.54%	36.47	40.65%	45.27%
s38584	12216	46	1464	8	265469	(9, 19, 4)	144	1.39%	618.68	48.80%	60.00%
Average	□	□	□	29	358402	(14, 15, 4)	514	6.76%	253.07	35.50%	39.21%

표 4. 기존의 RIN[14][15]에 의한 실험 결과 (32개 스캔 체인, MaxSkipPattern = 1000)

Table 4. Experimental results using the previous RIN[14][15] (32 scan chains, MaxSkipPattern = 1000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Storage requirement (bits)	Hardware overhead (percentage)	Encoding efficiency
s5378	1285	7	214	19	73223	285	13.31%	42.51
s9234	1557	8	247	39	117432	585	13.35%	33.06
s13207	3221	22	700	12	62201	192	2.69%	124.14
s15850	3257	20	611	49	140379	735	8.81%	38.06
s35932	7477	56	1763	3	13982	45	0.25%	697.60
s38417	7691	52	1664	262	492428	4192	19.00%	20.83
s38584	12216	46	1464	25	174695	400	1.86%	222.73
Average	□	□	□	58	153477	919	8.47%	168.42

표 5. 제안된 CRIN에 의한 실험 결과 (32개 스캔 체인, MaxSkipPattern = 1000)

Table 5. Experimental results using the proposed CRIN (32 scan chains, MaxSkipPattern = 1000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Scan chain groups	Storage requirement (bits)	Hardware overhead (percentage)	Encoding efficiency	Test time Reduction (percentage)	Storage requirement reduction (percentage)
s5378	1285	7	214	14	66831	(11, 18, 3)	210	17.05%	57.69	8.73%	26.32%
s9234	1557	8	247	21	86700	(12, 12, 7)	336	10.88%	57.56	26.17%	42.56%
s13207	3221	22	700	9	54524	(18, 11, 3)	153	4.29%	155.78	12.34%	20.31%
s15850	3257	20	611	32	103137	(16, 12, 4)	512	7.72%	54.64	26.53%	30.34%
s35932	7477	56	1763	3	11553	(17, 13, 2)	45	1.06%	697.60	17.37%	0.00%
s38417	7691	52	1664	152	352559	(12, 17, 3)	2432	11.92%	35.90	28.40%	41.98%
s38584	12216	46	1464	12	91444	(9, 19, 4)	204	1.69%	436.72	47.66%	49.00%
Average	□	□	□	35	109535	(14, 15, 4)	556	7.80%	213.70	23.89%	30.07%

구성에서 생산해야 할 패턴 숫자의 정보를 저장하는데 필요한 저장 공간의 양을 나타내다. 인코딩 효율(encoding efficiency)은 테스트 큐브 세트에 포함된 규정비트와 필요한 저장공간 비트의 비로 계산되었다. 표 2에 나타낸 기존 RIN의 실험 결과와 [14][15]에 표시된 실험 결과가 다른 이유는 실험에 사용된 테스트 큐브 세트와 LFSR의 특성다항식, 그리고 LFSR의 씨드가 서로 다르기 때문이다. 본 실험에서 사용된 테스트 큐브에 포함된 규정비트의 수는 [14][15]의 실험에서 사용된 테스트 큐브 세트에 포함된 규정비트의 수보다 많다. 본 실험에서 사용된 테스트 큐브 세트의 규정비트는 총 291066개인데 반해, [14][15]의 실험에 사용된 테스트 큐브의 수는 총 132812개이다.

내장된 자체 테스트의 효율은 완전한 혹은 높은 고장검출율을 달성하는데 필요한 테스트 길이와 하드웨어 오버헤드에 의해서 특징 지워진다. 본 논문에서 제안한 CRIN은 새로운 스캔 셀 재배치 알고리즘과 무작위패턴 저항패턴의 생산확률을 높일 수 있는 전용의 하드웨어 블록을 사용한다. 결과적으로 100% 고착 고장검출율을 달성하는데 필요한 총 구성의 수는 기존 연구의 반 정도인 29개로 줄일 수 있다. 제안된 CRIN 내장된 자체 테스트 기법은 M_{AND}개의 AND, 그리고 M_{OR}개의 OR 게이트로 구성된 별도의 하드웨어 블록을 필요로 한다. 이 부가적인 하드웨어 블록의 사용에도 불구하고, 총 구성의 수를 효과적으로 감축하여 하드웨어 오버헤드는 오히려 7.2%에서 6.75%로 줄어든 것을 실험결과는 보여준다. 기존의 연구 결과에 비해, 제안된 CRIN 내장된 자체 테스트 기법은 테스트 시간을 35% 감축시킬 수 있으며, 저장공간을 39% 절약할 수 있다.

테스트 대상 회로가 32개의 스캔 체인을 가지며, *MaxSkipPattern*이 1000으로 설정된 경우 표 4는 기존 RIN에 대한 실험 결과를 표 5는 제안된 CRIN에 의한 실험 결과를 보여준다. 예상한 바와 같이 *MaxSkipPattern*이 5000인 경우의 결과보다, 필요한 총 구성의 수는 증가하였으나, 'No' of BIST patterns' 열의 수치로 평가되는 테스트 시간은 많이 감축되었음을 볼 수 있다. s38417과 s38585의 가장 큰 2개의 회로에 대해서 제안된 CRIN 기법은 기존의 연구 결과와 비교해 하드웨어 오버헤드의 증가 없이, 거의 40%의 테스트 시간 감축과 거의 50%의 저장공간 절약의 효과가 있음을 볼 수 있다.

표 6, 표 7, 표 8 그리고 표 9는 테스트 대상 회로가

표 6. 기존의 RIN[14][15]에 의한 실험 결과 (64개 스캔 체인, MaxSkipPattern = 5000)
Table 6. Experimental results using the previous RIN[14][15] (64 scan chains, MaxSkipPattern = 5000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Storage requirement (bits)	Hardware overhead (percentage)
s13207	3221	11	700	7	362638	119	2.87%
s15850	3257	10	611	19	454698	323	6.42%

표 7. 제안된 CRIN에 의한 실험 결과 (64개 스캔 체인, MaxSkipPattern = 5000)
Table 7. Experimental results using the proposed CRIN (64 scan chains, MaxSkipPattern = 5000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Scan chain groups	Storage requirement (bits)	Hardware overhead (percentage)	Encoding efficiency	Test time Reduction (percentage)	Storage requirement reduction (percentage)
s13207	3221	11	700	6	115221	(37, 21, 6)	108	4.69%	220.69	68.23%	9.24%
s15850	3257	10	611	14	268587	(32, 23, 9)	252	6.78%	111.02	40.93%	21.98%
s35932	7477	28	1763	3	20087	(35, 25, 4)	45	1.27%	697.60	5.67%	0.00%
s38417	7691	26	1664	82	832211	(25, 34, 5)	1476	12.23%	59.16	37.18%	43.84%
s38584	12216	23	1464	7	312856	(25, 33, 6)	126	1.76%	707.06	22.14%	46.15%
Average	–	–	–	22	309792	(29, 27, 6)	401	5.34%	359.10	34.83%	24.24%

표 8. 기존의 RIN[14][15]에 의한 실험 결과 (64개 스캔 체인, MaxSkipPattern = 1000)
Table 8. Experimental results using the previous RIN[14][15] (64 scan chains, MaxSkipPattern = 1000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Storage requirement (bits)	Hardware overhead (percentage)	Encoding efficiency
s13207	3221	11	700	8	101413	128	3.28%	186.20
s15850	3257	10	611	23	104230	345	7.77%	81.09
s35932	7477	28	1763	5	14793	75	0.74%	418.56
s38417	7691	26	1664	173	352950	2768	23.94%	31.55
s38584	12216	23	1464	15	192376	240	2.06%	371.21
Average	–	–	–	44.8	153152	711	7.56%	217.72

표 9. 제안된 CRIN에 의한 실험 결과 (64개 스캔 체인, MaxSkipPattern = 1000)
Table 9. Experimental results using the proposed CRIN (64 scan chains, MaxSkipPattern = 1000).

Circuit	No. of test cubes	Length of scan chain	No. of scan cells	No. of Configurations	No. of BIST patterns	Scan chain groups	Storage requirement (bits)	Hardware overhead (percentage)	Encoding efficiency	Test time Reduction (percentage)	Storage requirement reduction (percentage)
s13207	3221	11	700	6	58314	(37, 21, 6)	102	4.69%	233.67	42.50%	20.31%
s15850	3257	10	611	17	77167	(32, 23, 9)	272	7.64%	102.85	25.96%	21.16%
s35932	7477	28	1763	3	11133	(35, 25, 4)	45	1.27%	697.60	24.74%	40.00%
s38417	7691	26	1664	112	201921	(25, 34, 5)	1792	16.34%	48.73	42.79%	35.26%
s38584	12216	23	1464	10	112697	(25, 33, 6)	170	2.19%	524.06	41.42%	29.17%
Average	–	–	–	30	92246	(29, 27, 6)	476	6.43%	321.38	35.48%	29.18%

64개의 스캔 체인을 포함한 경우에 대한 실험 결과를 의미한다. *MaxSkipPattern* 파라미터 값이 5000으로 설정되었을 경우 표 6은 기존의 RIN에 대한 실험 결과를 그리고 표 7은 제안된 CRIN 기법에 의한 실험 결과를 나타낸다. 또한 *MaxSkipPattern*이 1000으로 설정되었을 경우 표 8은 기존 RIN의 실험 결과를, 표 9는 제안된 CRIN에 의한 실험 결과를 나타낸다. 벤치마크 회로에 대해서 평균적으로 제안된 CRIN 기법은 기존의 연

표 10. 각 구성 당 고정된 수의 테스트 패턴 구성에 대한 실험 결과(32 스캔 체인, 컨피규레이션 당 패턴=1000)
Table 10. Experimental results for a fixed number of patterns per a configuration (32 scan chains, the number of patterns per a configuration = 1000).

				previous work [14][15]			Proposed approach				
	No. of test	Length of scan	No. of scan	No. of	No. of BIST	Hardware overhead	No. of	No. of BIST	Scan chain	Hardware overhead	Test time Reduction
Circuit	cubes	chain	cells	Configurations	patterns	(percentage)	Configurations	patterns	groups	(percentage)	(percentage)
s9234	1557	8	247	40	39043	14.41%	28	27029	(12, 12, 7)	13.75%	30.77%
s13207	3221	22	700	17	16601	4.11%	12	11013	(18, 11, 3)	4.88%	33.66%
s15850	3257	20	611	50	49050	9.41%	36	35129	(16, 12, 4)	8.88%	28.38%
s35932	7477	56	1763	3	2178	0.25%	3	2202	(17, 13, 2)	1.06%	-1.10%
s38417	7691	52	1664	227	226379	16.67%	161	153198	(12, 17, 3)	12.76%	32.33%
s38584	12216	46	1464	39	38039	3.03%	23	22033	(9, 19, 4)	2.62%	42.08%
Average	□	□	□	63	61882	7.98%	44	41767	(14, 14, 4)	7.33%	27.69%

구결과보다 거의 30% 저장공간 절약과 거의 35%의 테스트 시간 감축의 효과가 있다.

지금까지의 실험은 각 구성에서 인가해야 할 패턴의 수를 나타내는 정보가 칩에 저장되는 제어 형태를 가정하였다. 하지만 [14]에서 지적한 바와 같이, 각 구성 마다 고정된 수의 테스트 패턴을 인가한다면 별도의 저장공간은 필요하지 않게 된다. 표 10은 각 구성 마다 고정된 1000개의 테스트 패턴을 인가하는 테스트 제어 형태에 관한 실험 결과를 나타낸다. s35932 벤치마크 회로를 제외한 모든 회로에 대해 제안된 CRIN 기법은 기존의 연구 결과보다 28%-42%의 테스트 시간을 감축하였다. s35932 회로에 대해서는 기존의 연구 결과보다 약간 테스트 시간이 증가한 것을 볼 수 있다. 이것은 s35932 회로가 거의 무작위패턴저항패턴을 가지고 있지 않기 때문이다.

V. 결 론

내장된 자체 테스트 장치의 효율은 완전한 혹은 높은 고장검출율을 달성하는데 필요한 테스트 시간과 하드웨어 오버헤드에 의해서 특징 지워진다. 본 논문은 결정론적 테스트 큐브를 생산하기 위한 clustered RIN 내장된 자체 테스트 기법을 제안하였다. 제안된 내장된 자체 테스트 기법은 주어진 테스트 큐브 세트의 신호확률에 기반을 둔 스캔 셀 재배치 방법과 규정비트가 집중된 스캔 체인 테스트 큐브의 생산확률을 높일 수 있는 전용의 하드웨어 블록을 사용한다. 비록 본 논문에서 제안된 CRIN 기법은 부가적인 하드웨어 블록을 필요로 하지만, 총 구성의 효과적인 감축으로 인해 기존의 연구 결과보다 적은 하드웨어 오버헤드를 필요로 함을 실험을 통하여 증명하였다. 또한 제안된 기법은 100%의 고장검출율을 달성하는데 있어 기존의 연구 결과와 비교해 적은 저장공간과 짧은 테스트 시간을 필요로 함이 실험을 통하여 입증되었다. 제안된 CR IN 내장된 자체 테스트 기법은 적은 저장공간과 하드웨어 오버헤드로 높은 고장검출율과 짧은 테스트 시간을 달성하는 문제에 있어서 효과적인 해결 방법을 제시한다.

참 고 문 헌

[1] P. H. Bardell, W. Mcanney, and J. Savir, *Built-in Test for VLSI: Pseudo-Random Technique*, New York: Wiley, 1987.

[2] V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A tutorial on built-in self-test part 1: principles," *IEEE Design & Test of Computers*, vol. 10, pp. 73-82, Mar. 1993.

[3] V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A tutorial on built-in self-test part 1: applications," *IEEE Design & Test of Computers*, vol. 10. pp. 69-77, June 1993.

[4] A. J. Briers and K. A. E. Totton, "Random Pattern Testability by Fast Fault Simulation," *Proc. of IEEE Int. Test Conf.*, pp. 274-281, 1986.

[5] Y. Savaria, M. Youssef, B. Kaminska, and M. Koudil, "Automatic Test Point Insertion for Pseudo-Random Testing," *Proc. of IEEE Int. Symp. Circuit and Systems*, pp. 1960-1963, 1991.

[6] J. A. Waicukauski, E. Lindbloom, E. B. Eichelberger, and O. P. Forlenza "A Method for Generating Weighted Random Patterns," *IBM Journal of Research and Development*, vol. 33, pp. 149-161, Mar. 1989.

[7] H. S. Kim, J. K. Lee, and S. Kang, "A Heuristic for Multiple Weight Set Generation," *Proc. of IEEE Int. Test Conf.*, pp. 513-514, 2001.

[8] S. Hellebrand, J. Rajski, S. Tarnick, and B.

Courtois, "Built-in test for circuits with scan based on reseeding of multiple-poly-nomial linear feedback shift registers," *IEEE Trans. Computers*, vol. 44, pp. 223-233, Feb. 1995.

[9] C. V. Krishna, A. Jas, and N. A. Touba, "Test vector encoding using partial LFSR reseeding," *Proc. of IEEE Int. Test Conf.*, pp. 885-893, 2001.

[10] E. Kalligeros, X. Kavousianos, and D. Nikolos, "A ROMless LFSR reseeding scheme for scan-based BIST," *Proc. of 11th Asian Test Symp.*, pp. 206-211, 2002.

[11] H. S. Kim, Y. J. Kim and S. Kang, "Test-Decompression Mechanism Using a Variable-Length Multiple-Polynomial LFSR," *IEEE Trans. VLSI Systems*, vol. 11, pp. 687-690, Aug. 2003.

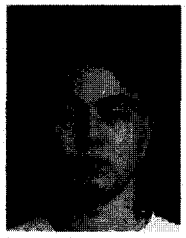
[12] H. J. Wunderlich and G. Kiefer, "Bit-flipping BIST," *Proc. of IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 337-343, 1996.

[13] G. Kiefer and H. J. Wunderlich, "Using BIST control for pattern generation," *Proc. of IEEE Int. Test Conf.*, pp. 347-355, 1997.

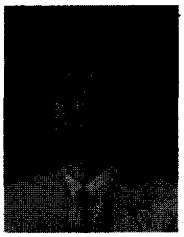
[14] L. Li and K. Chakrabarty, "Deterministic BIST Based on a Reconfigurable Interconnection Network," *Proc. of IEEE Int. Test Conf.*, pp.460-496, 2003.

[15] L. Li and K. Chakrabarty, "Test Set Embedding for Deterministic BIST Using Reconfigurable Interconnect Network," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, pp. 1289-1305, Sept. 2004.

저 자 소 개



송 동 섭(학생회원)
2000년 건국대학교 전기공학과 학사 졸업.
2002년 연세대학교 전기전자 공학과 석사 졸업.
2005년 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.
2005년 현재 연세대학교 전기전자공학과 교수.
<주관심분야 : SoC 설계 및 SoC 테스트>