論 文

고장위치 검출 가능한 BIST용 패턴발생 회로의 설계

正會員 金 大 翊* 正會員 鄭 鎭 泰* 正會員 李 昌 基* 正會員 田 炳 實*

Design of Fault Position Detectable Pattern Generator for Built-In Self Test

Dae Ik Kim*, Jin Tae Jurng, Chang Ki Lee*, Byoung Sil Chon* *Regular Members*

"이 논문은 1992년도 교육부지원 한국학술 진행재단의 자유공모과제 학술연구조성비에 의하여 연구되었음."

要 約

본 논문에서는 RAM의 Built-In Self Test(BIST)를 수행하기 위하여 제안되었던 Column Weight Sensitive Fault(CWSF) 테스트 알고리즘과 비트라인 디코더 고장 테스터 알고리즘에 적합한 패턴발생 회로와 고장위치 검출기를 설계하였다. 패턴발생 회로는 어드레스 발생부와 데이터 발생부로 구성되었다. 또한 어드레스 발생부는 실효 어드레스를 위한 행 어드레스 발생부와 순차 및 병렬 어드레스를 위한 열 어드레스 발생부로 나누어져 있다. 고장위치 검출기는 고장발생의 유, 무와 그 위치를 찾기위해 구성되었다. 설계한 회로들의 검증을 위하여 각 부분 및 전체적인 시뮬레이션을 통하여 동작을 확인하였다.

Abstract

In this paper, we design a pattern generator and a fault position detector to implement the proposed fault test algorithms which are Column Weight Sensitive Fault(CWSF) test algorithm and bit line decoder fault test algorithm for performing the Built-In Self Test(BIST) in RAM. A pattern generator consists of an address generator and a data generator. An address generator is divided into a row address generator for effective address and a column address generator for sequential and parallel addresses. A fault position detector is designed to determine whether fult occurred or not and to find the position of the fault. We verify the implemented circuits by the simulation.

I. 서 론

반도체 기술의 급속한 발전으로 단일 칩당 비트수 즉, 집적도가 증가 되어지고 있다. 특히 메모리의 경우에는 2~3년 주기로 거의 4배의 기억 용량이 증가되는 추세이다. 그러나, 고집적 메모리의 테스트 시간 증가와 미세화된 패턴과 최소치수에 의해 발생되는 더욱 복잡화된 고장 모델, 경제성 유지를 위하여 단순한 테스트방법을 채택할때 야기되는 고장 탐지 능력 저하때문에 테스트 문제는 제조비용 절감에 있어서 커다란 장애 요인이 되고 있다. 따라서, 메모리

*全北大學校 電子工學科
**Dept. of Electronics Engineering, Chonbuk National University
論文番號 : 93-155

용량이 증가함에 따라 복잡한 제조공정의 결함으로 인한 오동작을 검출해 낼 수 있는 테스트 기술이 시장 경쟁력을 높일 수 있는 중요한 요인으로 부각되었다[1].

RAM 테스트시 가장 중요한 요소는 테스트 비용을 줄이기 위해 최적의 테스트 시간내에 높은 고장 검출률을 얻는 것이다. 지금까지 RAM에서의 고장 검출 문제에 대하여 많은 연구가 이루어져 왔으며, 연구된 기존의 고장 모델은 stuck-at 고장[2, 3, 4], coupling 고장[3], adjacent pattern-sensitive 고장[5, 6], pattern-sensitive 고장[7, 8] 등이 있다.

그러나, RAM의 고집적화에 의하여 발생가능하고, 기존의 고장 모델로써 테스트 할 수 없는 새로운 형태의 고장을 검출할 수 있는 고장모델에 대한 연구를 요구하게 되었다.

또한, 메모리 테스트시 외부장비에 의해 행하는 경우에는 메모리 용량이 커짐에 따라 많은 테스트 시간이 요구되어 테스트 비용이 엄청나게 소요된다. 이에 부응하여 최근에는 테스트 회로를 메모리 칩내에 내장한 Built-In Self Test(BIST) 방식이 등장하여 테스트 비용과 시간을 절약하게 되었다[9].

본 논문에서는 기존의 고장모델들을 거의 검출할 수 있으며 테스트하는데 소요되는 시간을 상당히 단축시킬 수 있는 Column Weight Sensitive Fault (CWSF) 테스트 알고리즘과 비트 라인 디코더 고장 테스트 알고리즘[10]을 채택하였다. 그리고 두 알고리즘들을 H/W로 구현할때, BIST에 적용 가능한 패턴 발생 회로를 설계하였고 메모리 테스트 공정에서 셀의 수리(repair)에 적용할 수 있도록 고장의 유, 무와 그 위치를 검출해 주는 고장위치 검출기를 구현하였다.

Ⅱ. 테스트 알고리즘

본 논문에 적용한 테스트 알고리즘은 두 개의 알고리즘[10]으로 구성되어 있으며, 전체 알고리즘의 흐름은 CWSF 테스트를 먼저 수행한후, 디코더의 이상을 검출하는 비트라인 디코더 고장 테스트 알고리즘을 실행하게 된다.

그림1에 알고리즘의 수행 흐름도를 나타내고 있다.

1. Column Weight Sensitive Fault 테스트 알고리즘(알고리즘 1)

CWSF 모델은 테스트 시간을 단축하기 위한 병렬 테스트 기법인 Line Mode Test (LMT)를 적용 가능하게 하고, BIST를 쉽게 구현할 수 있도록 Row/Column Pattern Sensitive Fault(R/CPSF)[11]모듈을 수정 보완하여 정의되었다.

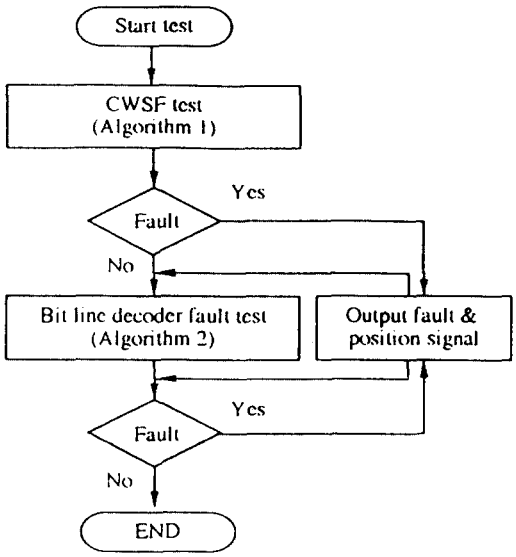


그림 1. 알고리즘 흐름도
Fig 1. Flowchart of algorithms

R/CPSF 모델은 기본셀을 포함하는 행(row)과 열(column)에 위치하고 있는 이웃셀들의 가중치(weight)가 기본셀에 영향을 줄 수 있다는 개념으로부터 정의된 반면에 CWSF 모델은 열 이웃셀에 대한 가중치만을 고려하였다. 따라서 CWSF 모델은 기존의 R/CPSF 모델 보다 같은 행을 갖는 이웃셀들에 대한 고장 검출률은 줄었지만, 리프레쉬, 읽기/쓰기

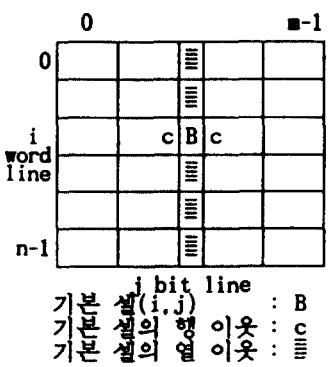


그림 2. CWSF 모델
Fig 2. CWSF model

(read/write) 회로를 공유하는 열 셀들에 대한 전기적 영향은 R/CPSF와 마찬가지로 고려되었다. 그림2에 정의한 CWSF 모델의 이웃 셀들에 도시되어 있다.

CWSF 테스트 알고리즘은 병렬 읽기/쓰기 동작에 의하여, 그림3의 메모리 배열에서 한 워드 라인에 연결된 모든 우수(even) 비트 셀(모든 기수(odd) 비트 셀)들이 동시에 선택되기 때문에 n×m 배열의 테스트는 n×2 배열의 테스트와 동일하다.

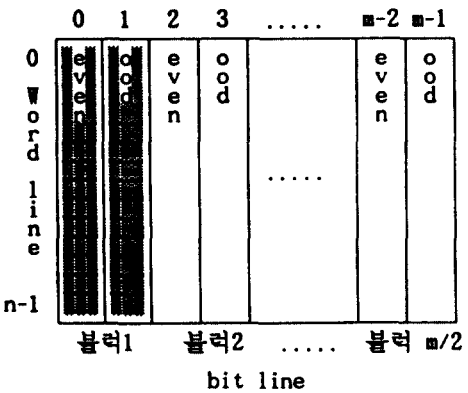


그림 3. 테스트 블럭
Fig 3. Test block

또한, n×2의 모든 테스트 블럭들이 동시에 테스트 되기 때문에 한 테스트 블럭에 대한 CWSF를 검출하기 위한 테스트 패턴만을 발생시키면 된다.

CWSF 테스트 알고리즘의 테스트 절차는 다음과 같다.

Step 1. 메모리 셀 배열을 1로 초기화 한다.
Step 2. 셀값 0과 1에 대하여 4개의 코너 셀을 테스트 한다.
Step 3. 셀값 0에 대하여 넌-코너(non-corner) 셀을 테스트한다.
Step 4. 메모리 셀 배열을 0로 초기화 한다.
Step 5. 셀값 0에 대하여 넌-코너 셀을 테스트한다.

테스트 절차에서 (0, 0)을 제외한 기본 셀은 항상 행 어드레스와 열 어드레스가 우수 셀을 선택한다. 그 이유는 한 기본 셀이 테스트되면 동시에 기본 셀에 대한 코너셀들이 테스트 되어지며, 코너 셀들은 행 어드레스나 열 어드레스 또는 양쪽 어드레스가 모두 기수 셀이기 때문에 기수 셀에 대한 특별한 테스트 절차가 필요하지 않기 때문이다.

(0, 0) 셀에 대한 테스트를 살펴보기 위해 다음을 정의한다.

$(i_e, 0)$의 코너셀 : $(i_e, 0)$을 테스트할 때 동시에 테스트 되는 셀들로, 다음의 3셀이 정의된다. $\{([i_e-1] \bmod n, 0), (i_e, 1), ([i_e-1] \bmod n, 1)\}$ (v, r, c) : v는 셀 값, r과 c는 행과 열의 가중치를 의미한다.

$[R, C]_v$: 셀값 v에 대해 행 가중치와 열 가중치를 각각 0부터 R까지, 0부터 C까지 가진 경험이 있음을 나타낸다.

$[R', C']_v$: 셀값 v에 대해 행 가중치와 열 가중치를 각각 R부터 n−1까지, C부터 n−1까지 가진 경험이 있음을 나타낸다.

기본 셀이 (0, 0)일때 그의 코너셀은 (0, 1), (n−1, 0), (n−1, 1)이 되며 이들 셀들이 동시에 테스트된다.

첫번째 단계로 1로 초기화되어 있는 메모리 셀들에 대해 0번째 행의 모든 셀에 대하여 1 값을 읽고 0 값을 쓴다. 이때 0번째 행의 모든 셀들은 $(0, 0, n-1)_0$ 상태를 갖게 된다. 계속해서 n−1번째 행에 도달할 때까지 이 동작을 수행한다. 이 결과로 0번째 행의 모든 셀들은 $[0, n-1]_0$ 상태를 경험한다. 일반적으로 r번째 행 셀들은 $[0, n-1-r]_0$과 $[1', (n-1-r)']_1$ 상태를 접하게 된다(단, $0 \le r \le n-1$). 여기까지 2n의 읽기와 2n의 쓰기 동작이 이루어진다. 위 단계가 끝난 후 메모리 배열의 모든 셀들은 0 값을 갖게 된다.

두번째 단계로 0번째 행 부터 위 단계를 반복한다. 이때 읽고 쓰는 데이터는 첫번째 단계와 반대값을 갖게되며, r번째 행은 $[0, r]_0$와 $[1', n-1-r]_1$ 상태를 접하게 된다. 이 단계 후 메모리 셀들은 모두 1로 되며, 지금까지의 읽기와 쓰기의 수는 모두 8n이 된다. 첫번째와 두번째 단계는 (0, 0) 셀에 대해 행 가중치를 0으로 고정시키고 열 가중치를 0부터 n−1까지 변화시킨 과정이었다.

세번째 단계에서는 (0, 0)에 행 가중치를 1로 주기 위해 첫번째 열의 모든 셀에 대하여 1값을 읽고 0값을 쓴다. 그후에 위의 첫번째와 두번째의 단계를 반복 수행한다. 이 결과 0번째 행 셀들 중 (0, 0)셀은 $[1, n-1]_0$, $[0, 7]_1$ 상태를 접하게 되며, (0, 1)은 $[0, n-1]_1$, $[1', n-1]_0$ 상태를 접하게 된다.

네번째 단계로 0번째 열의 모든 셀에 대해 0 값을 읽고 1 값을 쓰는 동작을 수행한다.

마지막 단계로 첫번째 단계를 한번 더 수행 함으로써 (0, 0)에 대한 테스트가 끝난다.

(0, 0)을 제외한 셀들을 테스트하기 위하여 다음을 정의한다.

P값 : (i_e, 0)에 대한 P값은 (i_e mod p) = 0을 만족하고 n보다 작거나 같은 2의 거듭제곱(Power) 중 가장 큰 수.

예) 8×2 메모리 배열에서 (0, 0)에 대한 p값은 각각 8이고, (6, 0)에 대한 p값은 각각 2이다.

부배열(Subarray) : (i_e, 0)에 대한 부배열은 $p_e \times 2$의 크기를 갖는 배열로 (i_s, j_s)들의 집합이다.

$$i_s = [i - (\frac{p}{2}) + ((\frac{p}{2}) + k) \bmod p] \bmod n, \quad (0 \le k \le p-1)$$

$$j_s = 0, 1$$

각 우수 셀들에 대한 테스트는 기본 개념에 있어 (0, 0)을 테스트 하는 단계들과 매우 흡사하다. (i_e, 0)에 대해 p_e 값을 계산하고, (i_e, 0)를 기본 셀로 하는 부배열을 발생시킨다.

이 부배열의 크기는 $p_e \times 2$이며, 행 가중치를 0 또는 1 값 중 한 값에 먼저 고정시키고, 각각의 행 가중치에 대해 열 가중치를 n-1-p에서 n-1까지 또는 n-1에서 n-1-p 값까지 변화시킴으로써 테스트를 수행한다.

위의 테스트 과정을 수행한 후 메모리 배열의 모든 셀들은 테스트 전과 같이 1로 초기화되며, 이 테스트 결과로 (i_e, 0) 뿐만아니라 (i_e, 0)의 코너셀들도 동시에 셀값 0에 대하여 테스트 된다. 셀값 1에 대한 (i_e, 0)의 테스트는 메모리 배열의 모든 셀들을 0으로 초기화 시킨 후 앞에서 언급한 테스트 과정을 똑같이 수행한다.

2. 비트라인 고장 테스트 알고리즘
(알고리즘 2)

CWSF 테스트 알고리즘은 비트라인 디코더가 항상 병렬 테스트 모드로 동작하기 때문에 정상 모드에서 올바른 동작을 하는지 점검할 수 없다. 따라서, 비트라인 디코더를 정상 모드로 전환한 후 한 워드 라인에 대한 모든 비트라인 셀들에 데이터 0, 1을 순차적으로 읽기/쓰기 동작을 행함으로써 테스트 한다.

테스트 절차는 다음과 같다.

Step 1. 임의의 한 한 워드 라인을 선택하여 셀 배열을 0으로 초기화 한다.

Step 2. 셀 배열의 최하위 비트로 부터 최상위 비트까지 0 값을 읽고 1 값을 쓴다.

Step 3. 셀 배열의 최상위 비트로 부터 최하위 비트까지 1 값을 읽고 0 값을 쓴다.

이 테스트 알고리즘으로 검출할 수 있는 고장들은 디코더 stuck-at 고장, 비트라인의 잘못된 선택 고장, 다중 접근 고장, non-access 고장 등이 있다. 그림4에 이 알고리즘의 기본개념이 도시되어 있다.

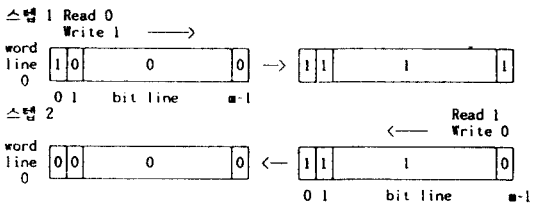


그림 4. 알고리즘 2의 테스트 스텝
Fig 4. Test Steps of Algorithm 2

Ⅲ. BIST 회로의 패턴 발생부 설계 및 시뮬레이션

본 논문에 적용한 두 알고리즘에 적합한 BIST RAM의 블럭다이어그램을 그림5에서 보여주고 있다. 여기에서 BIST 회로는 제어부, 패턴발생부, 고장 및 위치 검출부로 이루어져 있다.

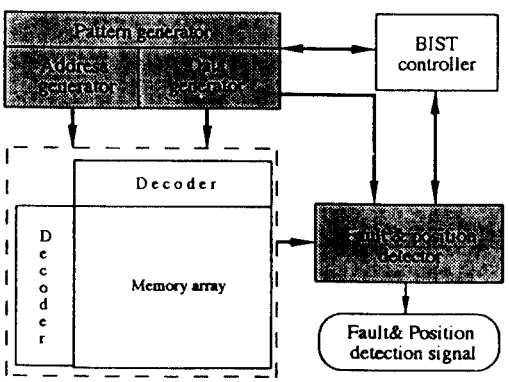


그림 5. BIST RAM의 블럭 다이어그램
Fig 5. Block diagram of BIST RAM

각 알고리즘에 적합한 패턴을 발생시키기 위하여 어드레스 발생부와 데이터 발생부가 필요하다. 어드레스는 행 어드레스와 열 어드레스로 나뉘어 발생된다. 각각의 어드레스 발생회로는 카운터로 구현되었으며, 패턴 발생부의 전체 구조는 그림6과 같다.

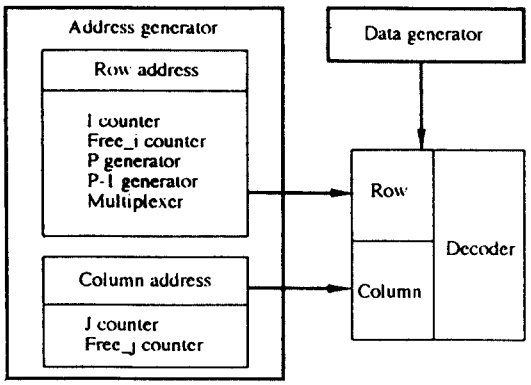


그림 6. 패턴발생부 블럭다이어그램
Fig 6. Block diagram of pattern generator

1. 어드레스 발생부

본 논문에서 제안한 알고리즘에 적합한 어드레스 발생은 표1과 같다.

표 1. 각 알고리즘의 어드레스 발생
Table 1. Address generation of algorithms

	행 어드레스	열 어드레스
알고리즘 1	실효 어드레스	병렬 어드레스
알고리즘 2	순차 어드레스	순차 어드레스

CWSF 테스트 알고리즘의 구현시 필요한 행 어드레스는 기본 셀의 어드레스와 기본 셀에 대한 부배열의 어드레스를 필요로 한다. 기본 셀들은 항상 우수 어드레스를 갖으며, 기본 셀에 대한 CWSF를 테스트하기 위하여 그 기본 셀에 적합한 부배열을 발생시켜 데이터 패턴을 인가해야 한다. 이 부배열에 대한 어드레스 시퀀스를 실효 어드레스 패턴이라 한다.

CWSF 테스트 알고리즘에 의해 기본 셀이 테스트 될 때, 기본 셀에 대한 부배열의 코너 셀 3개가 동시에 테스트 되기 때문에 우수 어드레스 만을 갖는 셀이 기본 셀이 되며, 따라서 기본 셀의 열 어드레스의 발생은 (log n)−1 비트 카운터로 구현할 수 있다(최하위 비트는 항상 0). 본 논문에서는 이 카운터를 I 카운터라 정의한다.

실효 어드레스 발생에 대하여 살펴보면, 각 기본 셀에 대해 p×2 크기를 갖는 부배열를 발생시켜야 한다. 기본 셀의 행 어드레스가 i일때, 실효 어드레스 시퀀스는 다음과 같은 식으로 얻어질 수 있다.

$$i_s = \begin{cases} [(i-p)+p+k] \bmod n, & (0 \le k \le \frac{p}{2}-1) \\ [(i-p)+0+k] \bmod n, & (\frac{p}{2} \le k \le p-1) \end{cases}$$

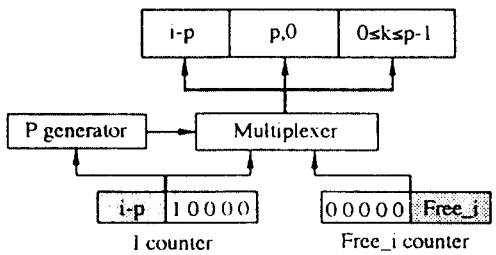


그림 7. 실효 어드레스 발생기의 구성
Fig 7. Stucture of Effective address generator

실효 어드레스 발생기에 대한 구성도는 그림7과 같다. I 카운터는 (log n)−1 비트의 업 카운터이고 Free_i 카운터는 병렬 로드 기능을 갖는 log n 비트 다운 카운터이다. P 발생기는 조합회로로 구현되며 I 카운터에서 입력을 받아 p 값을 발생시켜 P−1 카운터 및 MUX에 출력을 보낸다. P−1 발생기는 P 발생기에서 입력을 받아 p−1 값을 발생시켜 Free_i 카운터에 보낸다. 그리고, Multiplexer는 조합 회로로 구성되며 (log n)−1비트의 I 어드레스 카운터와 log n비트의 Free_i 카운터로부터 입력을 받아 p값의 제어에 의해 실효 어드레스를 생성한다.

그림8은 구현한 행 어드레스 발생기의 시뮬레이션 결과이다. 시뮬레이션 결과를 살펴보면, 초기화 신호(C0)에 의해 초기화된 후 로드신호(LD1)을 받은 다

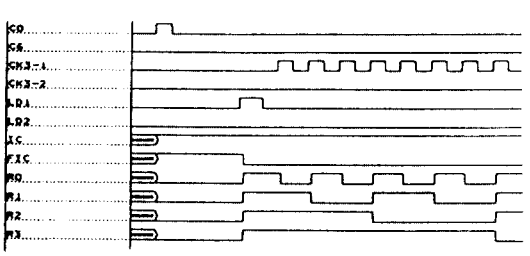


그림 8. 행 어드레스 발생기의 시뮬레이션
Fig 8. Simulation of row address generator

음에 CK3-1 신호를 클럭으로 하여 (0, 0) 셀에 대한 실효 어드레스를 발생시킴을 알 수 있다.

열 어드레스 발생회로는 CWSF 테스트에서 테스트 블럭이 n×2 배열을 갖기 때문에 열 어드레스 발생은 1비트 카운터로 구현하였으며, 이 카운터를 Free_j라 정의하였다.

그림9에 Free_j 카운터의 시뮬레이션 결과를 보여주고 있다. 병렬 테스트 모드 제어 신호(C1)가 1로 발생하는 동안 CK4를 클럭으로하여 열 어드레스의 우수, 기수 결정 제어 신호인 L1과 L2를 만족하게 발생시킴을 알 수 있다.

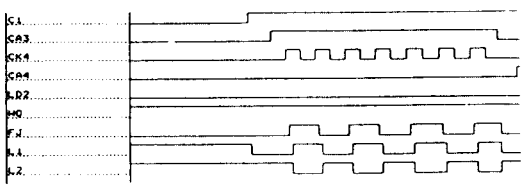


그림 9. Free_j 카운터의 시뮬레이션
Fig 9. Simulation of Free_j counter

병렬 테스트 모드일때에는 병렬 비트라인 디코더에 의해 열 어드레스를 병렬로 우수와 기수 비트라인에 연결시키고, 알고리즘2에서는 J 카운터라 정의한 오름/내림차순(up/down) 카운터에 의해 비트라인 디코더가 정상 동작을 수행하게 된다. 그림10에 J 카운터의 시뮬레이션 결과를 도시하였다.

이 카운터의 입력 클럭은 병렬 테스트 모드 제어 신호(C1)가 0으로 유지될때 인가 되고 C7 신호에 의해 오름/내림차순 카운트 동작이 결정된다.

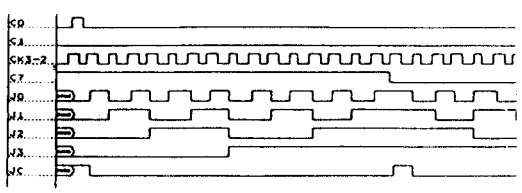


그림 10. J 카운터의 시뮬레이션
Fig 10. Simulation of J counter

2. 데이터 발생부

데이터 발생부는 초기화 제어 변수에 의해 메모리 배열을 0 또는 1로 초기화 시키거나 알고리즘1, 2를 실행할때 셀에 쓰여질 데이터를 정확하게 발생시키기 위하여 2비트로 설계된 Weight 카운터에서 발생되는 W[0]의 값과 Free_j 카운터에서의 출력값의 크기를 비교하여 셀 값을 결정한다.

표2에 제안한 알고리즘에 적합한 데이터 발생기의 동작이 나타나 있다.

표 2. 데이터 발생기의 동작
Table 2. Operation of data generator

	I=1	I = 0					
		C = 0			C = 1		
		W[1]=1		W[1]=0	W[1]=1		W[1]=0
		Free_j ≤ W[0]	Free_j > W[0]		Free_j ≤ W[0]	Free_j > W[0]	
Compare data	V'	V'	V	V	V	V'	V'
Write data	V	V	V'	V'	V'	V	V

(V:셀값, I:초기화 제어 변수, C:V=V')

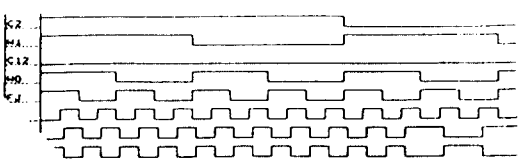


그림 11. 데이터 발생기의 시뮬레이션
Fig 11. Simulation of data generator

그림11은 데이터 발생기의 시뮬레이션 결과를 보여주고 있다. 표2와 비교해 보면 동작조건에 맞는 제어 신호를 발생시키고 있음을 알 수 있다.

본 장에서 구현한 회로는 부록에 도시되어 있다.

Ⅳ. 고장위치 검출부의 구현 및 시뮬레이션

고장위치 검출기는 테스트시 메모리에 고장이 발생하였을때 셀의 수리를 실시하기 위하여 고장 및 그 위치를 테스터에게 알려주는 기능을 한다.

고장 검출을 위해 설계한 병렬 비교기 및 고장 검출기에서는 비교신호가 발생될때, 셀로부터 읽은 RD

(Read Data)와 데이터 발생 회로에서 발생한 CD (Compare Data)를 비교할 수 있도록 설계하였다. RD와 CD 신호가 일치하지 않을시에는 ERROR 신호가 1로 발생된다.

고장위치 검출기는 고장 검출기에서 고장이 검출되었을때 즉, ERROR 신호가 1로 발생될때 테스트를 중지시키고 고장난 셀의 행 어드레스와 셀 배열 블럭을 출력해주고 테스트를 다시 진행시킨다. 이 검출기는 RAM에서 고장이 발생하였을 때, 그 위치를 검출할때까지 BIST 회로의 클럭을 홀드시켜서 다른 제어 신호를 발생시키지 못하게 하는 인터럽트부, 이 인터럽트부에서 HOLD 신호를 받아 그 신호가 1로 발생할 동안 고장 블럭을 찾아내기 위한 블럭 카운터부와 블럭 위치 검출부로 이루어져 있다.

그림12에 고장위치 검출 가능한 BIST RAM을 도시하였다. 구현한 BIST 회로의 패턴 발생기와 고장위치 검출기의 동작 검증을 위해 4×4로 구성된 RAM 배열을 4 블럭으로 구성하여 한번은 정상상태의 RAM에서 실행해 보았고, 다른 한번은 RAM에 임의로 고장을 발생시켜 실행해 보았다.

구현한 회로들의 시뮬레이션 결과를 살펴보면, 그림 13(a)에서는 RAM의 상태가 정상이므로 ERROR 핀에서 발생하는 신호가 0으로 유지되면서 테스트를 계속 진행하고 있다. 반면에 메모리 배열 블럭1의 (0, 1)셀에 임의로 stuck at−1을 준 RAM에서의 시뮬레이션 결과는 그림 13(b)에서 보는바와 같이 Fault and Position Detection (F&P Detection) 핀에서 고장신호와 그때의 고장 셀의 행 어드레스와 블럭 번호를 발생시켜 준다. 즉, BIST 회로에서 패턴 발생기가 제안한 알고리즘에 적합한 패턴을 발생시켜 이 정보로 부터 고장위치 검출기가 고장 셀의 유, 무와 그 위치를 정확하게 검출하고 있음을 확인할 수 있다.

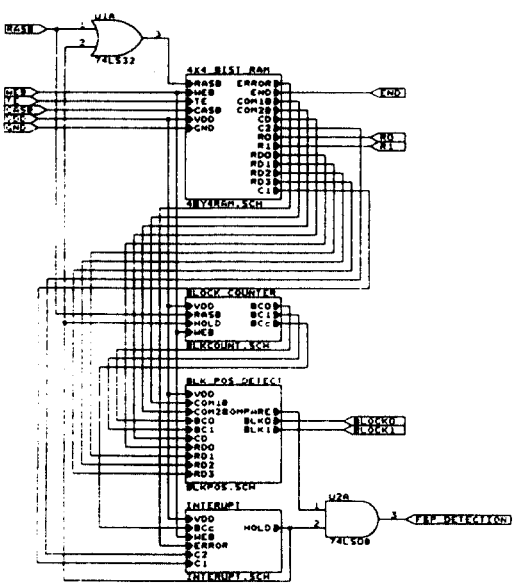
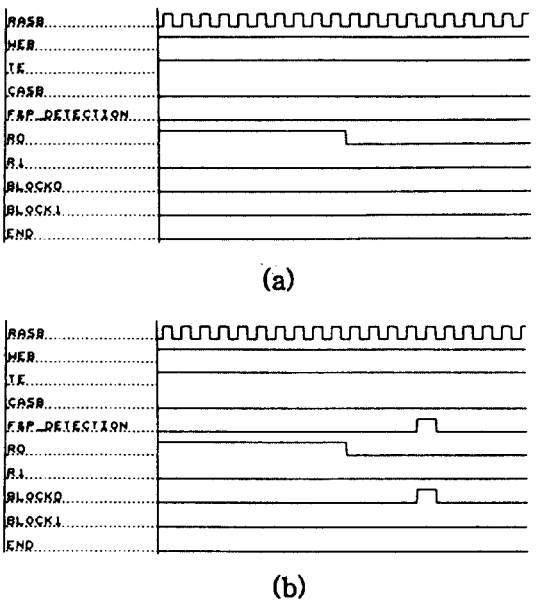


그림 12. 고장위치 검출 가능한 BIST RAM
Fig 12. Fault position detectable BIST RAM



(a)

(b)

그림 13. 시뮬레이션 결과
(a) 정상적인 RAM
(b) 고장 발생한 RAM
Fig 13. Simulation result
(a) Fault free RAM
(b) Faulty RAM

V. 결 론

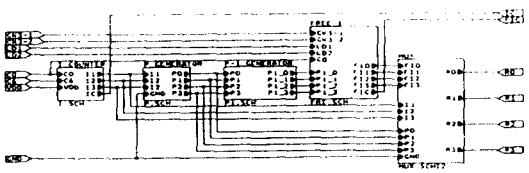
본 논문에서는 외부 테스트 장비에 의존하지 않고 칩 내에 자체 테스트 회로를 삽입시켜 메모리에서의 고장을 검출할 수 있는 BIST 회로에 병렬 기법을 이용하여 테스트 시간을 단축할 수 있는 CWSF 테스트 알고리즘과 비트라인 디코더의 정상 동작시 발생하는 고장을 검출할 수 있는 비트라인 디코더 고장 테

스트 알고리즘을 적용하였다.

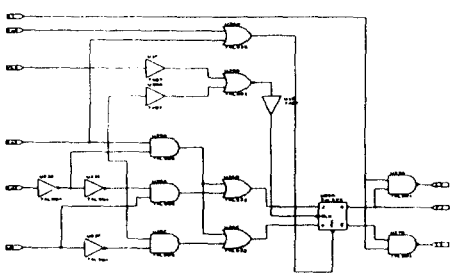
또한, BIST 회로에 적합하고 테스트에 필요한 데이터와 어드레스를 발생시키는 패턴발생 회로를 설계하였고, 패턴 발생회로에서 발생되는 정보로써 고장의 유, 무와 그 위치를 검출하여 메모리의 수리를 돕기위한 고장위치 검출기를 구현하였으며 컴퓨터 시뮬레이션을 통하여 RAM에서 발생하는 고장과 그 위치를 제대로 검출함을 확인하였다.

앞으로 이러한 기법이 실제 DRAM의 테스트 공정에 적용될 수 있도록 BIST 회로 자체의 고장 및 진단 문제 등을 고려한 지속적인 실용화 연구가 필요하다.

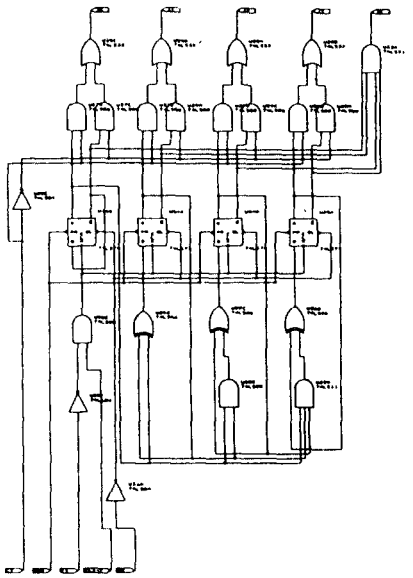
附 錄



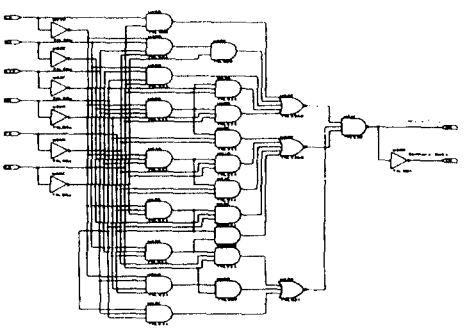
행 어드레스 발생기



Free_j 카운터



J 카운터

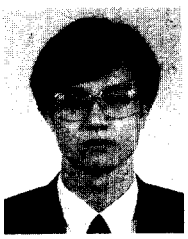


데이터 발생기

參考文獻

1. Y. You, "Testing of memories with tolerable defects," *Int'l Conf. on Elect, Info. and Comm.*, pp. 214-217, Yanji, China, Aug. 23, 1991.
2. M.M. Breuer and A.D. Friedman, *Diagnosis and Reliable Design of Digital Systems*, Rockville, MD : Computer Science Press, 1976. Hayes,
3. R. Nair, S.M. Thatle, and J.A. Abraham, "Efficient Algorithms for Testing Semiconductor Random-Access Memory," *IEEE Trans. Comput*, Vol. C-27, pp. 572-576, June 1978.

4. J. Knaizuk, Jr. and C.R.P. Hartmann, "An Optimal Algorithm for Testing Stuck-at Faults in Random Access Memories," *IEEE Trans. Comput.*, Vol. C-26, pp. 1141-1144, Nov. 1977.
5. V.P. Srini, "API test for RAM chips," *IEEE Trans. Comput.*, Vol. 10, pp. 32-36, July 1977.
6. D.S. Suk and S.M. Reddy, "Test procedures for a class of pattern-sensitive faults in semiconductor random-access memories," *IEEE Trans. Comput.*, Vol. C-29, pp. 419-429, June 1980.
7. J.P. hayes, "Detection pattern-sensitive faults in random-access memories," *IEEE Trans. Comput.*, Vol. C-24, pp. 150-157, Feb. 1975.
8. S.C. Seth and K. Narayanswamy, "A graph model for pattern-sensitive faults in random-access memories," *IEEE Trans. Comput.*, Vol. C-30, pp. 973-977, Dec. 1981.
9. M. Franklin, K.K. Saluja and K. Kinoshita, "Design of a BIST RAM with Row/column Pattern Sensitive fault detection capbility," *IEEE Int'l Test Conf.*, pp. 327-336, May 1989.
10. 전병실, Test 시간 단축을 위한 Parallel Test Algorithm 개발, 초고집적 반도체기술(차세대 기억소자) 공동개발사업 년차 연구보고서, 한국전자통신연구소, 1990.
11. M. Franklin, K.K. Saluja and K. Kinoshita, "A Built-In Self Test Algorithm for Row/Column Pattern Sensitive Faults in RAM's," *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 2, pp. 514-524, April 1990.



金 大 翊(Dae Ik Kim) 정회원
1969년 1월 23일생
1991년 : 전북대학교 공과대학 전자공학과 졸업
1993년 2월 : 전북대학교 대학원 전자공학 석사
1993년 3월~현재 : 전북대학교 대학원 전자공학 박사과정
※주관심분야 : 반도체 메모리 테스트, VLSI 설계, 병렬처리 컴퓨터

鄭 鎭 泰(Jin Tae Jurng) 정회원
1960년 9월 14일생
1984년 : 전북대학교 공과대학 전자공학과 졸업
1986년 : 전북대학교 대학원 전자공학 석사
1986년~1991년 : 금성산전(주) 연구소 근무
1992년 3월~현재 : 전북대학교 대학원 전자공학 박사과정
※주관심분야 : 병렬처리 컴퓨터, 지능망, VLSI 설계

李 昌 基(Chang Ki Lee) 정회원
1961년 7월 15일생
1988년 : 전북대학교 공과대학 전자공학과 졸업
1990년 : 전북대학교 대학원 전자공학 석사
1991년 3월~현재 : 전북대학교 대학원 전자공학 박사과정
※주관심분야 : 반도체 메모리 테스트, VLSI 설계

田 炳 實(Byoung Sil Chon) 정회원
1945년 2월 14일생
1967년 : 전북대학교 공과대학 전기공학과 졸업
1969년 : 전북대학교 대학원 전자공학 석사
1974년 : 전북대학교 대학원 전자공학 박사
1969년 : 미국 Univ. of Notre Dame 전기공학과 객원교수
1986년 : 전북대학교 전자계산소장
1971년 3월~현재 : 전북대학교 공과대학 전자공학과 교수
※주관심분야 : 병렬처리 컴퓨터, 지능망, VLSI 설계