論文93-30A-1-2

Sliding Diagonal Pattern에 의한 Memory Test Circuit 설계

(Design of Memory Test Circuit for Sliding Diagonal Patterns)

金大煥*, 薛秉洙*, 金大容**, 劉泳甲*

(Dae Hwan Kim, Byung Su Seol, Dae Yong Kim, and Young gap You)

要 約

본 논문에서는 고집적 메모리의 testability 향상을 위한 built-in test 기법 채택에 있어 기존의 sliding diagonal test pattern에 따라 효과적으로 메모리의 테스트가 가능한 회로를 delay-element control 설계 방식으로 설계하여, 원래의 $O(n^{3/2})$ 테스트 알고리듬을 O(n)으로 구현 가능하도록 하였다.

설계된 회로의 논리 시뮬레이션을 수행하여 회로의 정상 동작을 확인하였고, 레이아웃을 수행하여 on-chip화 과정에서 필요한 면적 평가를 시도한 것으로 16Mbit DRAM에 대하여 1% 정도의 추가 면적이 필요한 것으로 판명되었다.

Abstract

A concrete design of memory circuit is presented aiming at the application of sliding diagonal test patterns. A modification of sliding diagonal test pattern includes the complexity reduction from $O(n^{3/2})$ to O(n) using parallel test memory concept. The control circuit design was based on delay-element, and verified via logic and circuit simulation. Area overhead was evaluated based on physical layout using a 0.7 micron design rule, resulting in about 1% area increase for a typical 16Mbit DRAM.

Ⅰ. 서론

반도체 설계 및 공정기술의 발전과 대용량 기억장치를 요구하는 시장의 요구에 부응하여 수십 메가 비트의 DRAM 칩이 발표되고 있다. 국내에서도 16M bit DRAM의 상용화와 64M bit DRAM의 시제작을 눈앞에 두고 있다. 그러나 반도체에서의 집적도 증가는 더욱 다양하고 복잡한 고장을 유발하게 되었으며 이에 따라 메모리의 테스트 문제가 메모리 제품이 시장에서 가격경쟁을 이기며 적절한 이윤을 보장받기 위하여 먼저 해결해야 할 가장 심각한 문제 중의 하나로 떠오르고 있다. 즉 매 세대마다 메모리 제품의 제조원가에서 테스트 비용이 차지하는 비중이 증대될 것으로 예견되며, 따라서 테스트의 품질을 유지하면서 테스트 시간의 증가를 억제하기 위한 여러 연구가 진행되고 있다.

*正會員, 忠北大學校 情報通信工學科
(Dept. of Computer & Comm. Eng. Chungbuk Nat'l Univ.)
**正會員, 韓國電子通信研究所
(E T R I)
接受日字:1992年 5月 30日

초고집적 메모리의 테스트는 두 가지의 중요한 목표를 달성해야 하는데, 그 첫 번째가 테스트 시간의 증가에 의한 테스트 비용 상승을 억제하는 일이다. 메모리는 매 세대마다 그 용량이 네 배씩 증가하기 때문에, 종래의 O(n)의 복잡도를 갖는 테스트를 사용하는 경우, 새로운 세대가 등장할 때마다 네 배의 테스트 시간을 요구하게 된다. 이는 동등 가격의 테스트 장비 사용시 장비에 의한 비용 부담이 네 배로 증가한다는 것을 의미한다. 두 번째는 집적도의 증가에 따르는 테스트의 품질열화를 방지하기 위한 새로운 알고리듬의 개발이다. 즉 메모리의 집적도 증가에 따라 종래에는 볼 수 없었던 더욱 복잡한 고장 유형이 발견되는데, 이를 검출하기 위해서 복잡한 테스트 알고리듬을 사용하여야만 만족할만한 fault coverage를 얻을 수 있게 된다. 따라서 종래의 테스트 개념으로는 경제적인 타당성을 갖는 테스트가 불가능하게 되며 테스트 품질열화를 가져오게 된다.

본 연구에서는 메모리 테스트 패턴 중 높은 fault coverage를 가지는 sliding diagonal test pattern을 효과적으로 구현하기 위한 회로를 설계하였다. 설계 과정은 테스트 알고리듬을 정리하여 간략화 하고 이것을 delay element control 설계 방식으로 구현하였다. 이 과정은 논리 시뮬레이션을 통하여 검증되었으며, 상업용 layout tool을 이용하여 면적 평가를 시도하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기능적 테스트(functional test)의 개념과 sliding diagonal test 및 DRAM에서 BIST를 구현하는 방식을 소개하고, Ⅲ장에서는 delay-element를 이용한 제어 회로의 구현 방법과 실제로 설계한 sliding diagonal test pattern generator의 회로 그리고 논리 시뮬레이션을 보이며, 마지막으로 Ⅳ장에서는 설계된 회로의 레이아웃을 통하여 추가 면적을 고려한 회로의 효율성과 추가 연구방향을 기술하였다.

Ⅱ. Sliding Diagonal Test Pattern과 Testability

일반적으로 메모리 테스트는 적절한 고장 모델링 작업으로부터 시작한다.[1] 가장 널리 사용되는 고장 모델은 기능적 모델로서 다양한 물리적 고장들은 SAF(Stuck-At Fault), TF(Transition Fault), CF((Coupling Fault), NPSF(Neighborhood Pattern Sensitivity Fault) 등의 대표적인 4가지 모델로 해석된다. 이 기능적 모델로부터 많은 메모리 테스트 알고리듬 들이 도출되었는데 그 중 고집적 메모리에서 높은 fault coverage를 얻을 수 있는 테스트 알고리듬 중의 하나가 sliding diagonal pattern

```
for d:= 0 to 1 do
begin
   write d to all non_diagonal cells:    /*
   write d/ to all diagoanl cells:
   for diag := 0 to maxcolumn - 1 do
   begin
     read all cells:
     for i := 0 to maxrow - 1 do
     begin
        A[i, (diag+i) mod(maxcolumn)] := d:
        A[i, (diag+i+1) mod(maxcolumn)] := d/:
     end:
   end:
end:
```

그림 1. Sliding diagonal test algorithm의 Pseudo-code

Fig. 1. Pseudo code of sliding diagonal test algorithm.

Sliding diagonal pattern은 그림 1에 보여진 바와 같이 셀 어레이(cell array)에 '0'의 배경을 가지고 어레이의 대각선 상에 놓인 CUT(Cell Under Test)에 '1'을 저장하였다가 다시 읽어내는 것이다. 이 테스트 과정에서는 CUT의 대각선을 한 칸씩 옮겨가며 반복적으로 시행되고 모든 셀이 한 번씩 대각선 아래의 CUT로서 테스트된 후에 배경 셀과 CUT의 데이타 값을 바꾸어 전체 과정을 반복한다.

이 sliding diagonal pattern은 높은 fault coverage를 가지는데 우선 어드레스 디코더에서의 고장과 SAF, TF는 모두 검색이 가능하며 coupling fault, 5 cell pattern sensitivity의 일부 테스트가 가능하다. 또한 고집적 메모리에서의 주요 고장으로 등장하고 있는 bit/word line crosstalk, bit line interference등도 테스트가 가능하다.[2]

일반적으로 복잡한 양상을 보이는 고장 유형을 검사하기 위하여는 다양한 테스트 패턴에 의한 검사가 바람직하다. 그러나 실제 산업체에서 널리 사용되고 있는 checkerboard나 march 패턴이 검색할 수 있는 대부분의 고장에 대한 테스트를 sliding diagonal 패턴으로도 테스트가 가능하므로 본 연구에서는 한 가지 패턴에 의한 on-chip 테스트 회로 구현에 중점을 두었다.

이 패턴의 복잡도(complexity)는 $O(n^{3/2})$로서 이 패턴이 고집적 메모리의 양산 테스트로 사용 가능하

기 위해서는 built-in test의 도입에 따른 테스트 시간 단축 방안이 마련되어야 한다. 주어진 failure를 충분히 cover하여 높은 fault coverage를 유지시키면서 메모리 테스트 시간을 단축시킬수 있는 방법 중의 하나가 built-in self-test(BIST)의 도입이다. BIST 방식 사용시의 장점은 고가의 테스트 장비가 필요 없으며, 칩 내부에서 여러 비트를 동시에 테스트할 수 있는 병렬 테스트의 도입이 가능하고, 이에 따라 테스트 시간 단축이 가능하다는 점이다.

물론 BIST 방식의 사용시 고려해야할 여러 사항이 있는데 첫째가 BIST 회로가 차지하는 silicon die의 면적이 커질 경우 메모리 수율과 가격에 악영향을 줄 수 있다는 점이다. 둘째 메모리 access time은 메모리 제품의 주요 parameter이므로 BIST의 도입에 따른 access time에의 영향이 최소화 되어야 한다. 셋째 테스트를 위해 추가되는 핀의 사용은 가능한 억제되어야 하며, 마지막으로 BIST를 위한 회로 그 자체도 테스트가 가능해야 한다는 점이다. 이들 여러가지 사항을 고려한 적절한 BIST 방식의 사용은 메모리 테스트를 더욱 쉽고, 간단하며, 빠르게 수행 가능하도록 한다.[3]

DRAM에 BIST를 구현하기 위해서 일반적인 DRAM의 구조에 테스트 제어 회로들이 추가되어야 한다.[4] 그것은 테스트 시작을 감지할 수 있는 회로, 테스트 중에 어드레스와 해당 어드레스에 적절한 테스트 값을 만들어 내는 테스트 패턴 발생회로, 테스트 중에 read/write를 제어할 수 있는 회로, 저장되어 있는 값을 읽어 RAM의 정상여부를 검증할 수 있는 비교회로 등이다.

먼저 테스트 시작 감지회로의 기능은 보통의 RAM 동작모드에서 테스트 모드로 변환할 수 있도록 제어하는 것이다. 이 기능은 여러 방식으로 실현이 가능한데, 통상 사용하지 않는 순서로 제어신호를 입력함으로써 테스트 모드로 들어가거나 RAM의 단자에 전원전압 이상의 고전압으로 테스트 모드로 들어가는 방식이 도입되고 있다. 물론 RAM에 테스트 전용의 핀을 추가하는 방법도 있을 수 있으나, 이는 핀이 제조원가에 미치는 영향이 크므로 바람직하지 않다.[5,6]

메모리 테스트가 시작되면 모든 메모리 셀내에 적절한 테스트 데이타를 써 넣을 수 있는 것이 BIST의 주요 기능중 하나인데, 이 어드레스와 데이타의 발생은 카운터, LFSR(linear feedback shift register), ROM-based 방식등을 이용할 수 있다.[5,7,8] 여기서 카운터를 이용한 어드레스의 발생은 카운터를 구성하기 위해 필요한 주변 게이트등으로 인하여 LFSR보다 하드웨어가 더 필요한 단점이 있으나 NPSF를 위한 테스트에는 효과적이다. LFSR은 march test등의 응용에 사용되어 왔으며 카운터보다 실리콘 면적 소요가 작고 자체 테스트가 쉬운 장점이 있다. 또한 마이크로프로세서를 이용한 어드레스 생성방식은 하드웨어 부담이 심하고 performance가 낮은 반면 여러 가지의 테스트 패턴의 사용이 자유롭다는 장점이 있다.[8]

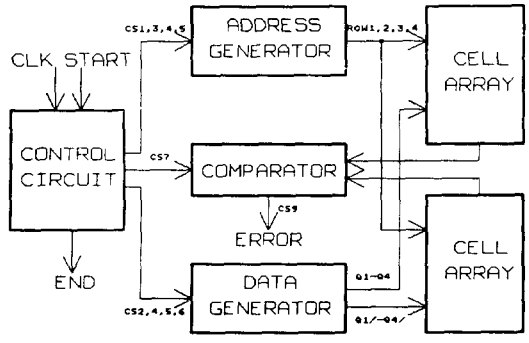


그림 2. BIST 회로 블럭도
Fig. 2. BIST circuit block diagram.

테스트 결과의 검증에 있어서 테스트 패턴을 기록한 후에 다시 데이타를 읽어 DRAM의 정상여부를 가리는 검증회로 또한 여러 방식이 사용되고 있으나, 두 가지로 분류할 수 있다. 하나는 기대값과의 비교방식과, 다른 하나는 복수 개의 메모리 어레이 블럭으로부터 데이타를 읽어 서로 비교하는 방식이 대표적이다.[2,7,9,10]

Ⅲ. Sliding Diagonal Test Pattern Generator 설계

그림 2는 본 연구에서 설계된 sliding diagonal test pattern의 on-chip generation을 위한 BIST 회로의 block diagram이다. BIST 회로는 전체 4부분으로 구성되어 있는데, 제어회로(control circuit), 어드레스 생성기(address generator), 테스트 데이타 생성기(test data generator), 비교기(comparator)가 그것이다. 우선 테스트 회로 설계를 효과적으로 보여주기 위하여 2 block의 4×4 메모리 어레이에 대하여 올바른 sliding diagonal test pattern의 발생과 읽기 동작 신호를 만들고 있다. 물론 이 회로는 카운터와의 비교로 테스트 하고자 하는 셀의 수를 결정하므로 카운터의 규모를 확장하여 쉽게 대규모 RAM에 맞는 구조로의 변환이 용이하다. 각 블럭별 기본 구조와 기능을 살펴보면 우선 제어회로는 다른 BIST 회로에 대한 클럭의 공급과 초

기화, 테스트 시작 신호의 입력, 어드레스 생성기와 테스트 데이타 생성기의 구동, 읽기 신호의 발생 그리고 테스트 종료시 END 신호의 출력 등의 제어기능을 한다. 어드레스 생성기의 기본 구조는 동기식 카운터로서 제어회로에서의 제어신호에 따라 순차적으로 row address를 지정하는 기능을 한다. 테스트 데이타 생성기의 기본 구조는 circular shift register로서 역시 제어 회로로부터의 제어신호와 클럭의 입력으로 해당 row별로 테스트 데이타를 출력시키는 기능을 담당한다. 마지막으로 비교기는 서로 다른 메모리 어레이로부터 데이타를 읽어 비교하여, RAM의 정상 여부를 검증하며 간단한 게이트로 구성이 가능하다.

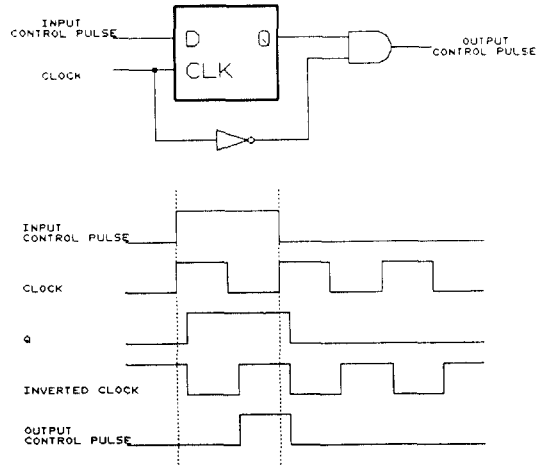
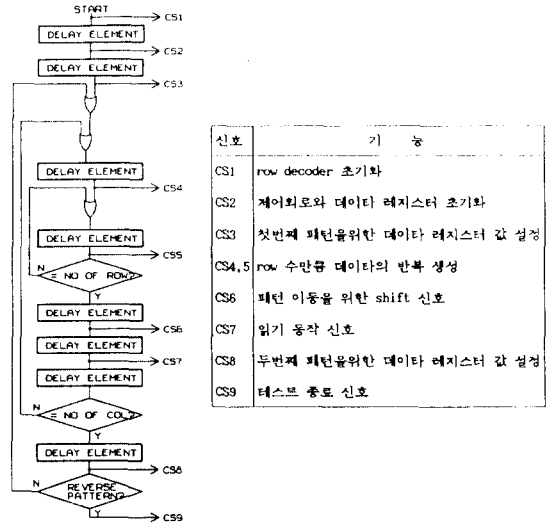


그림 3. Delay element와 동작파형
Fig. 3. Wave form of delay element output signal.

본 연구에서는 BIST 회로의 제어를 담당하는 제어회로에서의 신호제어 방식으로 delay-element를 이용한 신호 발생, 비교, 진행방식을 이용하였다. 그림 3은 이 방식에서 사용되는 기본 모듈인 delay element의 회로와 논리 파형으로 신호의 지연을 고려하여 설계하였다. 입력 펄스가 D flip flop의 출력으로 나올 때는 약간의 지연이 발생하므로 반전된 클럭신호와 함께 AND 게이트를 통과시켜 글리치가 발생하지 않는 반 클럭분의 제어신호를 얻을 수 있다. 물론 이 신호를 다음 단의 입력 펄스로 다시 사용한다. 제어회로에서의 신호제어 회로설계 방식으로 state-table method나 sequence counter를 이용한 방법도 생각해 볼 수 있다.[11] 그러나 state-table method는 최소 갯수의 게이트를 사용하며 속도가 빠른 장점은 있으나 설계가 복잡하며, 무엇보다 테스트 패턴 회로에서 많이 사용되는 반복되는 패턴이나 루프등의 회로 동작에 대한 정보를 쉽게 알 수 없어 설계의 디버깅이 어려운 random structure가 되기 쉬운 단점이 있다. 그리고 sequence counter를 이용한 제어회로의 구성은 많은 양의 하드웨어가 요구되므로 BIST에는 적합하지 않다. 결국 flowchart로부터 delay-element를 이용하여 쉽게 회로 구성이 가능한 delay-element를 이용한 신호제어 방식이 BIST용으로 매우 적합하다고 볼 수 있다.



신호	기 능
CS1	row decoder 초기화
CS2	제어회로와 데이타 레지스터 초기화
CS3	첫번째 패턴을위한 데이타 레지스터 값 설정
CS4, 5	row 수만큼 데이타의 반복 생성
CS6	패턴 이동을 위한 shift 신호
CS7	읽기 동작 신호
CS8	두번째 패턴을위한 데이타 레지스터 값 설정
CS9	테스트 종료 신호

그림 4. 제어회로의 제어신호 발생
Fig. 4. Control signal generation of control circuit.

다음 그림 4는 앞에서 설명한 delay element를 사용하여 sliding diagonal test 방식을 control unit의 전체적인 흐름과 각 단계에서 발생하는 제어신호의 종류와 기능을 보여준다. 그림에서 표시된 CS1 - CS9는 테스트가 진행됨에 따라 발생되는 회로의 제어신호로서 본 회로에서는 9가지의 제어신호가 발생된다. 먼저 테스트 시작을 알리는 START 신호가 일발 펄스로 입력되면 최초의 제어신호인 CS1이 발생되는데 CS1이 하는 일은 row decoder의 초기화이다. 하나의 delay element를 통과하게 되므로 다음 제어신호인 CS2는 다음 클럭에서 발생되며 제어 회로의 모든 부분과 데이타 생성기를 초기화 시킨다. CS3에서 첫 번째 패턴의 생성을 위하여 데이타 생성기에 초기값을 주며 CS4, CS5가 메모리 어레이의 row 수만큼 발생되어 하나의 패턴이 완성된다.

하나의 패턴이 완성되면 다음 패턴을 위해 데이타 생성기내의 정보를 단순히 이동시키는 신호인 CS6이 따르고 각각의 패턴이 완성되면 셀 전체를 읽어 정상 여부를 판별하기 위한 읽기신호인 CS7이 발생된다. 이상의 과정을 column의 수만큼 비교하여 반복하면 "0" 바탕위에 "1"의 대각선이 완전히 한 바퀴 이동되는 검사 과정까지 마치게 된다. 다음은 패턴의 형태를 "1" 바탕 위에 "0"의 대각선 형태로 바꾸어 다시 테스트가 반복된다. 패턴의 모양을 바꾸는 제어신호가 CS8이며 이것이 반복되어 모든 테스트가 끝나면 테스트의 끝을 알리는 CS9가 발생되며 테스트가 종료된다.

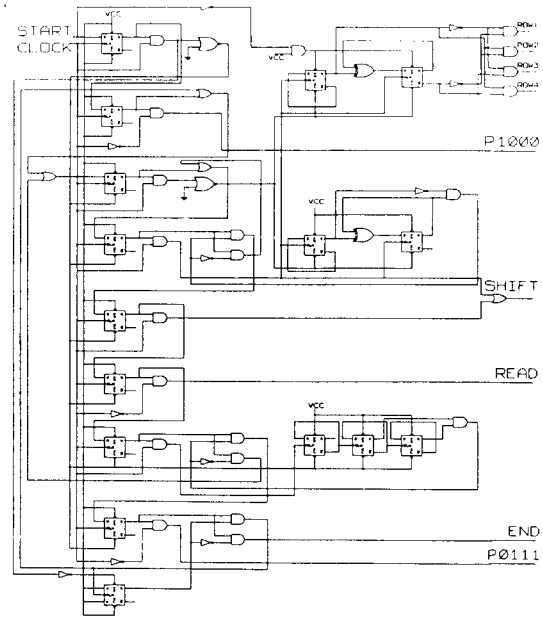


그림 5. Sliding diagonal test generator 회로도

Fig. 5. Circuit diagram of sliding diagonal test pattern generator.

다음 그림 5는 본 연구에서 설계한 sliding diagonal test pattern generator의 회로도이다. D flip-flop과 기본 게이트들만을 이용하여 그림 3의 제어회로를 구성하고 있다. 이 회로의 동작을 확인하기 위해 논리 시뮬레이션을 수행한 결과 파형이 그림 6에 제시되었다. 테스트가 진행됨에 따라 설계된 sliding diagonal test pattern generator에서 출력되는 결과 파형을 보여주는데, 4개의 row address 신호인 ROW1, ROW2, ROW3, ROW4, 데이타 생성기의 출력인 Q1, Q2, Q3, Q4, 읽기 신호, 에러 신호, 테스트 종료신호인 END 신호의 출력 파형 등이 표시되어 있다.

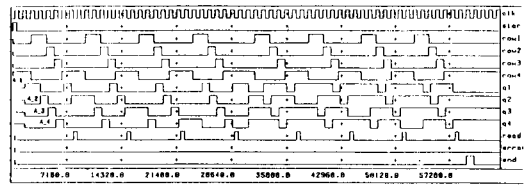


그림 6. 제어회로의 논리 시뮬레이션결과

Fig. 6. Logic simulation results of control circuit.

이 시뮬레이션 결과로 row address인 ROW1, ROW2, ROW3, ROW4는 순차적으로 HIGH가 된다. 이때 데이타 생성기에서의 출력값이 단계별로 변해가는 것을 볼 수 있는데 첫 번째 ROW1이 열리는 동안 발생되는 테스트 데이타 Q1, Q2, Q3, Q4의 값은 "1,0,0,0"이다(A-1). 다음 ROW2가 열릴때 data 값은 "0,1,0,0" (A_2)이고 첫 번째 패턴이 완성되는 순간은 A_4로서 하나의 패턴이 완성된 후 다음 클럭에서 읽기 동작의 수행을 위한 읽기 신호가 발생한다. 4X4 어레이의 에에서 8개의 패턴이 완성된 후 END 신호로서 테스트가 종료된다.

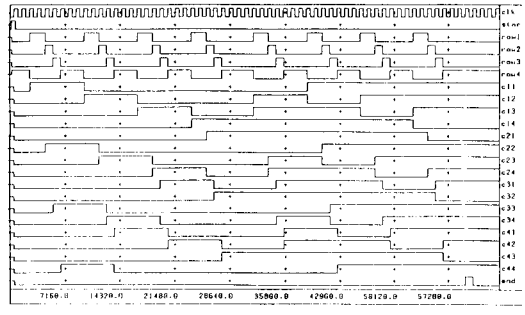


그림 7. 테스트 진행중의 셀 값의 변화

Fig. 7. Simulation results of cell data change during test.

그림 7은 테스트가 진행되는 동안 셀에 저장되는 정보의 변화를 보기위해 C11-C44까지 16개 셀에서의 출력 결과를 보여주는 시뮬레이션 결과이다. 이 결과를 각 단계별로 도식화 하면 그림 8과 같은 형태가 된다. 이로써 sliding diagonal test pattern generator가 정상 동작함을 쉽게 확인할 수 있다.

다음 그림 9는 테스트 데이타를 발생시키는 테스트 데이타 생성기의 회로도로서 기본 구성은 D flip-flop으로 연결된 feedback shift register이다. 원하는 데이타 패턴으로 레지스터내의 정보를 바꾸는 제어신호선인 P1000, P0111, P000과 이동을 위한

CLK 신호로서 제어를 하며, Q1, Q2, Q3, Q4에서 데이타의 출력이 이루어진다. 데이타 생성기의 동작을 로직 시뮬레이션으로 살펴보면 다음 그림 10과 같다.

1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0
0 0 0 0	0 1 0 0	0 1 0 0	0 1 0 0
0 0 0 0	0 0 0 0	0 0 1 0	0 0 1 0
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1
a_1	a_2	a_3	a_4
0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0
0 1 0 0	0 0 1 0	0 0 1 0	0 0 1 0
0 0 1 0	0 0 1 0	0 0 0 1	0 0 0 1
0 0 0 1	0 0 0 1	0 0 0 1	1 0 0 0
b_1	b_2	b_3	b_4
0 0 1 0	0 0 1 0	0 0 1 0	0 0 1 0
0 0 1 0	0 0 0 1	0 0 0 1	0 0 0 1
0 0 0 1	0 0 0 1	1 0 0 0	1 0 0 0
1 0 0 0	1 0 0 0	1 0 0 0	0 1 0 0
c_1	c_2	c_3	c_4
0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
0 0 0 1	1 0 0 0	1 0 0 0	1 0 0 0
1 0 0 0	1 0 0 0	0 1 0 0	0 1 0 0
0 1 0 0	0 1 0 0	0 1 0 0	0 0 1 0
d_1	d_2	d_3	d_4
0 1 1 1	1 0 1 1	1 1 0 1	1 1 1 0
1 0 1 1	1 1 0 1	1 1 1 0	0 1 1 1
1 1 0 1	1 1 1 0	0 1 1 1	1 0 1 1
1 1 1 0	0 1 1 1	1 0 1 1	1 1 0 1
e_1	f_2	g_3	h_4

그림 8. 셀 어레이에서의 패턴 변화
(4×4 어레이의 예)

Fig. 8. Pattern migration of a cell array (4×4 array case).

제어신호 P1000이 입력되면 래지스터의 내용은 첫 번째 D flip-flop만이 HIGH이며 나머지는 모두 LOW인 상태로 변하고 다음 연이은 CLK 신호에 따라 이 데이타들이 순차적으로 이동하며 출력된다. 또 다른 제어신호인 P0111의 입력에 역시 래지스터 내의 정보가 변하고 이동 과정이 반복된다.

이 테스트 패턴의 기록 후에 병렬 비교기를 사용하여 RAM내에 고장의 유무를 가리게 된다. 이 비교기로 비교되는 두 개의 어레이는 반대 패턴으로 기록되는데, 이는 똑같은 고장이 메모리의 같은 부위에서 발생할 수 있는 commom mode failure를 방지하는 목적이다. 그러므로 row단위로 4bit씩 검증을 위해 설계한 비교기는 간단한 게이트로 구성이 가능하며, 같은 논리값이 입력될 경우에는 error 신호를 내보낸다.

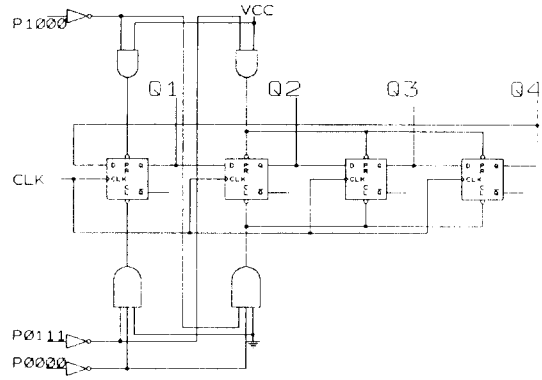


그림 9. 데이타 발생기의 회로도

Fig. 9. Circuit diagram of data generator.

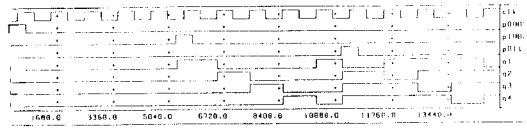


그림 10. 데이타 발생기의 논리 시뮬레이션의 결과

Fig. 10. Logic simulation results of data generator.

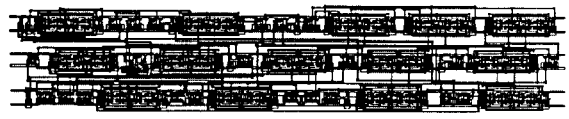


그림 11. 제어회로의 레이아웃

Fig. 11. Overall layout of control circuit.

설계된 sliding diagonal test pattern generator가 차지하는 추가면적을 검토하고 그 타당성을 알아보기위해 레이아웃 작업을 수행하였다. 그림 11은 설계된 제어회로의 레이아웃 모습으로서 16Mb급에서 사용되는 0.7μ m design rule에 따라 만들어졌다. 면적 평가 결과 제어회로가 차지하는 면적은 110μ m × 1200μ m이고 테스트 데이타 생성기가 60μ m × 600μ m이었다. 따라서 이들 둘을 합

한 면적은 약 0.17mm²이다.

일반적으로 0.8μ m design rule로 설계된 4M DRAM의 면적이 약 60-70mm²이고, 0.6-0.5μ m design rule을 사용하는 16M급의 DRAM 면적이 110-130mm²임을 생각해 볼 때 실제 DRAM에 구현시 예상되는 구조 변경에 따른 추가 면적을 고려하더라도 전체 추가되는 면적은 최고 1%를 넘지 않으리라 예상된다. 참고로 본 연구에서 설계한 회로의 시뮬레이션은 Mentor Graphics사의 logic simulator인 Quicksim에서 수행되었으며 레이아웃은 Valid사의 Construct에서 수행되었다.

Ⅳ. 결론

본 논문에서는 고집적 DRAM의 testability 향상을 위한 BIST회로의 구현 방식을 알아 보고, 높은 fault coverage를 갖는 sliding diagonal test pattern generator를 구현하였다. 설계된 회로의 논리 시뮬레이션을 수행하여 정상 동작을 확인하였고, 레이아웃을 진행하여 면적 평가를 시도한 결과 실제 DRAM에 구현시 추가되는 전체 면적은 약 1%를 초과하지 않으리란 결론을 얻었다.

지금까지 설계한 BIST를 병렬테스트[6] 기법과 함께 RAM 설계에 적용할 경우 RAM 내부에서 복수 병렬 비트 테스트가 수행되어 word line 단위로 테스트 패턴의 기록이 가능하다. 따라서 메모리 내의 셀 갯수가 n이라할 때 $O(n^{3/2}) \times O(n^{1/2})$의 테스트 효과를 볼 수 있으므로 sliding diagonal pattern의 복잡도는 $O(n^{3/2})$에서 O(n)으로 줄어 들게 된다. 결국 fault coverage가 높은 sliding diagonal pattern을 고집적 DRAM에서 사용하는 것이 가능하리라 생각된다.

參考文獻

[1] M.S.Abadir and H.K.Reghbati, "Functional testing of semiconductor random access memories", *ACM Computing Survey*, vol. 15, no. 3, pp. 175-198, Sept. 1983.

[2] Y.You and J.P.Hayes, "A self-testing dynamic RAM chip," *IEEE J. Solid-State Circuits*, pp. 428-435, Feb. 1985.

[3] A.J.van de Goor, *Testing Semiconductor memories : Theory and Practice*, John Wiley & Sons, Reading, Mass, 1991.

[4] M.Hentz, "An advanced pattern generator for memory testing," *Int'l Test Conf.*, pp. 45-49, 1980.

[5] T.Takeshima et al., "A 55ns 16Mb DRAM with built-in self test function using microprogram ROM," *IEEE J.Solid State Circuits*, vol. 25, no. 4, pp. 903-911, August 1990.

[6] Y.You, "Multi-mega bit memory test technology," *SEMICON/KOREA, 91, Technical Symposium*, pp.IV:22-31, Sept. 1991.

[7] M.Nicolaidis, "An efficient built-in self test scheme for functional test of embedded RAMs," IMAG/TIM3, France, 1985.

[8] H.C.Ritter, and B.Muller, "Built-in test processor for self- testing repairable random access memories," *Int'l Test Conf.*, pp. 1078-1084, Sept. 1987.

[9] R.Dekker, F.Beenker and L.Thijssen, "A realistic self-test machine for static random access memories," *Int'l Test Conf.*, pp.353-361, Sept. 1988.

[10] T.Ohsawa et al., "A 60ns 4Mb CMOS DRAM with built-in self-test," *IEEE Int'l Solid-State Circuits Conf.*, pp.286-287 and 430, Feb. 1987.

[11] J.P.Hayes, *Computer Architecture and Organization*, 2nd Ed., McGraw-Hill, New York, 1988.

著者紹介

金 大 煥(正會員)
1991年 2月 충북대학교 정보통신공학과(공학사). 1993年 2月 충북대학교 대학원 정보통신공학과 석사과정 졸업예정. 주관심분야 초고집적 메모리의 Test 및 Testable Design, ASIC Design 등임.

•

薛 秉 洙(正會員)
1992年 2月 충북대학교 정보통신공학과(공학사). 1992年 2月 충북대학교 대학원 정보통신공학과 석사과정 재학중. 주관심분야 초고집적 메모리의 Test 및 Testable Design, ASIC Design 등임.

•

金 大 容(正會員)
현재 한국전자통신연구소 기억소자 개발 사업 본부 개발1실 책임연구원

劉 泳 甲(正會員)
1948年 3月 22日生. 1975年 8月 서강대학교 전자공학과(공학사). 1981年 8月 미국 미시간대학교 전기전산학과(공학석사). 1986年 4月 미국 미시간 전기전산학과(공학박사). 1975年 8月-1979年 8月 국방과학연구소 연구원 1982年 4月-1986年 4月 미시간 전산연구소. 1986年 2月-1988年 2月 금성반도체(주) 책임연구원. 1988年 8月-현재 충북대학교 정보통신공학과 부교수, 학과장, 정보통신산업연구소 소장 1991年 5月 전국 정보통신 교수협의회 부회장. 주관심분야 반도체 집적회로 테스트, 고장극복형 컴퓨터구조, 가변익항공기제어, 중·대형 컴퓨터 제작 및 제조기술, 정밀인쇄장치 구조설계등임.