論文2002-39SD-12-4

Redundancy Cell Programming이 용이한 병렬 I/O DRAM BIST

(Parallel I/O DRAM BIST for Easy Redundancy Cell Programming)

柳 在 熙 *, 河 昌 佑 **

(Jae Hee You and Chang Woo Ha)

요 약

테스트와 동시에 오류 비트의 수와 위치를 파악하도록 하여 redundancy 프로그래밍이 용이한 다수 비트 출력 DRAM을 위한 BIST 구조가 소개되었다. 일반적으로, DRAM 셀이 n개의 블록으로 구성된 경우에, 단지 n개의 비교기와 한 개의 3가지 상태 엔코더를 사용하여, 무오류 상태, 한 개의 오류가 있을 경우, 오류 상태 및 오류비트가 존재하는 블록의 위치, 두개의 블록에 오류가 있을 경우 오류 상태 등 총 n + 2개의 상태를 나타낼 수 있다. 제안된 방법을 통하여, 두개 이상의 블록에 오류가 있을 경우, 오류 비트의 위치와 수를 파악하는 방법으로 용이하게 확장 구현가능하다. 8블록으로 구성된 64MEG DRAM 경우의 성능 비교 결과 단지 0.115%의 칩 면적 증가로, 테스트 및 redundancy 프로그래밍 시간이 1/750로 감소하였다.

Abstract

A multibit DRAM BIST methodology reducing redundancy programming overhead has been proposed. It is capable of counting and locating faulty bits simultaneously with the test. If DRAM cells are composed of n blocks generally, the proposed BIST can detect the state of no error, the location of faulty bit block if there is one error and the existence of errors in more than two blocks, which are n + 2 states totally, with only n comparators and an 3 state encoder. Based on the proposed BIST methodology, the testing scheme which can detect the number and locations of faulty bits with the errors in two or more blocks, can be easily implemented. Based on performance evaluation, the test and redundancy programming time of 64MEG DRAM with 8 blocks is reduced by 1/750 times with 0.115% circuit overhead.

Keyword : DRAM, BIST, redundancy 프로그래밍, defect 셀, 테스트

* 正會員, 弘益大學校 電子電氣工學部
(School of Electronics and Electrical Engineering, Hongik University)
** 正會員, 三星電氣 技術總括硏究所 ASIC 2팀
(Samsung Electro-Mechanics, ASIC 2 Team, R & D Center)
※ 본 연구는 산학협동재단(과제명 : Faulty bit 추적이 용이한 고집적 DRAM Multibit 테스팅)의 지원을 받아 수행되었습니다.
接受日字:2002年3月19日, 수정완료일:2002年11月13日

I. 서 론

현재 DRAM은 CMOS 공정기술의 향상과 컴퓨터의 기억용량 및 비트 폭에 대한 요구 증가로 향후 Multi-Giga DRAM의 개발에 박차를 가할 것으로 보인다. 따라서 다수의 DRAM 셀의 테스트 시간 감소를 위하여 각각의 DRAM 칩 안에서 자체적으로 테스트를 신속히 수행하는 BIST (Built in Self-Test)와 더불어 DRAM 수율 향상을 위하여, 다수의 defect 셀 들을 효율적으로

redundancy 셀로 대체하는 방법의 중요성이 더욱 증가하고 있다. 특히, DRAM defect 셀의 수와 위치는 각 칩마다 달라, redundancy 셀 프로그래밍이 별도로 필요하여, 생산성에 큰 영향을 미치게 된다.

기존의 연구를 살펴보면 DRAM의 redundancy 교체를 용이하게 하기 위한 연구는 많이 이루어 졌으나, defect 셀을 구성하는 faulty 비트 위치 탐색을 위한 연구는 거의 전무한 편이었다. 그러나 실제로 DRAM의 테스트에 있어 faulty 비트를 탐색할 수 있을 경우, [1]에서 나타난 바와 같이 테스트 및 redundancy 프로그래밍 시간을 약 1/100 정도로 감소시킬 수 있다. [1]에서는 이를 위해 DLL, APG, Data converter를 포함하고 있는 TMU 2개, VLW (Very long word) 버퍼, Failed address align등의 회로가 요구되며, 또한 많은 수가 칩 안에서 반복되는 DRAM 셀 읽기 회로에 2개의 transistor가 추가되어, 오버헤드가 막대하나, 본 연구에서는 on-chip controller를 사용하여, faulty 비트가 위치하는 세분화된 블록을 탐색하고, 이 안에서 faulty 비트 위치를 파악하는 2단계 테스트구조를 사용하여, 오버헤드를 최소화 시키는 방안을 제안하였다. [2, 3]에서도, redundancy steering 및 allocation logic등이 설계되어 있으나 faulty 비트를 탐색하는 회로는 없다. 테스트 시간과 경비가 생산원가에 차지하는 비중이 증가함에 따라, [4]에서는 DRAM 셀을 4개의 블록으로 나누어 4개의 비트를 동시에 테스트하는 방법이 제안되었다. 그러나 집적도가 증가함에 따라 테스트해야 할 셀 숫자가 증가되어, [4]에 비해 더욱 많은 양의 bit를 동시에 테스트 할 필요가 있다. Line Mode Test (LMT), Merged match-line 테스트를 사용한 [5]에서는 우선 정상 I/O를 이용하여 테스트 데이터를 셀 로우에 병렬적으로 쓰기한다. 이후 여러 개 셀의 column selection line을 한꺼번에 동작시켜 한 개의 정상 I/O를 통해 읽기 하여 테스트한다. 따라서, 모두 동일한 데이터에 대한 테스트로 제한된다. 따라서 [6]에서는 각 비트 line pair 마다 multipurpose register를 두어, random 데이터에 대해 셀 로우에 있는 각 비트 line pair를 독립적으로 테스트하고 있다. 그러나 [5, 6]에서는 많은 셀이 있는 블록단위로 테스트하므로 시간은 현저히 감소될 수 있으나, 오류가 발생한 블록 전체를 한 비트씩 테스트를 하여 faulty 비트를 찾아내야 하므로 추적하는데 많은 시간이 소요된다. 위의 문제점은 향후 고집적 Multi-Giga DRAM에 있어서 더욱 중요해지리라 생각된다. 따라서 위의 문제점을 개선하기 위하여, Multi bit test & Faulty bit Location test (MTFL) 방법이 제안되었다. 즉, 고속으로 병렬 테스트를 수행하여, 다량의 셀을 동시에 테스트하고, 폐기 DRAM 칩의 신속한 판별을 용이하게 한다. 만일 폐기 하지 않을 경우는, redundancy 프로그래밍을 위한 faulty 비트의 수와 위치를 테스트와 병렬적으로 추적 가능하게 하되, 이를 위한 테스트 회로에 대한 부담을 최소화를 시키기 위하여 정상적인 읽기/쓰기를 위한 I/O 및 회로를 공유하는 방법을 소개하였다. II장에서는 테스트 아키텍처에 대해 설명하였고, III장에서는 MTFL 구성회로를 설계하였으며, IV장에서는 검증 및 레이아웃 결과를 기술하고, V장에서는 성능 분석 및 비교 결과를 기술하였다.

II. 테스트 아키텍처

일반적인 DRAM BIST에서는 Pre-Laser Repair 테스트시 오류가 발생한 셀을 판단하여 Redundancy 셀로 치환한다. On chip BIST 회로를 구현하기 위한 칩 면적 증가를 최소화시키기 위하여 I/O를 Local 및 Global I/O와 공유하도록 하고, 현재의 추세인 Multi I/O DRAM의 I/O 경로를 병렬 I/O로 사용함으로써 기존보다 훨씬 짧은 시간에 테스트가 가능하도록 하였다. 또한 외부에 테스트를 위한 핀을 따로 만들지 않도록 기존의 DQ 핀을 사용하는 방법을 사용하였다.

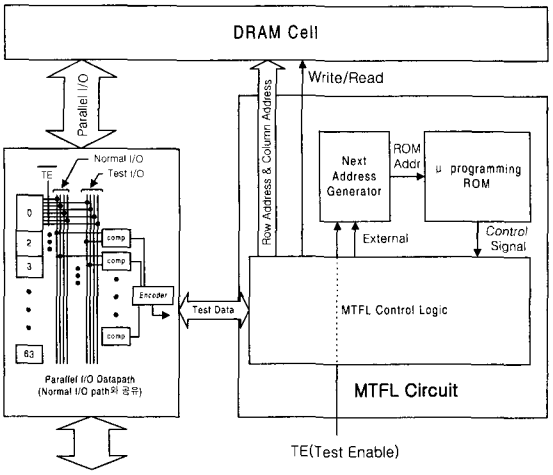


그림 1. MTFL 상위 아키텍처
Fig. 1. Top level architecture of MTFL.

<그림 1>은 MTFL 상위 아키텍처이다. MTFL 회로는 제어 조직과 Next address Generator 그리고 마이크로프로그래밍 롬으로 구성되어 있다. TE (test enable)를 통해 MTFL을 동작시키면, 마이크로프로그래밍 롬은 각 제어신호를 발생한다. 또한 Next address Counter와 데이터 라인을 통해 메모리 셀 어레이에 테스트를 위한 주소와 쓰기할 테스트 데이터를 각각 보내주며, 읽기한 데이터는 Comparator를 통해 비교하게 된다. Comparator는 datapath I/O와 연결되어 있으며 encoder는 comparator와 연결하여 마이크로프로그래밍 롬에서 생성한 테스트 패턴과의 비교 결과를 바탕으로, encoder를 거쳐 3개의 출력을 통해 오류의 발생여부와 오류가 발생한 블록을 동시에 확인 할 수 있도록 하였다. 이때 Next address generator는 블록 각 I/O 비트 위치에 대해 서로 다른 테스트 데이터 쓰기를 정상 쓰기와 동일하게 계속 할 수 있다.

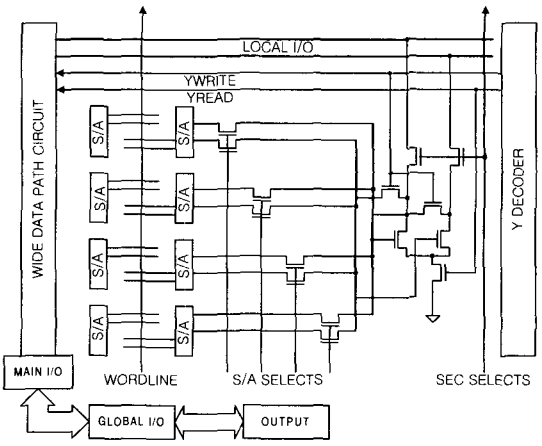


그림 2. TI DRAM을 바탕으로 한 MTFL의 전체 구조
Fig. 2. Overall architecture of MTFL based on TI DRAM.

이후, 읽기 시에는 블록내의 같은 비트 위치 데이터는 한꺼번에, 또한 모든 비트의 위치에 대해서는 병렬적으로 읽는 방식을 사용하여 Multi 비트 블록 단위의 오류발생 여부와 위치를 확인한다. 이와 같은 읽기/쓰기는 정상적인 읽기/쓰기와 동일한 동작으로 별도의 회로를 요구하지 않는다. 이와 같은 방식을 사용하여 각 블록과 비트 위치마다, 여러 형태의 테스트 데이터에 대한 테스트가 수행 가능하다. 만약 어느 한 블럭에서 오류가 발생했을 때에 대하여 오류가 발생한 비트의 위치를 블럭 내에서만 순차적으로 추적가능하게 하여 테스트시 걸리는 시간을 크게 줄일 수 있도록 하였다. 이에 더 나아가 Multi 비트 블럭에서 1개, 또는 2개 이상의 셀에서 오류가 발생하였는지 여부도 DQ 핀을 통하여 확인이 가능하도록 하여 효율적인 redundancy 프로그래밍이 가능하도록 하였다. 본 연구결과는 최근에 중요성이 급증하고 있는 프로세서에 사용되는 Embedded DRAM에도 적용 가능하다. 이 경우, 프로세서를 제어 조직으로 사용 가능하다.

DRAM에는 여러 가지 아키텍처가 있으므로 본 논문에서는 단지 redundancy 대체회로만 있는 TI의 64 MEG DRAM [4]를 바탕으로 본 논문에서 제안하는 MTFL을 설계하였다. 여기서 설명된 MTFL은 여러 가지 다른 DRAM 아키텍처 또는 다른 집적도의 DRAM을 위한 테스트 방법으로 용이하게 변형, 확장 될 수 있다. <그림 2>는 TI DRAM의 전체적인 아키텍처를 나타내었다. 전체 칩은 4개의 16Mb quadrant (quad)로 구성되어 있고 각 quad는 2개의 octan으로 구성되어 있다. 각 octan은 다시 17개의 sense amp bank로 나뉘어져 있으며, 각 sense amp bank는 512Kb의 셀로 구성되어 있다. 셀 출력은 sense amp로 연결되며, 4개의 sense amp가 1개의 differential amp (DA)로 연결된다.

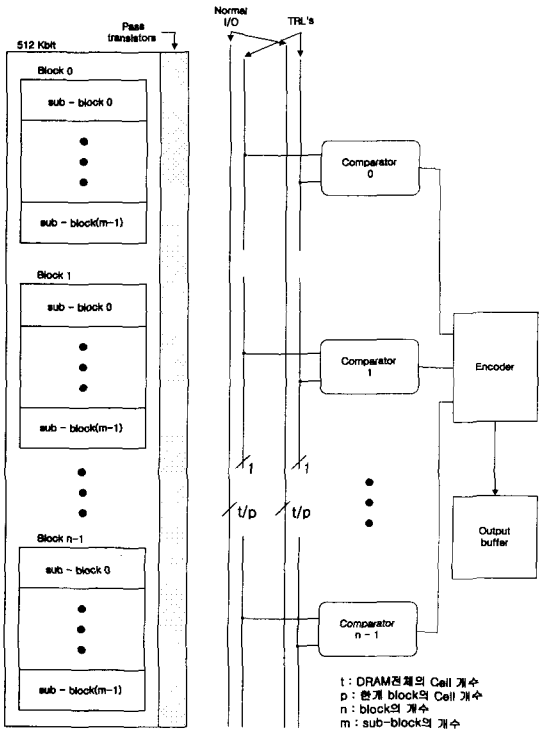


그림 3. 셀 블럭에서의 MTFL의 전체적인 구조
Fig. 3. Overall structure of MTFL in cell blocks.

Octan과 octan 사이에 local sense amp가 있어 DA로부터의 신호를 증폭시켜 칩 중앙부로 보내게 된다. MTFL은 크게 comparator와 encoder로 구성되어 있다. Comparator는 octan과 octan 사이의 local sense amp 옆에 위치하며 각 quad마다 1개의 encoder가 칩 중앙부에 위치한다.

<그림 3>은 <그림 2>에서 한 개의 sense amp bank에서의 MTFL 구조를 나타낸다. 여기서 일반적인 경우를 위해 한 개의 sense amp bank를 n개의 블록으로 나누고, 각 블록을 또 m개의 서브 블럭으로 나눈다고 가정한다. I/O는, 크게 정상 I/O와 Test readout line (TRL)으로 나뉘어져 있으며 각 블록마다 독립된 TRL을 갖고 있다. 각 TRL은 독립된 comparator를 갖고 있으며 한 quad안에 있는 모든 comparator의 출력은 encoder로 연결된다. 각 quad의 encoder 출력은 최종적으로 출력 버퍼로 연결되어 테스트 출력을 칩 바깥으로 출력한다.

MTFL 각 부분의 동작을 자세히 설명하면 다음과 같다. 테스트시에는 먼저 n개의 서브 블럭, 즉 일반적으로 각 블록의 모든 p ($1 \leq p \leq m$) 번째 서브 블럭에 동일한 데이터를 쓰기한 후, 각 서브 블럭들의 데이터를 동시에 n개의 comparator를 통해 읽기한다. 따라서 많은 수의 비트를 동시에 테스트할 수 있다. 여기서 만일 정상 I/O만을 통해 테스트할 경우 [7]과 같이 각 서브 블럭에서 읽혀진 n개의 데이터가 통합되므로 n개 서브 블럭 전체에서의 오류 발생 유무는 알 수 있으나, faulty 비트의 위치는 알 수 없다. 이러한 문제점을 해결하기 위해 n개의 블록 단위로 끊어진 TRL를 두고 정상 읽기 시에는 정상 I/O로, 테스트 읽기 시에는 TRL로 multiplex할 수 있는 pass transistor를 통해, n개의 각 블럭을 독립적으로 테스트한다. 그러므로 각 블럭의 p번째 서브 블럭에 있는 여러 개의 비트 line pair들이 독립적으로 TRL에 의해서 통합되어진다. 만일 서브 블럭에 오류가 있다면 comparator는 비정상임을 출력한다. 이 n개의 comparator 출력은 n 서브 블럭 전체의 오류가 없거나, n 서브 블럭중 한 개의 서브 블럭에서 오류 발생시 오류가 발생한 서브 블럭의 위치, n 서브 블럭중 두 개 이상의 서브 블럭에서의 오류 발생 유무를 나타낼 수 있는 encoder로 입력된 후, 3가지 상태 (0(low), 1(high), Hi-Z)를 출력할 수 있는 출력 버퍼를 통하여 위의 테스트 결과를 출력한다. Redundancy의 양이 충분하여 두 개 이상의 서브 블럭 오류도 대체 가능할 때는 두 개 이상의 서브 블럭 위치를 동시에 알 수 있도록 용이하게 확장될 수 있다. Faulty 비트를 redundancy로 교체할 때, 만일 redundancy 블록과 서브 블럭의 크기가 같다면, 오류가 발생한 서브 블럭을 곧 바로 redundancy로 대체할 수 있다. 만일 서브 블럭의 크기가 redundancy 블록 보다 크다면, 테스트시간은 감소하나, 오류가 발생한 서브 블럭을 곧바로 redundancy로 대체하지 못하고, 오류가 발생한 서브 블럭을 순차적으로 테스트하여 위치를 파악한 후 redundancy로 대체할 필요가 있다.

블럭의 수 (n)와 서브 블럭의 수 (m)에 대해 고찰해 보면, n이 증가하면 테스트에 필요한 시간이 감소한다. 그러나 comparator와 comparator와 encoder를 연결시키는 interconnection의 양이 증가하여 테스트로 인한 면적을 증가시킨다. 또한 m이 증가하면 해상도가 증가하여, 서브 블럭에서 오류 발생시 단시간에 faulty 비트를 추적할 수 있으나, 각 블럭 내의 서브 블럭 단위로 직렬적으로 테스트가 수행되므로 전체 칩을 모두 테스트하는데 걸리는 시간이 증가하게 된다. 따라서 DRAM의 집적도에 따라 n, m을 적절히 조절 가능하며, 이에 관해서는 V장에서 후술된다.

Ⅲ. MTFL 회로설계

<그림 4>에서는 한 개의 블럭에 해당되는 MTFL의 회로 구조를 나타내었다. 이 부분은 DRAM 칩 내에서 많이 중복되는 부분으로 하드웨어 양을 최소화시킬 필요가 있다. 우선 정상 읽기 시에는 sense amp select (SS)에 의해서 한 개의 DA에 연결되어 있는 4개의 sense amp 출력중 하나가 선택되어진다. 이때 column decoder selection (CD)에 의해 한 개의 DA가 선택되고, M2, M3 등의 두개의 pass transistor를 통하여 DA의 출력이 정상 I/O로 연결되어 출력된다. 테스트 읽기 시에는 n개의 각 블럭 내의 선택된 서브 블럭들이 동시에 동작한다. 그러므로 정상 읽기 시와는 달리 서브 블럭에 포함되는 모든 CD를 동시에 동작시킬 필요가 있다. 서브 블럭 단위의 테스트 해상도를 갖기 위하여, 각 블럭마다 모든 DA의 출력을 별도의 TRL에 M1, M4 등의 pass transistor에 의해서 연결한다. 따라서 n개의 블록에서 동시에, 그리고 서로 독립적으로 테스트가 가능하여진다.

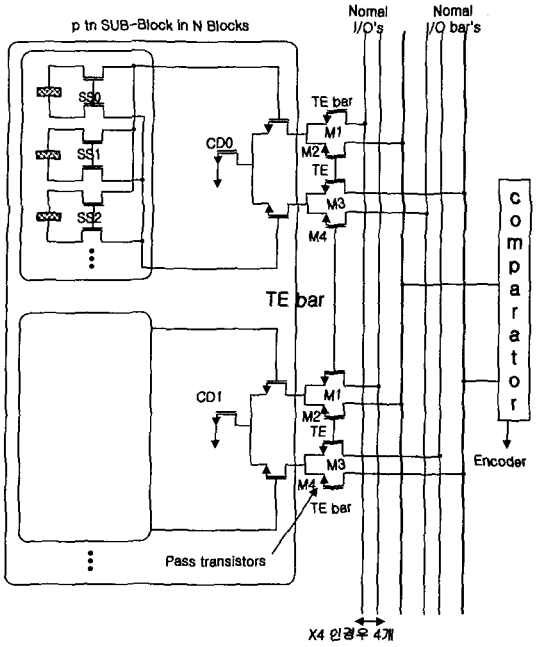
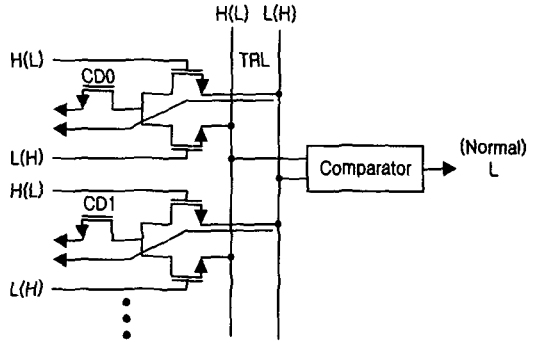
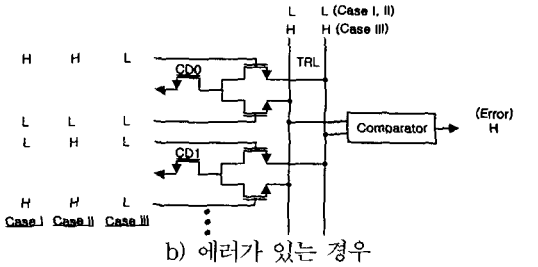


그림 4. 셀 서브 블럭에서의 MTFL의 구조
Fig. 4. The structure of MTFL in cell sub-blocks.



a) 에러가 없는 경우

a) The case of no error



b) 에러가 있는 경우

b) The case of errors

그림 5. 서브 블럭의 테스트 동작 원리
Fig. 5. The operations of test in the sub-block.

<그림 5>에 서브 블럭 테스트 동작 원리를 나타내었다. <그림 5(a)>에서와 같이 테스트되어지고 있는 서브 블럭에 에러가 없다면 테스트 데이터가 1 또는 0에 관계없이 두 개의 TRL중 하나는 high가 되고 나머지는 low가 되어 comparator로 입력된다. <그림 5(b)>와 같이 테스트 되어지고 있는 서브 블럭에 에러가 있다면 TRL pair 모두가 low 또는 high가 되어 comparator에 입력된다. 즉 동일한 data-in에 대해 반대의 출력을 나타내는 경우 (case I), 어느 한 비트 line pair 모두가 high인 경우 (case II), 모든 비트 line pair가 low인 경우 (case III) 등의 에러를 탐지할 수 있다. 위의 세 가지 경우 에러를 모두 탐지함으로 인해 셀뿐만 아니라 sense amp, DA의 에러도 테스트 가능하다. <그림 5(b)>에 나타난 동작 원리는 서브 블럭에서 한 개 이상의 비트 line pair에서 에러가 발생해도 동일하다. 정상 쓰기 또는 테스트를 위한 병렬 쓰기는 단지 동작하는 CD 라인의 수만 차이가 있고, 정상 I/O를 통해 기존의 병렬 쓰기방법을 사용할 수 있으므로 여기서는 자세한 설명을 생략하기로 한다.

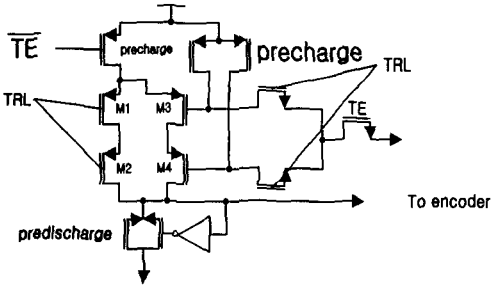


그림 6. Comparator
Fig. 6. Comparator.

<그림 6>에 comparator의 회로를 나타내었다. Compa- rator는 에러가 없다면 두개의 입력 (TRL)중 하나가 '1'이고 나머지가 '0'이므로, M1, M2중 하나, M3, M4중 하나의 transistor가 각각 off되어 '0'을 출력한다. 에러가 있다면 두개의 입력 모두가 '0' 또는 '1'이 되어 M1, M2 또는 M3, M4 transistor가 모두 on이 되므로 '1'을 출력한다. M3, M4의 입력을 TRL과 직접 연결하지 않고 DA를 통하여 간접적으로 연결한 이유는 만일 M3, M4가 TRL과 직접 연결되어 있고 TRL을 3.3V로 precharge하지 않고 1.65V로 precharge할 경우 M3, M4가 on이 되어 누설 전류가 발생할 수 있기 때문이다. 이를 방지하고 DA의 빠른 동작 특성을 이용하기 위하여 differential amplifier를 거쳐 M3, M4의 입

력하였다. Comparator와 encoder는 zipper CMOS logic을 사용, 하드웨어 양을 최소화 시켰다. n개의 블록마다 달려있는 comparator의 출력은 n 입력 encoder로 입력되어 n + 2 상태 (에러 없음, 한 개의 서브 블럭에서 에러시 에러 위치, 2개 이상의 서브 블럭에서 에러 상태)를 k (log2 (n + 2) ≤ k) 개의 DQ를 이용하여 출력하게 된다. <그림 7>은 n = 8 즉 sense amp bank를 8개의 블럭으로 나누었을 때를 예를 들어, encoder와 3개 DQ 신호를 나타낸다. 여기서 encoder는 각 quad마다 위치하고, 출력 버퍼는 테스트시 multiplexer를 이용하여 4개의 encoder 출력을 시간적으로 다중화 시킬 수 있다. Encoder는 <그림 7>에서와 같이 크게 Hi-Z encoder 와 0/1 encoder로 이루어진다. Hi-Z encoder는 output enable을 통해 출력을 Hi-Z로 만들고, 0/1 encoder는 에러발생블록의 위치를 출력한다. Encoder는 테스트 출력 (O1 - O3)과 테스트의 최종 출력을 Hi-Z로 만들기 위한 2개의 output enable (OE1, OE3), 총 5개의 라인을 통해 출력 버퍼에 테스트 출력을 보내게 된다. 각 블럭에서의 에러 상태에 따라 compa- rator 출력 (C1 - C8), encoder를 통한 최종 칩 테스트 출력 (Dout1 - Dout3)을 <표 1>에 정리하였다.

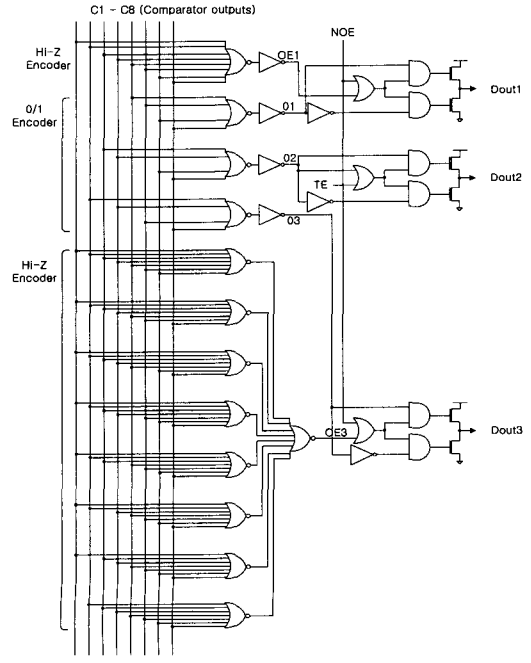


그림 7. 테스트 출력 Encoder 구조
Fig. 7. The structure of the test output encoder.

8개의 블록의 10가지 상태를 8개의 comparator, 1개 '0'이 된다. 그러므로 Dout1은 Hi-Z 상태가 된다. 테스트시 NOE (normal output enable) = 0 이 되므로 <표 1>에서와 같이 Dout2, Dout3의 상태에 관계없이 8개의 블록 모두가 에러 없음을 알 수 있다. 만일 n개의 블럭 중 한 블럭에 에러가 발생할 때는 다음과 같다. 여기서 다섯 번째 블럭에서 에러가 발생하였다고 가정하면 다섯 번째 블럭의 comparator 출력만이 '1'이 된다.즉 C5 = 1 이고 나머지 C1 = C2 = C3 = C4 = C6 =C7 = C8 = 0 이 되어, OE1이 '1'이 되고 OE3 역 패턴시 '1'이 된다. 따라서 3개의 출력 버퍼 모두가 데이터 출력이 가능한 상태가 되며 0/1 encoder의 출력 모두가 출력 버퍼를 통하여 출력될 수가 있게 된다. 따라서 <표 1>에서와 같이 Dout1 = 1, Dout2 = 0, Dout3 = 0을 출력하여 다섯 번째 블럭에서 에러가 발생하였음을 알아낼 수 있다. 만일 n개의 블럭중 두 개 이상의 블럭에서 에러가 발생할 때는 다음과 같다. 여기서는 앞에서 서술한 바와 같이 에러가 발생한 블럭의 위치를 알 필요가 없고 단지 두 개 이상의 에러가 발생한 사실만을 출력하면 된다. 두 번째와 여덟 번째 블럭에서 에러가 발생하였다고 가정하면, C2 = C8 = 1이고 C1 = C3 = C4 = C5 = C6 = C7 = 0이 되어 OE1이 '1'이 되고 OE3은의 encoder, 3개의 DQ 핀을 이용하여 나타낼 수 있는 방법은 다음과 같다. 만일 8개의 블록 모두에서 에러가 없다면, 블록의 comparator 출력은 모두 '0'이 되어 (C1 C2 = C3 = C4= C5 = C6 = C7 = C8 = 0) OE1이 '0'이 되어 Dout3는 Hi-Z 상태가 된다. 따라서 <표 1>에서와 같이 Dout1, Dout2의 상태에 관계없이 두 개 이상의 블럭에서 에러가 발생하였음을 알 수 있다. 일반적으로 DRAM 칩의 DQ의 수는 세계적으로 표준으

표 1. 테스트 출력 구조
Table 1. Test output pattern.

		D_{out1}	D_{out2}	D_{out3}
No error		Hi-Z	don't care	don't care
One error	Block 1	Φ	Φ	Φ
	Block 2	Φ	Φ	1
	Block 3	Φ	1	Φ
	Block 4	Φ	1	1
	Block 5	1	Φ	Φ
	Block 6	1	Φ	1
	Block 7	1	1	Φ
	Block 8	1	1	1
Two error 이상		don't care	don't care	Hi-Z

로 정하여지게 된다. 제안된 테스트 방법은 전혀 부가의 테스트를 위한 핀을 요구하지 않아, 구현비용의 증가를 발생시키지 않는다.

Ⅳ. 시뮬레이션과 레이아웃

1. 회로 시뮬레이션

MTFL I/O 경로의 Hspice 시뮬레이션에서는 DRAM 전체 읽기 경로 대신 본 연구에서 제안된 테스트 회로를 중점적으로 sense amp 출력부터 출력버퍼까지 시뮬레이션 하였다. Sense amp bank를 8개의 블럭으로 나누었으며, 모든 sense amp differential 출력은 0.067V/ns의 slew rate로 변화한다고 가정되었다. Sense amp 출력이 갈라지기 시작하여 약 15ns후에 각 블럭의 p번째 서브 블럭안의 모든 CD가 동작한다. Interconnection 지연을 정확히 시뮬레이션하기 위해

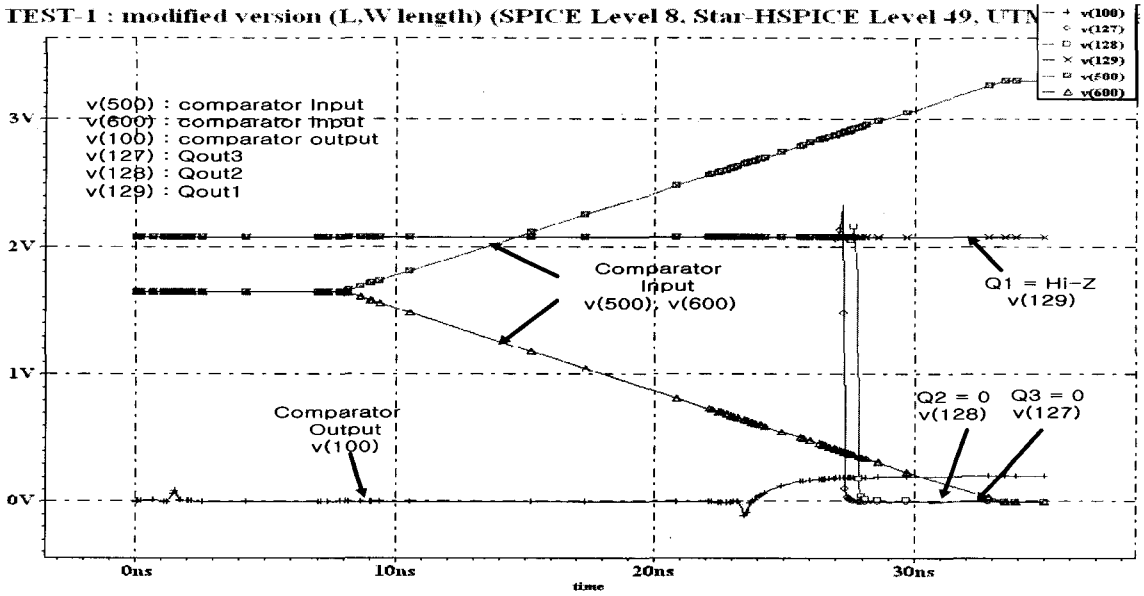


그림 8. 8개의 블록에 모두 에러가 없는 경우
Fig. 8. Simulation results for no error in 8 blocks.

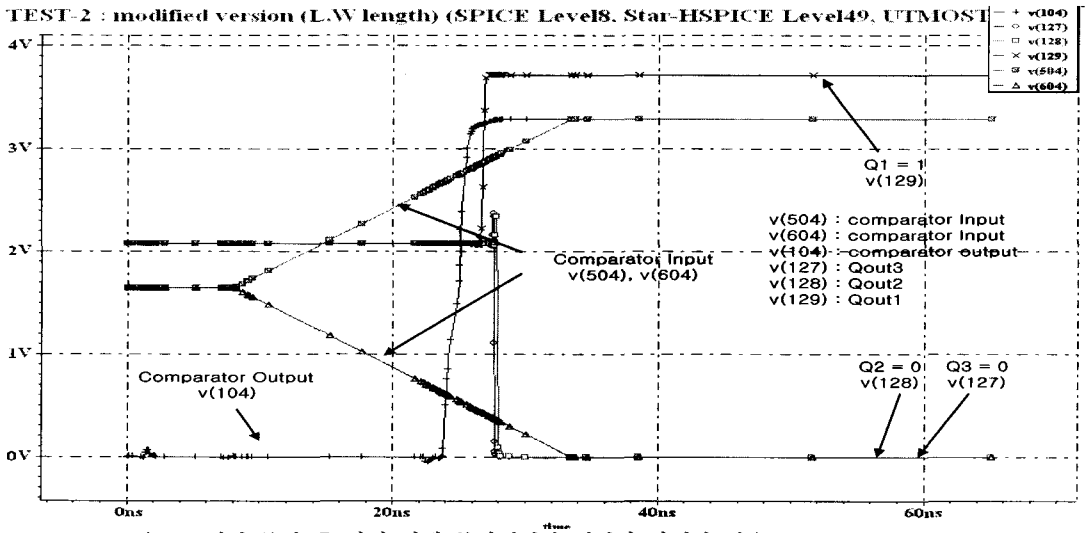


그림 9. 8개의 블럭 중 다섯 번째 블럭에서만 에러가 발생한 경우
Fig. 9. Simulation results for error in 5th block.

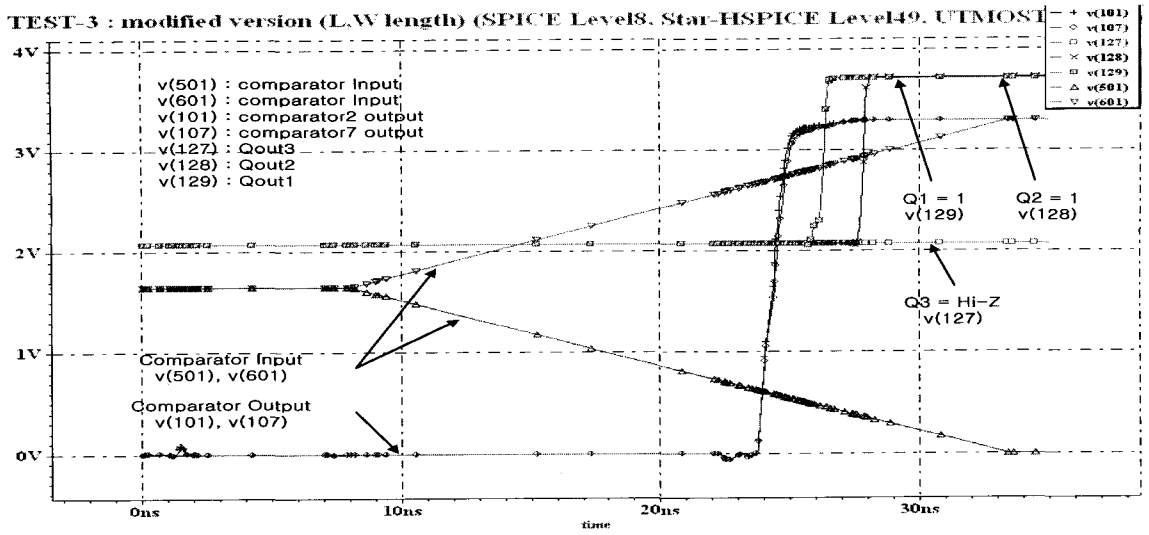


그림 10. 8개의 블럭중 두 번째와 여덟 번째 블럭에 에러가 발생한 경우
Fig. 10. Simulation results for errors in 2nd and 8th blocks.

metal line의 Rsheet = 0.05 Ohm/square, C/μm^2 = 0.037 fF 로하고 네트워크로 모델링 하였다. 테스트를 위하여 부가된 pass transistor에 의해서 정상 읽기시 발생하는 시간 지연을 알아보기 위해 spice 시뮬레이션을 실행하였다. 그 결과 metal line으로 약 4,000μm (quad의 높이) 떨어진 곳에서 약 0.5ns의 지연이 발생하였으나, 이 정도의 시간 지연은 정상 읽기시 큰 문제가 되지 않을 것으로 예상되어진다.

<그림 8>은 8개의 블럭에 모두 에러가 없을 경우의 시뮬레이션 출력을 나타낸다. 8개의 TRL pair마다 서로 complement 상태로 comparator에 입력되어지고 comparator의 출력이 '0'임이 나타나고, 각 comparator의 출력이 encoder로 입력되어 Dout1 (Q1)이 Hi-Z 상태가 됨을 보여주고 있다. Hi-Z 상태는 출력 버퍼의 외부 loading 조건에 의해서 2.08V를 나타낸다. <그림 9>는 8개 블럭중 한 개, 즉 다섯 번째 블럭에서만 에러가 발생하였을 때의 spice 시뮬레이션 결과로써, 다섯 번째 블럭의 TRL 모두가 low가 되고 나머지 TRL은 모두 complement상태로 comparator에 입력되어, 다섯 번째 comparator의 출력만 '1'이고, 나머지 comparator의 출력은 모두 '0'이 됨을 보여주고 있다. 따라서 Dout1 (Q1) = 1, Dout2 (Q2) = 0, Dout3 (Q3) = 0 이 되어 다섯 번째 블럭에 faulty 비트가 위치함을 알 수 있음을 보여주고 있다. <그림 10>은 8개 블럭 중 두개,

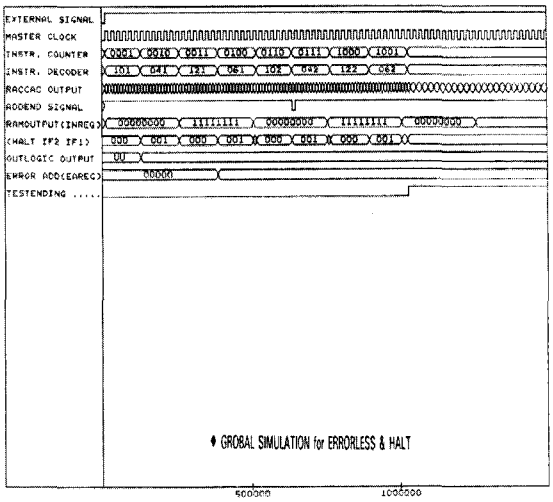


그림 11. MTFL 전체 시뮬레이션 결과
Fig. 11. Overall MTFL simulation results.

즉 두 번 째와 여덟 번째 블럭에서 에러가 발생한 경우의 spice 시뮬레이션 결과로써, 두 번째와 여덟 번째 블럭의 TRL이 모두 '0'이 되고 나머지 TRL pair는 모두 complement상태로 comparator에 입력됨을 알 수 있다. 따라서 Dout3 (Q3)이 Hi-Z 상태가 되어 두 개이상의 오류가 발생했음을 보여주고 있다. <그림 11>은 <그림 1>에 나타낸 MTFL 전체 시뮬레이션 결과이다.

2. MTFL Data I/O path 레이아웃

Spice simulation을 통해 검증한 MTFL Data I/O 경

로를 레이아웃 하였다. <그림 12>와 <그림 13>은 Magic 프로그램을 이용한 comparator와 encoder의 레이아웃으로써, 0.35μm CMOS technology를 사용하였으며, 각각 3,263μm^{2}와 31,424μm^{2}의 크기를 나타내었다. <그림 12>와 <그림 13>의 레이아웃 결과를 바탕으로 MTFL의 면적 증가를 계산하여 보면 다음과 같다.

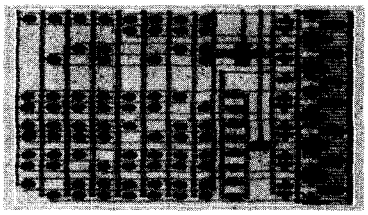


그림 12. Comparator 레이아웃
Fig. 12. The layout of comparator.

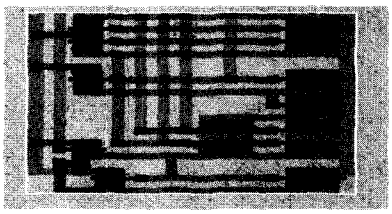


그림 13. Encoder 레이아웃
Fig. 13. The layout of encoder.

각 quad 마다 8개의 comparator와 1개의 8입력 encoder가 필요하므로, DRAM 칩 전체에 MTFL을 구현하기 위한 총 면적은 (3,263 × 8 + 31,424) × 4 = 230,112 μm2가 된다. 따라서 약 2cm × 1cm 크기의 64M DRAM 칩에 MTFL을 적용하였을 때 (230,112 / 2 x 108) × 100 = 0.115 % 가 되어 면적 증가가 극히 적음을 알 수 있다. 여기서 테스트만을 위한 interconnection이 필요로 하는 면적은 극히 적으므로 생략하였다. Magic은 manhattan 스타일의 CAD tool이므로 만일 45° 레이아웃 할 수 있는 CAD tool이 사용된다면 추가 면적은 더욱 감소되어질 것으로 기대된다. 시뮬레이션에서는 Sense amp의 출력으로부터 DA까지의 거리는 약 4,500 μm 로 R = 282 Ohm, C = 133.2 fF로, TRL의 길이는 5,000 μm / 8 = 625μm 로 R = 39 Ohm, C = 18.6 fF로, comparator에서 encoder까지의 길이는 4,687 μm 로 R = 300 Ohm, C = 138 fF로, encoder에서 출력버퍼까지의 길이는 10,000 μm로 R = 624 Ohm, C = 295.8 fF로 시뮬레이션 하였다.

V. 테스트 시간 성능향상 분석

일반적인 DRAM 테스트에서 메모리 셀의 용량에 따라 테스트 시간은 크게 차이가 난다. 예를 들어 N개의 메모리 셀을 갖고 있는 DRAM에 '1'과 '0' 의 데이터를 사용하여 테스트할 경우, 각각에 대해 1회씩 테스트를 해야 하므로 전체 테스트시간은

$$2N \times T_{cell} \quad (1)$$

Tcell : 셀 한 개를 테스트하는데 걸리는 시간
이 된다. MTFL의 성능 분석을 위하여 일반적으로 널리 사용되고 있는바와 같이 메모리 셀을 블록 n개 (동시에 MTFL 테스트할 셀 개수 단위)와 블럭당 m개의 서브 블럭으로 구성되어 있다고 가정한다. 이때 총 테스트시간은,

$$T_{total} = m \cdot T_{group} + \frac{N}{n \cdot m} \cdot T_{cell} \cdot E \quad (2)$$

T_{total} : 전체 테스트시간

T_{group} : 서브 블럭 단위로 블록별로 병렬하게 테스트 시간

E : 병렬 테스트시, 서브 블럭 단위로 오류가 발생한 횟수 = 서브 블럭 serial 테스트 횟수

위 식에서 T_{group} 을 T_{cell} 과 비교하여 α 배의 시간이 소요된다고 할 때, $T_{group} = \alpha \cdot T_{cell}$ 이 되어 (2)는 (3)으로 정리된다. 여러 개의 셀을 병렬 테스트할 경우 약간의 loading증가로 읽기/쓰기시간은 증가하나 이는 미미하여, α값은 거의 1이 된다.

$$T_{total} = m \cdot \alpha \cdot T_{cell} + \frac{N}{n \cdot m} \cdot T_{cell} \cdot E \quad (3)$$

MTFL은 테스트를 두 단계로 나누어서 처음에는 병렬 적으로 테스트를 하여 오류가 발생한 서브 블럭을 알아내고, 다시 오류가 발생한 블록을 서브 블럭(테스트를 위한 셀의 묶음으로 원래 서브 블럭 크기와는 무관하나 본 연구에서는 편의상 동일한 크기로 가정)별로 순차적으로 테스트하게 된다. 이후, 오류가 발생한 셀의 위치를 정확하게 알아내어 redundancy로 치환하게 되므로 1개의 서브 블럭의 셀 개수가 일정할 때, 일정한 셀 용량 N에 대하여, 병렬 처리되는 블럭의 수 n이 증가하면 첫 번째 단계의 테스트시간이 감소하고, 서브

블럭의 수 m이 증가하면 순차적으로 테스트되는 서브 블럭 테스트시간으로 인하여 첫단계 테스트시간이 증가한다. 그러나 faulty 비트 위치 추적을 위해 순차적으로 테스트되는 서브 블럭의 셀의 개수는 n과 m이 증가하면 감소하므로 두 번째 단계의 테스트시간은 감소한다. 한가지 이와 같은 사항의 검증 예로써, 8 비트의 병렬 I/O를 가진 N Mb DRAM을 식 (3)에서 α=1.5, T_{cell} = 80ns (Read access Time + Write access Time)이라고 가정하여, 오류의 발생 횟수 (E)와 DRAM의 용량 (N) 및 서브 블럭의 개수 (m)에 따른 테스트 Time (T)을 <그림 14>에 나타내었다. <그림 14>와 같이 서브 블록의 수와 에러의 개수에 따른 테스트시간의 변화를 잘 나타내고 있다.

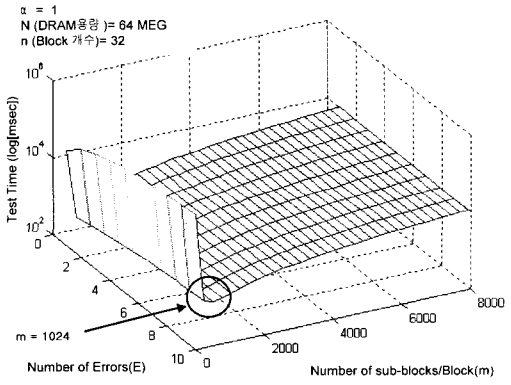


그림 14. 오류발생 횟수(E)와 서브 블럭 개수(m)에 따른 테스트시간

Fig. 14. Test time vs. error(E) and sub-block count (m).

1개의 서브 블럭의 셀 개수가 일정할 때, 에러의 발생횟수 (E)가 증가하거나 블럭당 서브 블럭의 개수 (m)가 너무 적거나 일정량 이상이면 테스트시간이 크게 증가한다. 따라서 적절한 m을 사용하면 테스트시간을 최소화 할 수 있는데, (3)를 m에 대한 미분으로 T_{total}을 최소화할 수 있는 m을 구하면 (4)와 같다.

$$m=\sqrt{\frac{N\cdot E}{n\cdot \alpha}} \quad \text{(단, m > 1 이고 2의 배수)} \quad (4)$$

아래의 <표 2>는 (4)를 사용하여 4회 에러가 발생할 경우를 가정했을 때 DRAM 용량과 블럭의 개수에 따른 테스트시간을 최소화 할 수 있는 m을 나타낸다.

<표 2>에서 예를 들어, 블럭 개수 (n)가 8 이고 DRAM의 용량 (N)이 64 MEG일 때 (1)를 사용하여 1 비트씩 테스트할 때와 m을 최적화 하여 MTFL을 사용했을 때의 테스트시간을 비교하면 <그림 15>와 같다. 즉 1 비트씩 테스트를 하면 테스트시간이 0.8 sec가 소요가 되고 오류가 4개 발생했을 경우를 가정하여 MTFL 테스트를 하면 1.1 ms가 걸렸다. 즉 MTFL을

표 2. 테스트시간을 최소화 할 수 있는 m

Table 2. m values to minimize test time.

DRAM 용량(N) \ 블럭개수(n)	4	8	16	32
64 MEG	4096	4096	2048	1024
256 MEG	8192	4096	4096	2048
1 G	16384	8192	4096	4096

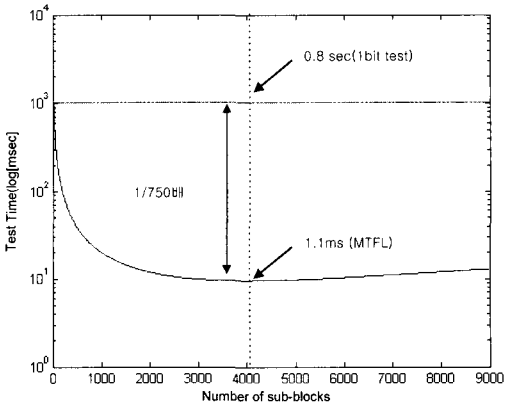


그림 15. 블록의 개수가 8개인 64MEG DRAM의 경우 single bit과 MTFL 테스트 시간비교

Fig. 15. The comparison of the test time between single bit and MTFL test of 8 blocks for 64M DRAM.

적용했을 때 단일 비트씩 테스트할 경우와 비교하여 최대 테스트시간은 0.1375%에 해당하는 약 1/750로 감소한다.

VI. 결 론

셀 어레이 블럭을 서브 블럭으로 나누고 여러 개의 블럭내의 서브 블럭에 대해 병렬 multi 비트 테스트를 수행함과 동시에, faulty 비트의 수와 위치를 병렬 적으로 추적할 수 있는 MTFL 테스트 방법이 제안되었다. 이 방법에 의해 redundancy 프로그래밍에 필요한 정보는 테스트 종료후 즉시 용이하게 알 수 있다. 마이크로 프로그래밍 방식의 MTFL Controller와 MTFL 경로

공유를 위한 정상 I/O와의 multiplexing, comparator, encoder, 출력 버퍼가 설계되었다. 검증 및 레이아웃이 수행되었으며, 테스트 시간 성능이 비교, 분석되었으며 8 블록으로 이루어진 64MEG DRAM의 경우, 0.115%의 회로 증가로 1/750의 큰 테스트 시간 감소를 얻을 수 있었다.

참 고 문 헌

[1] S. Tanoi, Y. Tokunaga, T. Tanabe, K. Takahashi, A. Okada, M. Itoh, Y. Nagatomo, Y. Ohtsuki, M.Uesugi, "On-Wafer BIST of a 200Gb/s Failed-Bit Search for 1Gb DRAM", ISSCC97, Vol. 40, pp. 70~71, Feb, 1997.

[2] Jeffrey Dreibelbis, John Barth Jr., Rex Kho, Howard Kalter, "An ASIC Library Granular DRAM Macro with Built-In Self Test", ISSCC98, Vol. 41, pp. 74~75, Feb, 1998.

[3] S. Takase, N.Kushiyama, "A 1.6GB/s DRAM with Flexible Mapping Redundancy Technique and Additional Refresh Scheme", ISSCC99, Vol. 42, pp. 410~411, Feb, 1999.

[4] K. Komatsuzaki, S. Sukegawa, K. Fung, T. Inui, T. Suzuki, R. Rountree, J. You, B. Borchers, T. Komatsuzaki, H. Shichijo, H. Tran, D. Scott, "Circuit Technique for wide-word I/O path 64MEG DRAM", VLSI Symposium, Japan, May, 1991.

[5] P. Mazumder, "Parallel Tes ting of Parametric Fault in a Three Dimensional Dynamic Random-Access Memory", IEEE JSSC, Vol. 23, No. 4, pp. 933~942, Aug., 1988.

[6] Y. Nishimura, M. Hamada, H. Hidaka, H. Ozaki, K. Fujishima, "A Redundancy Test-Time Reduction Technique in 1-Mbit DRAM with a Multibit test Mode", IEEE JSSC, Vol. 24, No. 1, pp. 43~49, Feb. 1989.

[7] S. Mori, H. Miyamoto, Y. Morooka, S. Kikuda, M. Suwa, M. Kinoshita, A. Hachisuka, H. Arima, M. Yamada, T. Yoshihara, S. Kayano, "A 45ns 64Mb DRAM with a Merged Match-Line Test Architecture," IEEE J. Solid-State Circuits, vol. 26, pp. 1486~1492, Nov. 1991.

[8] K. Arimoto, K. Fujishima, Y. Matsuda, M. Tsukude, T. Oishi, W. Wakamiya, S. Satoh, M. Yamada, T. Nakano, "A 60-ns 3.3-V-Only 16-Mbit DRAM with Multipurpose Register," IEEE JSSC, Vol. 24, pp. 1184~1190, Feb., 1989.

저 자 소 개

柳 在 熙(正會員)

1963年 3月 서울 출생. 1985年 2月 서울대학교 전자공학과 학사. 1987年 8月 Cornell 대학교 전기공학과 공학석사, 1990年 2月 Cornell 대학교 전기공학과 공학박사. 1990年 3月~1991年 3月 Texas Instruments. Dallas VLSI Design Lab. MTS. 2000年 2月 ~ 2001年 2月 Global Communication Technologies 기술자문, 2001年 3月~ Primenet 기술자문, 현재 홍익대학교 전자공학과 부교수. <주관심분야 : Multimedia VLSI 아키텍쳐 및 시스템 설계, Home Networking 시스템, DRAM 회로 설계 등임>

河 昌 佑(正會員)

1973年 경남 마산 출생. 2000年 2月 홍익대학교 전자공학과 학사. 2002年 2月 홍익대학교 전자공학과 공학석사. 2002年 3月 ~ 삼성 전기 기술총괄 중앙연구소 ASIC 2팀 연구원. <주관심분야 : Analog ASIC 및 Opto-electronic device 등임>