공학 연구논문집 제28권 제1호 pp. 259~273, 울산대학교.
Journal of Engineering Vol. 28, No. 1, pp. 259~273, 1997
University of Ulsan.

테스트가 용이한 RAM의 설계 및 레이아웃에 관한 연구

강동철 · 조상복
전자공학과

<요 약>

효과적인 고집적 RAM테스팅을 위해 BIST(Built-In Self Test)기법으로 메모리 내부에 테스트 회로를 부가하여 테스트할 수 있도록 하였는데, 이때 고집적 메모리에 가장 효율적인 PSFs(Pattern Sensitive Fault) 검출 알고리듬을 적용하였다. 128가지의 Gray Code의 테스트 패턴과, 타일링을 위한 4개의 기본셀을 바탕으로 한 행에 대해서 전체 비트라인으로 쓰기와 읽기동작을 하고 에러가 발생하면 외부의 핀을 통해 신호를 전달하게 함으로써 메모리셀의 고착고장, 천이고장, ANPSFs(Active Neighborhood PSF), PNPSFs(Passive Neighborhood PSF), 그리고 SNPSFs(Static Neighborhood PSF)를 검출할 수 있다. 가능한 현재 사용되는 메모리의 기본 소자를 사용하여 부가 회로의 면적을 줄이도록 하였다.

A study on the design and layout of easily testable RAMs

Kang, Dong-Chual · Cho, Sang-Bock
Dept. of Electronics Engineering

<Abstract>

For the effective test of highly densed RAMs, the detecting algorithm of PSFs is proposed and a test circuit is designed as an additive circuit inside a memory chip,

* 본 연구는 96년도 울산대학교 학술연구비 지원에 의해 수행되었음

which is often called a BIST(Built-In Self Test) circuit. With 128 gray codes and 4 basic cells for the tiling, read and write operations are performed to entire column lines in each row.

If there is an error, it is detected simultaneously (because the outputs of error detecting circuits are connected with output pins) and the algorithm covers Stuck-at Faults, Transition Faults, ANPSFs(Active Neighborhood PSF), PNPSFs(Passive Neighborhood PSF), and SNPSFs(Static Neighborhood PSF).

Furthermore, the area for the BIST circuit can be minimized by using the existing components.

Ⅰ. 서 론

VLSI 기술의 발전으로 RAM의 집적도는 더욱 높아지고 있으며 컴퓨터 보급 확대 및 발전에 따라 그 사용도 갈수록 많아지고, 이에따라 RAM의 중요성이 더욱 커지고 있다. [1~4] 국내 반도체 산업은 세계 3위에 육박하고 있으며 이 분야에서 RAM의 위치는 절대적이라 할만큼 중요하다. 이렇게 중요한 RAM에 있어 국내에서 공정 수준은 미국 일본에 근접해가고 있으나, 이에 비해 설계 분야는 상대적으로 뒤떨어져 있는 형편이다. [9]

특히 고집적 RAM에 있어 종래의 고착고장, 결합고장, 천이고장 여러 가지 PSF 고장 등 각종 고장들을 모두 검출할 수 없고, 그러한 고장에 대해 고장 검출이 용이한 RAM의 테스트 용이한 설계에 관한 분야에 있어서는 상당히 연구가 미진한 형편이다. 특히 RAM이 64M이상으로 고집적화 되면 메모리 셀들이 더욱 근접해 짐으로써 셀간의 전자기적 영향이나 기생 커패시턴스, 전하누설 등으로 다른 셀들의 간섭에 의해 정보가 소실되는 등 고장 및 테스트를 고려한 RAM의 설계방식에 관한 연구가 절실해 진다. [5~8] 따라서 본 연구에서는 이러한 고집적 RAM에서 발생 할 수 있는 여러 가지 고장을 쉽게 검출할 수 있는 알고리즘을 개발하고 이에 대한 회로를 설계한다. 설계에 있어 부가회로 면적을 줄이고 테스트를 쉽게 하도록 하여 설계의 효율성을 최대로 진작시키도록 하였다.

PSF 검출을 위한 회로는 크게 TPG(Test Pattern Generator), CSL(Column Select Loader), PTS(Parallel Tile Selector), RAG(Row Address Generator), ED(Error Detector) 등으로 구성된다.

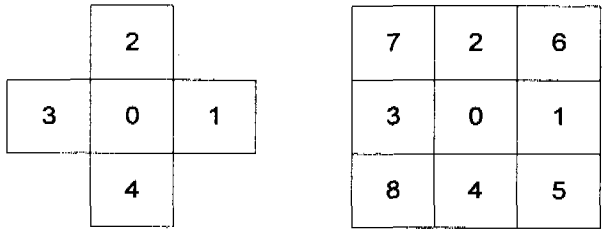
Ⅱ. PSF 검출 알고리즘

2.1. 메모리셀 타일링

기존의 이웃셀 설정형태는 그림1과 같이 type1 방식과 type2 방식이 있는데 type1은 기본셀을 포함하여 5개의 이웃셀을 가지며, type2는 9개의 이웃셀을 가진다. 이러한 셀에 대하여 일정한 형태의 테스트 패턴을 가하여 PSF를 검출하게 되는데, 이때 일반적으로 Eulerian cycle에 의해 발생된 패턴을 각 이웃셀들에 가하게된다.

테스트 패턴의 가짓수는 각 type마다 다르며 $m \times 2^m$개(m:이웃셀 수)이다. 따라서 type1 방식일 경우 총 160개이며, type2 방식일 경우 4608개의 테스트 패턴이 필요하다.

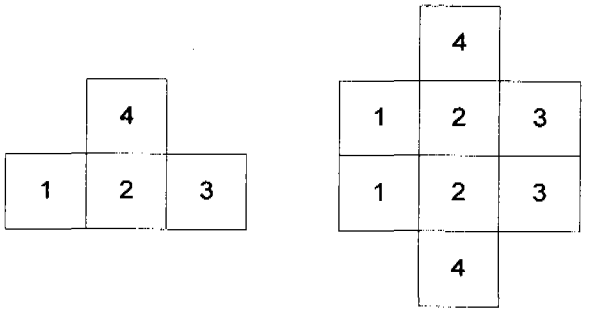
이때 똑같은 테스트 시퀀스를 가할 때 type2 방식은 type1 방식에 비해 288배 이상의 테스트 복잡도를 가진다. 물론 고장 검출을 위한 영역은 type1 방식보다 더욱 넓어지겠지만 테스트 복잡도가 너무 커서 type2 방식에 대해 PSF를 고려할 때, 대각으로 존재하는 이웃셀(5~8)에 대해서는 별도의 테스트 알고리즘을 채택하고 있다.



(a) type 1 방식 (b) type 2 방식

그림 1. 기본셀과 이웃셀의 타일링
0:기본셀 1-8: 이웃셀

본 연구에서는 type1 방식보다 테스트 복잡도를 감소시키면서, 누설전류로 인한 PSF의 영향을 더욱 증대시킨 그림2와 같은 타일방식을 제안하였다. 그림2 (a)를 기초로 하여 PSF로 인한 영향을 더욱 증대시킨 효과적인 타일방식이 (b)이다. i-1번째 워드라인과 i+2번째 워드라인에 위치한 셀의 레이블 상태는 i, i+1의 레이블 상태와 달라서 원하는 테스트를 수행할 수 없으므로 전체 타일에 대해 워드라인을 증가시켜 쉬프트 시켜주면 i-1, i+2의 셀도 똑같은 상태를 가질 수 있다. 이에 대한 전체 메모리셀을 set A, set B로 구분하여 그림3에 나타내었다. 제안된 쐐기형 타일에 인가될 테스트 패턴 가짓수는 4개로서 set A, B를 수행함으로서 실질적으로 총 128가지의 테스트 패턴이 발생된다. 이 테스트 패턴은 Eulerian cycle에 의해 유출된 것이다. 표1에 테스트 패턴을 나타내었으며 set A와 B에 똑같은 64개의 테스트 패턴이 가해지게 된다.



a) 쐐기형 타일 (b) 이웃셀을 고려한 타일

그림 2. 쐐기 모양의 이웃셀 타일 방식

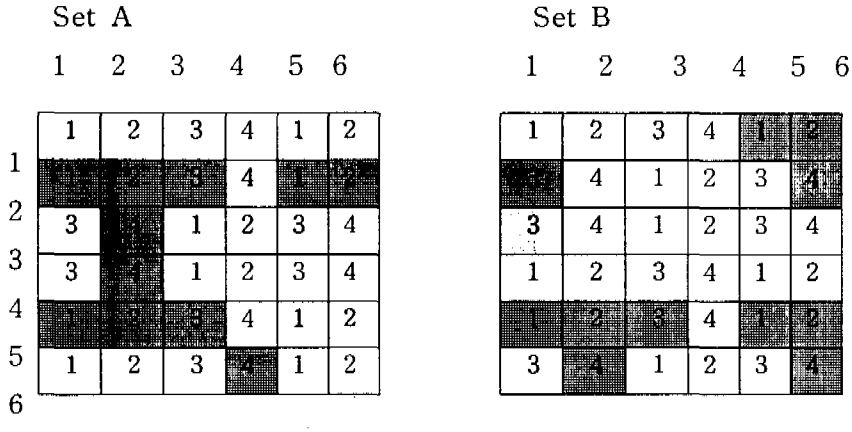


그림 3. 타일링된 6×6 메모리 셀 어레이

표1. Eulerian cycle 에 의해 유출된 테스트 패턴

TP#	B3	B2	B1	B3	TP#	B3	B2	B1	B3	TP#	B3	B2	B1	B3	TP#	B3	B2	B1	B3
1	0	0	0	0	17	1	0	0	1	33	0	1	0	1	49	0	0	1	1
2	0	0	0	1	18	0	0	0	1	34	0	0	0	1	50	0	0	0	1
3	0	0	1	1	19	0	0	0	0	35	1	0	0	1	51	0	1	0	1
4	0	0	1	0	20	1	0	0	0	36	1	1	0	1	52	0	1	1	1
5	0	1	1	0	21	1	0	1	0	37	1	1	0	0	53	1	1	1	1
6	0	1	1	1	22	0	0	1	0	38	1	0	0	0	54	1	1	0	1
7	0	1	0	1	23	0	0	1	1	39	0	0	0	0	55	1	0	0	1
8	0	1	0	0	24	1	0	1	1	40	0	1	0	0	56	1	0	1	1
9	1	1	0	0	25	1	1	1	1	41	0	1	1	0	57	1	0	1	0
10	1	1	0	1	26	0	1	1	1	42	0	0	1	0	58	1	0	0	0
11	1	1	1	1	27	0	1	1	0	43	1	0	1	0	59	1	1	0	0
12	1	1	1	0	28	1	1	1	0	44	1	1	1	0	60	1	1	1	0
13	1	0	1	0	29	1	1	0	0	45	1	1	1	1	61	0	1	1	0
14	1	0	1	1	30	0	1	0	0	46	1	0	1	1	62	0	1	0	0
15	1	0	0	1	31	0	1	0	1	47	0	0	1	1	63	0	0	0	0
16	1	0	0	0	32	1	1	0	1	48	0	1	1	1	64	0	0	1	0

2.2. 알고리즘

이러한 타일 방식에 의한 테스트 알고리즘은 다음과 같다.

1. 첫번째 테스트 패턴(TP#1)을 발생시키고, 테스트 값 B3를 선택
2. 타일내부의 레이블이 1인 셀을 선택

3. 메모리의 첫번째 행을 선택
4. 지정된 레이블의 셀 내용을 선택된 행의 전체셀에 대해 동시에 테스트 값을 기록
5. 1024번째 행까지 한 행씩 증가시켜 가면서 4를 반복
6. 타일내의 모든 셀을 전체 메모리셀에 대해 모두 read
7. 다음 레이블 1→2→3→4의 순으로 셀을 선택하고, 다음 테스트값 B3→B2→B1→B0의 순으로 설정하고 과정 3~5를 반복한다. 레이블이 4이면 다음 테스트 패턴을 발생시키고 2로 간다.
8. 128개의 테스트패턴을 모두 발생했으면 테스트를 종료한다. 그렇지 않으면 2~5를 반복 수행한다.

2.3. 알고리즘의 고장 검출 범위

이상과 같은 알고리즘에 의한 고장 검출 범위는 PSF 각 고장의 종류에 따라 다르다. 알고리즘 4의 단계에서 테스트 값을 쓰고 6의 단계에서 타일내의 모든 셀을 읽어봄으로서 SNPSF 검출을 위한 조건을 만족시킬 수 있다. 쐐기형 타일에 대한 모든 SNPSF에 대한 검출 패턴은 표2와 같으며 타일 내에서 발생하는 모든 SNPSF를 검출할 수 있다.

표2. SNPSF의 검출패턴

기본셀	가능한 모든 패턴	표1의 검출 패턴	기본셀	가능한 모든 패턴	표1의 검출 패턴
1번셀	0 0 1 -	TP#1 - TP#2	2번셀	0 0 - 0	TP#63-TP#64
	0 0 0 -	TP#22- TP23		0 0 - 1	TP#2 -TP#3
	0 1 1 -	TP#5 - TP#6		0 1 - 0	TP#40-TP#41
	0 1 0 -	TP#30- TP#31		0 1 - 1	TP#51-TP#52
	1 0 1 -	TP#12- TP#13		1 0 - 0	TP#20-TP#21
	1 0 0 -	TP#16- TP#17		1 0 - 1	TP#55-TP#56
	1 1 1 -	TP#44- TP#45		1 1 - 0	TP#59-TP#60
	1 1 0 -	TP#9 - TP#10		1 1 - 1	TP#10-TP#11
3번셀	0 - 0 0	TP#39- TP#40	4번셀	- 0 0 0	TP#19-TP#20
	0 - 0 1	TP#50- TP#51		- 0 0 1	TP#34-TP#35
	0 - 1 0	TP#4 - TP#5		- 0 1 0	TP#42-TP#43
	0 - 1 1	TP#47-TP#48		- 0 1 1	TP#23-TP#24
	1 - 0 0	TP#58-TP#59		- 1 0 0	TP#8- TP#9
	1 - 0 1	TP#35-TP#36		- 1 0 1	TP#31-TP#32
	1 - 1 0	TP#43-TP#44		- 1 1 0	TP#27-TP#28
	1 - 1 1	TP#24-TP#25		- 1 1 1	TP#52-TP#53

표3. PNPSF의 검출패턴

기본셀	가능한 모든 패턴	표1의 검출 패턴	기본셀	가능한 모든 패턴	표1의 검출 패턴
1번셀	0 0 1 -	TP#1 - TP#2	2번셀	0 0 - 0	TP#63-TP#64
	0 0 0 -	TP#22- TP23		0 0 - 1	TP#2 -TP#3
	0 1 1 -	TP#5 - TP#6		0 1 - 0	TP#40-TP#41
	0 1 0 -	TP#30- TP#31		0 1 - 1	TP#51-TP#52
	1 0 1 -	TP#12- TP#13		1 0 - 0	TP#20-TP#21
	1 0 0 -	TP#16- TP#17		1 0 - 1	TP#55-TP#56
	1 1 1 -	TP#44- TP#45		1 1 - 0	TP#59-TP#60
	1 1 0 -	TP#9 - TP#10		1 1 - 1	TP#10-TP#11
3번셀	0 - 0 0	TP#39- TP#40	4번셀	- 0 0 0	TP#19-TP#20
	0 - 0 1	TP#50- TP#51		- 0 0 1	TP#34-TP#35
	0 - 1 0	TP#4 - TP#5		- 0 1 0	TP#42-TP#43
	0 - 1 1	TP#47-TP#48		- 0 1 1	TP#23-TP#24
	1 - 0 0	TP#58-TP#59		- 1 0 0	TP#8- TP#9
	1 - 0 1	TP#35-TP#36		- 1 0 1	TP#31-TP#32
	1 - 1 0	TP#43-TP#44		- 1 1 0	TP#27-TP#28
	1 - 1 1	TP#24-TP#25		- 1 1 1	TP#52-TP#53

마찬가지로 표3은 PNPSF에 대한 검출 패턴이며, 이 패턴으로 모든 PNPSF를 검출할 수 있다.

타일내의 임의의 셀에 대해 테스트 값을 쓰고 난 후에 타일내의 모든 셀을 읽어봄으로서, 테스트 값이 기본셀에 영향을 준 ANPSF를 검출할 수 있다. 이는 알고리즘의 4 단계를 마치고 6단계를 수행함으로써 가능하다. 진이웃셀의 가능한 모든 패턴에서 앞의 과정을 수행함으로서 모든 ANPSF를 검출할 수 있으며 가능한 모든 패턴은 앞의 표3과 같다.

첫번째 테스트 패턴이 0000이었고 두번째 테스트 패턴이 0001이었다면 실제로는 제일 하위 비트만 0→1로 변화한 결과이지만, 첫번째 테스트 패턴으로 테스트시 최하위 비트에서 최상위 비트까지 모두 write-read를 수행하고, 다음 두번째 테스트 패턴 또한 최하위에서 최상위 비트까지 모두 write-read를 수행하므로 transition operation되지 않는 세 비트에서 시간 경과에 따른 고장이 발생했다 하더라도 모두 검출 가능하다. 또한, transition operation된 최하위 비트는 다음번째 테스트 패턴 테스트 시에 모두 검출할 수 있으므로 지연 PSF고장의 검출도 가능하다.

이상의 패턴은 기존의 알고리즘에서 검출하는 고착고장과 천이고장을 당연히 검출할 수 있다.

Ⅲ. 반도체 RAM의 PSF를 자체 테스트하는 회로 설계

기존의 RAM에 부가적으로 TE(Test Enable)단, TPG(Test Pattern Generator)단, 에러 검출단 외에 BIST회로의 각 블럭들을 제어해주는 제어부가 있고 병렬 동작단으로 PTS(Parallel Tile Selector), CSL(Column Selector Loader), 및 RAG(Row Address Generator)등의 회로를 부가함으로써 알고리즘을 실현할 수 있다. 테스트 신호가 RAS와 CAS 신호에 의해 인가되면 제어부에 의해 테스트 패턴을 발생시키고, 동시에 RAG와

PTS 및 CSL을 제어한다. TPG에서는 64개의 테스트 패턴을 순차적으로 발생하며 각 패턴을 연속적으로 선택하여 메모리셀에 병렬 쓰기 동작을 PTS에 의해 행하게 된다. 이때 RAG는 각 행을 알고리즘에 맞게 하나의 행씩 순차적으로 선택하는 신호를 발생한다. 이때 행 디코더는 기존의 디코더를 사용한다. 그림4와 5는 전체 BIST scheme을 서두로직의 MyCAD로 구현한 회로 및 시뮬레이션 결과이다. 편의상 256RAM이라 가정하고 회로를 완성하였고, Error Detector 회로 추가와 대용량 RAM블락을 테스트하기 위해 다시 Workstation용 tool인 COMPASS로 다시 구현하였다.

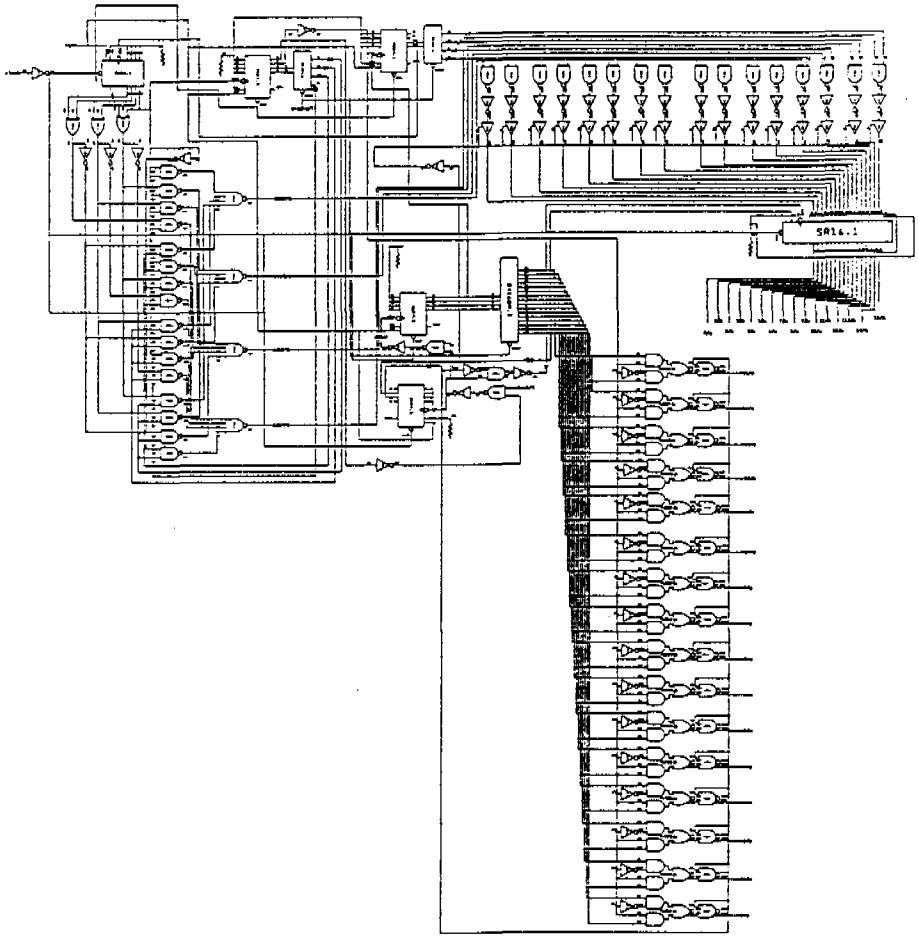


그림 4. 전체 회로도

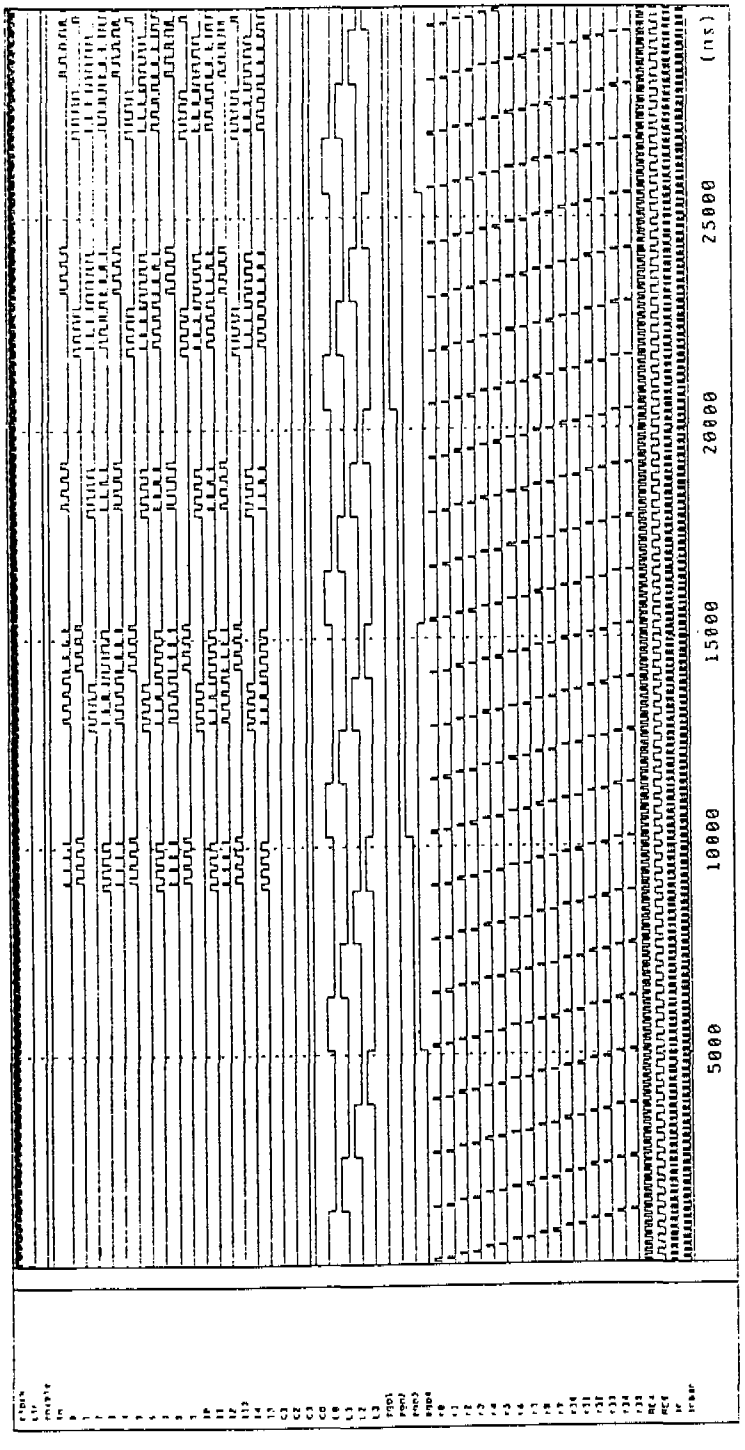


그림 5. 시뮬레이션 결과

3.1. 테스트 모드

메모리 테스트 모드로 들어가기 위해 부가적인 테스트핀을 따로두지 않고 기존의 입력핀을 이용하는 방법을 적용했다. 즉, RAS와 CAS의 입력핀을 사용하였다. 정상메모리 동작시에는 RAS신호가 CAS신호보다 먼저 LOW로 떨어진다. 그러나 테스트모드로 들어가려면 CAS신호가 RAS신호보다 먼저 LOW로 떨어지게 하며 그림6은 회로 및 타이밍 다이어그램을 보여 주고 있다.

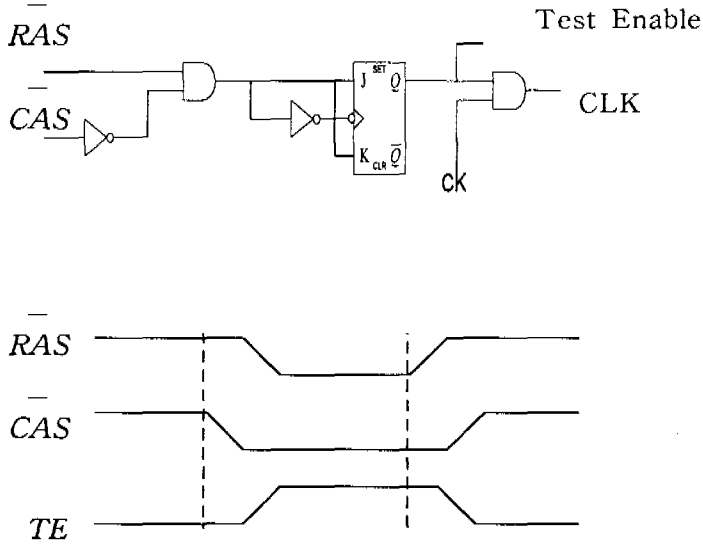


그림 6. 테스트 enable 회로 및 타이밍 다이어그램

3.2. 테스트 패턴 발생기(TPG)

TPG는 Eulerian cycle에 의해 유출된 64개의 테스트 패턴을 발생시키는 시퀀스 발생기와 시퀀스 콘트롤러로 구성되며 그림7과 같다.

그림7은 SC(Sequence Controller)와 TPG를 연결한 것이며 왼쪽 상단부분이 SC단이다.

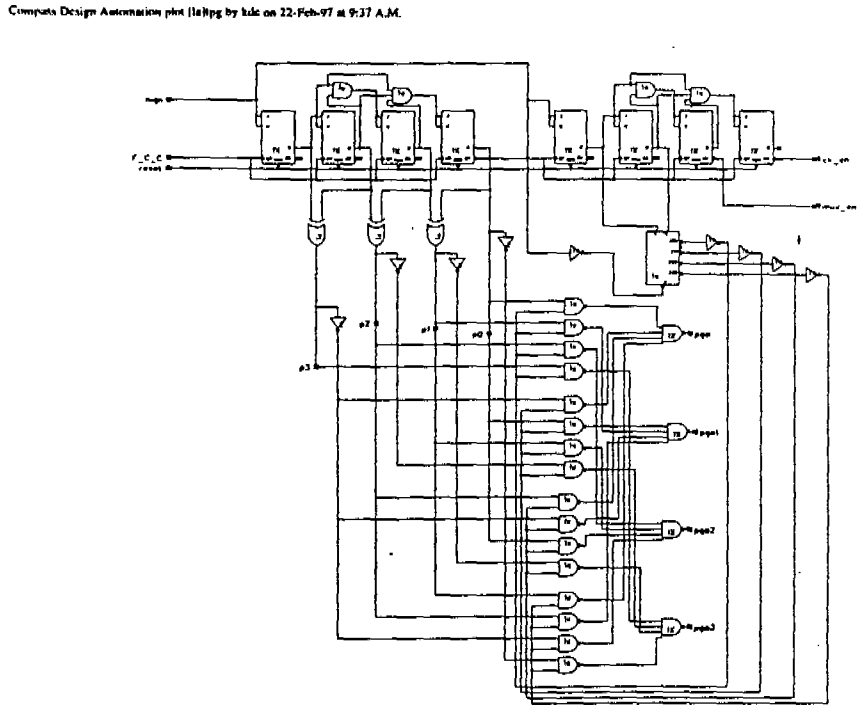


그림 7. SC, TPG 회로도

3.3. 병렬 타일 선택기(PTS)와 행 선택기(CSL)

그림8과 9는 각각 PTS의 동작개요도 및 회로도이다.

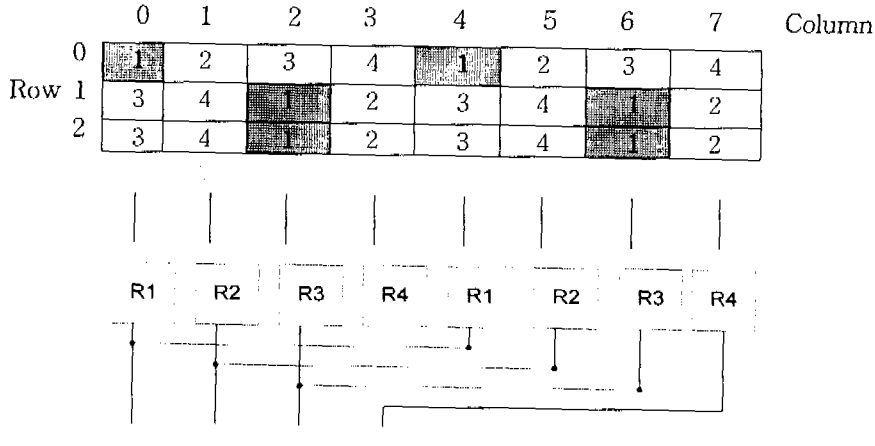


그림 8. PTS의 동작 개요도

PTS는 행 어드레스 디코더에 의해 선택된 행의 셀들에 대하여 병렬로 write 동작을 수행하도록 한다. 즉, PTS는 m개의 쉬프트 레지스터들로 구성되는데 메모리 배열에서 열(비트선)의 수가 된다. CSL로부터 출력된 데이터들은 PTS의 데이터들을 비트라인으로 enable 시키는데, 예를 들어 첫번째 행이 선택되어지면 CSL로부터 발생된 데이터들을 각각의 PTS에서 로드하여 원하는 비트라인만을 선택해 준다. 이와 같은 CSL로부터의 데이터 로드는 모든 열이 한번씩 선택되어질 때까지 오직 한번의 로드만 있을 뿐이다.

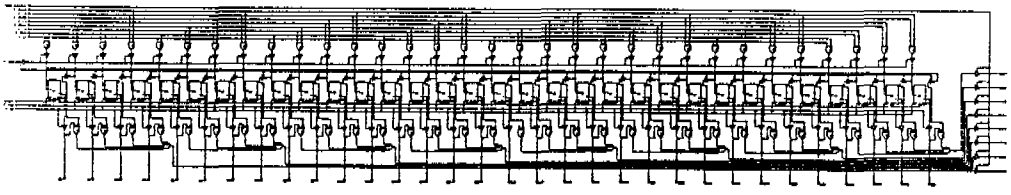


그림 9. PTS의 회로도

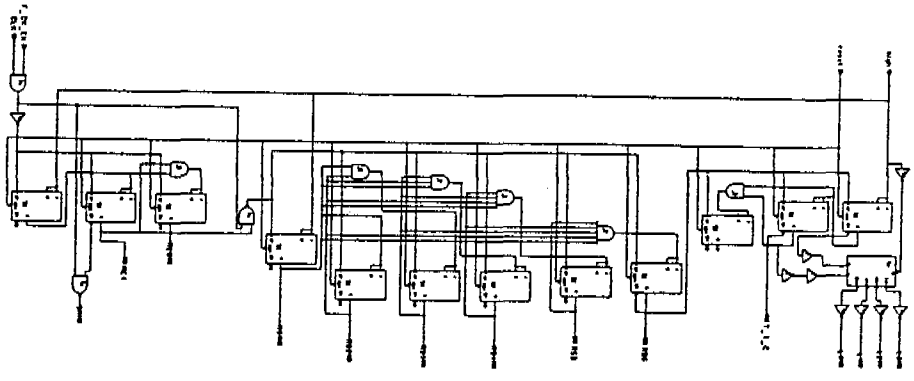


그림 10. CSL의 회로도

그림10은 CSL의 회로이며 메모리 배열의 행을 선택해주는 역할을 한다. CSL의 클럭은 행 선택기의 RS6에 의해서 동작된다. 즉, RS6의 신호에 의해 각각의 데이터 패턴을 발생시킨다. 예를 들어 첫번째로 행 0이 선택되어질 때 CSL로부터 첫번째 행의 1로 타일링된 셀을 선택하기 위한 1000의 데이터를 발생시켜 주며, 두번째로 첫번째 행의 2로 레이블된 셀을 선택하기 위한 0100의 데이터를 발생시켜준다. 이와같이 각 레이블된 셀에 대해 그 셀을 선택해주는 데이터를 각각 발생시켜준다.

3.4. 에러 검출기(ED: Error Detector)

간단하게 쓰기동작의 값과 읽기동작의 값을 배타적 논리합하여 "1" 이면 에러임을 알 수 있다. PTS 회로 상단 우측 부분이 ED단이다

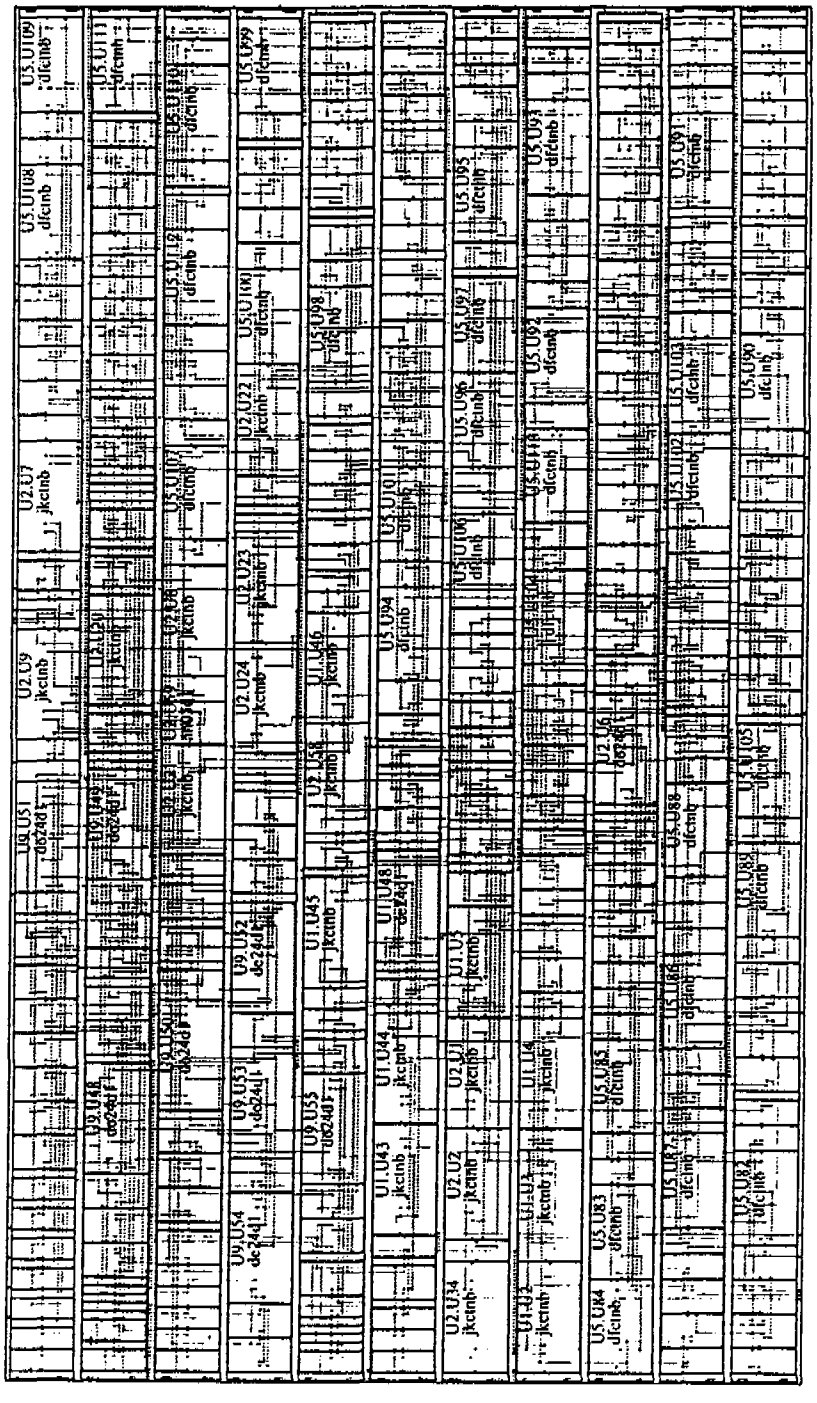
Ⅳ. 회로의 레이아웃

각 회로를 부분별로 레이아웃을 하였고 COMPASS 툴상에서 CMPSC6UL3B 테크날리지 라이브러리, 0.6μm 공정 Phantom 레벨로 레이아웃을 하였다. 그림11은 부분 부분을 모두 결합한 전체 레이아웃이다. chip 크기는 2570×1458 lambda, 사용된 트랜지스트는 4760개이며 그림12는 이에 대한 시뮬레이션 결과이다.

Ⅴ. 결 론

효과적인 고집적 RAM 테스팅을 쉽게 할 수 있도록 BIST scheme에 의한 테스트 알고리즘 및 회로를 제안하였다. 제안된 방법은 종래의 고착고장, 천이고장 뿐만 아니라 고집적 메모리의 가장 효과적인 고장 모델인 PSF 고장 모델을 사용하였다. PSF 고장 모델을 사용하는데 있어 가장 기본인 메모리 셀의 타일링 방법에 있어 새로운 방법을 제안하고, 그에 적합한 알고리즘 및 회로를 제안하여 적은 부가회로로 고집적 RAM을 테스트할 수 있도록 하였다. 부가회로로 TPG, PTS, CSL, RAG, ED 등이 들어갔는데, 이는 전체 알고리즘을 수행하기 위해 모두 새로 설계한 것이다. 각각의 부분회로 및 전체 회로를 로직 레벨로 설계하고 검증하였으며, 또한 설계된 전체 회로를 레이아웃하고 검증하였는데, 이에 대한 전 과정에 서두로직의 MyCAD 와 COMPASS tool을 사용하였다.

설계한 회로는 메모리가 64M이상으로 고집적화 되면 될수록 전체면적에 대해 차지하는 부가회로의 면적이 줄어들어 매우 효과적일 것으로 예상된다. 1G이상의 초고집적 메모리에 적합한 설계로 발전시켜 나가야 하는 것이 앞으로의 과제이다.



Compass Design Automation plot TOTAL by kdc on 25-Feb-97 at 7:29 A.M.

그림 11. 전체 회로의 layout

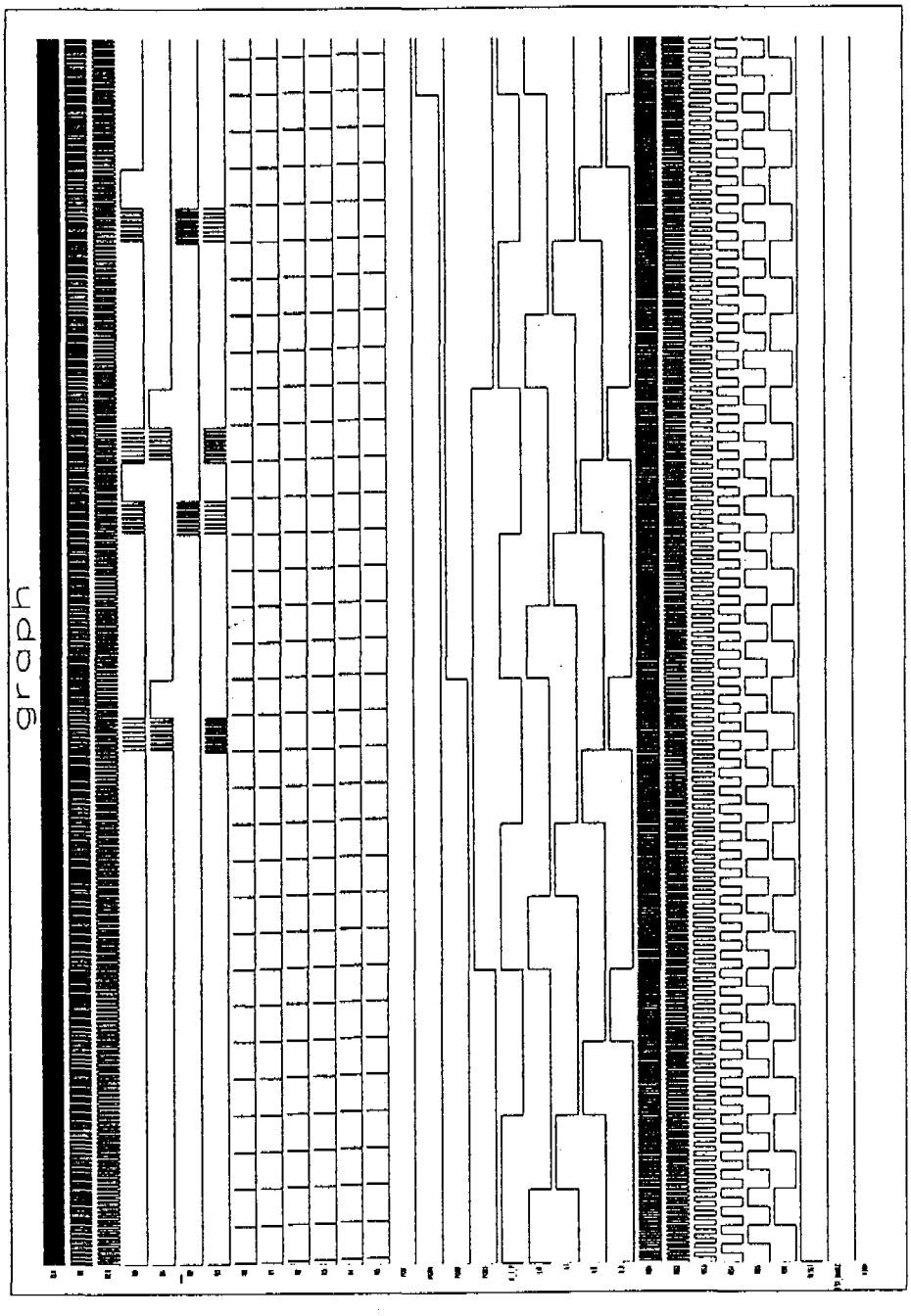


그림 12. 시뮬레이션 결과

참 고 문 헌

[1] N.C.C.Lu, T.V.Rajeevakumar, G.B.Bronner and E.J.Sprogis, "A Buried-trench DRAM Cell Using a Self-aligned Epitaxy Over Trench Technology," IEDM Tech. Dig., 1988.

[2] D.Hisamoto, S.Kimura, and E.Takeda, "A New Stacked Cell Structure for Giga-bit DRAMs Using Vertical Ultra-thin SOI(DELTA) MOSFETs," IEDM Tech. Dig., 1991.

[3] 尹瓚洙, 黃昌圭, "ULSI Cell Technology", KITE Review, vol. 19, no. 5, pp. 343-357, May 1992.

[4] A.J.Van De Goor and C.A.Verruijt, "An Overview of Deterministic Functional RAM Chip Testing," ACM Comp. Sur. vol. 22, no. 1. pp. 5-33, March 1990.

[5] S.Fuij, M.Ogihara, M.Shimizu, et al., "A 45-ns 16-Mbit DRAM with triple-well structure," IEEE J.Solid-State Circuit, vol. SC-24, pp. 1170-1175, Oct. 1989.

[6] P.Mazumder and J.K.Patel, "Parallel Testing for Pattern-Sensitive Faults in Semiconductor Random-Acess Memories," IEEE Trans Comput., vol. 38, no 3, pp. 394-407, March 1989.

[7] 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 no. 6, pp. 419-429, June 1980.

[8] K.K.Saluja and K.Kinoshita, "Test Pattern Generation for API Fault in RAM," IEEE Trans. Comput., vol C-34, no. 3, pp. 284-287, March 1985.

[9] 閔衛植, "DRAM의 발전 방향과 전망", KITE Review, vol. 19, no. 5, pp. 343-357, May 1992.

[10] T.Kirihata, Y.Watanabe, et al.," Fault-Tolerant Designs for 256Mb DRAM," IEEE Jour. of Solid-state Circuits, vol. 31, pp. 558-566, April 1996.