한국시뮬레이션학회 논문지
Vol. 18, No. 4, pp. 59-66 (2009. 12)

수율향상을 위한 반도체 공정에서의 RRAM (Redundant Random Access Memory) Spare Allocation

한영신[1†]

RRAM (Redundant Random Access Memory) Spare Allocation in Semiconductor Manufacturing for Yield Improvement

Young-Shin Han

ABSTRACT

This has been possible by integration techniques such as very large scale integration (VLSI) and wafer scale integration (WSI). Redundancy has been extensively used for manufacturing memory chips and to provide repair of these devices in the presence of faulty cells. If there are too many defects, the momory has to be rejected. But if there are a few defects, it will be more efficient and cost reducing for the company to use it by repairing. Therefore, laser-repair process is nedded for such a reason and redundancy analysis is needed to establish correct target of laser-repair process. The proposed CRA (Correlation Repair Algorithm) simulation, beyond the idea of the conventional redundancy analysis algorithm, aims at reducing the time spent in the process and strengthening cost competitiveness by performing redundancy analysis after simulating each case of defect.

Key words : RRAM, Redundant spare, Semiconductor manufacturing yield, Memory test, CRA

요 약

VLSI(Very Large Scale Integration)와 WSI(Wafer Scale Integration)와 같은 통합기술로 인해 큰 용량의 메모리 대량생산이 가능 하게 된 지금 Redundancy는 메모리 칩의 제조와 결함이 있는 셀을 지닌 디바이스를 치료하는데 광범위하게 사용되어져왔다. 메모리칩의 밀도가 증가함에 따라 결함의 빈도 또한 증가한다. 많은 결함이 있다면 어쩔 수 없겠지만 적은 결함이 발생한 경우에는 해당 다이를 reject 시키는 것 보다는 수선해서 사용하는 것이 메모리생산 업체 입장에서는 보다 효율적이고 원가절감 차원에서 필수적이다. 이와 같은 이유로 laser repair라는 공정이 필요하고 laser repair공정의 정확한 타깃을 설정하기 위해 redundancy analysis가 필요하게 되었다. CRA시뮬레이션은 기존의 redundancy analysis 알고리즘의 개념에서 벗어나 결함 유형별로 시뮬레이션한 후 RA를 진행함으로써 RA에 소요되는 시간을 절약함으로써 원가 경쟁력 강화를 할 수 있다.

주요어 : 수율, 메모리 테스트, 리던던트 분석, RRAM, laser repair

1. 서 론

반도체 메모리에 있어서의 지속적인 발전은 메모리 칩의 저장용량을 꾸준히 증가시켜왔다. 메모리 발전과정을 보면 대체로 메모리 칩의 용량은 18개월마다 두 배로 증가해 왔다. 이러한 회로의 소형화는 보다 정밀한 제조공정을 요구하고 이에 따른 제조비용 및 테스트 비용이 급격히 상승한다[1]. 이러한 VLSI 기술 혁신은 칩에 더 많은 저장 셀을 통합하면서 반도체 메모리의 고집적화와 대용량화를 이루었지만 불량 발생의 개연성도 상대적으로 증가되어왔다. 메모리의 용량이 급속도로 증가함에 따라 결함의 빈도 또한 높아지고 있으며 메모리 생산에 있어 중요한 요소 중 하나인 수율도 낮아지는 문제가 발생한다. 이 문제를 해결하고 수율을 높이기 위해 메모리를 수리하는 것이 중요한 역할을 차지하게 된다. DRAM 셀은 일종

2009년 8월 5일 접수, 2009년 11월 2일 채택
[1)] 성균관대학교 반도체 시스템 공학과
주 저 자 : 한영신
교신저자 : 한영신
E-mail; hanys@skku.edu

의 어레이 구조로 되어 있고 한꺼번에 여러 데이터를 In/Out 시키게 된다. DRAM의 경우 수많은 미세 셀 중 한 개라도 결함이 있으면 메모리로서 제구실을 하지 못하므로 불량품으로 처리된다. 하지만 DRAM의 집적도가 증가함에 따라 확률적으로 소량의 셀에만 결함이 발생할 확률이 높은데도 이를 불량품으로 폐기한다는 것은 수율을 낮추는 비효율적인 처리 방식이다. 따라서 이 경우 미리 DRAM내에 설치해둔 예비 메모리 셀을 이용하여 불량 셀을 대체시킴으로써 수율을 높이는 방식을 채용한다[2-6].

RA(Redundancy Analysis) 알고리즘은 불량 셀을 예비 메모리 셀로 대체하기 위해 필요한 정보를 얻는데 사용되는 알고리즘이다. 그러나 리던던시는 오버헤드 영역과 잠재된 수율의 손실 때문에 다른 형태의 비용을 추가한다. 그러므로, 최대의 수율과 최소의 비용을 위한 RA 알고리즘은 제조공정에서 매우 중요하다. 따라서 본 논문에서는 반도체 제조공정에서 사용되는 RA알고리즘에 대해 연구하였다. 웨이퍼상태의 테스트에서 RRAM (Redundant Random Access Memory) 을 치료하는데 많은 시도가 있었다. Repair Most는 메모리의 모든 행과 열의 결함 카운터를 사용하여 치료할 수 있는 행과 열의 위치정보를 얻고자 하였다[7]. Fault-driven comprehensive redundancy algorithm[8]은 모든 가능한 조합으로부터 최적의 치료 결과를 구현하기 위하여 사용자 정의의 선호도에 의존한다. Efficient spare allocation in reconfigurable arrays[9]에서는 RRAM의 치료를 위해서 새로운 2개의 알고리즘을 제시하였다. 첫 번째 알고리즘은 치료 공정에서 초기에 걸러내는 branch-and-bound 접근방식이고 두 번째 알고리즘은 heuristic 기준을 사용하는 것이다.

본 논문에서 제안한 시뮬레이터는 기존의 RA알고리즘 개념에서 벗어나 결함 유형별로 CRA(Correlation Repair Algorithm)를 진행함으로써 RA에 소요되는 시간을 절약함으로써 메모리의 생산비용을 줄일 수 있다는 것을 보여주는 것이 목적이다.

2. 관련 연구

EDS(Electronic Die Sort)공정에서 웨이퍼 테스트를 마치면 주된 셀의 어느 부분에서 결함이 발생하였는지를 알 수 있다. (즉, 결함의 발생 위치, 행과 열의 주소를 알 수 있음) 이 정보를 바탕으로 redundancy analysis는 다이가 가진 여유 셀을 할당하게 되는데 즉, 디바이스에 발생한 결함을 효율적으로 고치기 위해 디바이스가 가진 여유 셀을 할당하는 과정이 redundancy analysis이다.

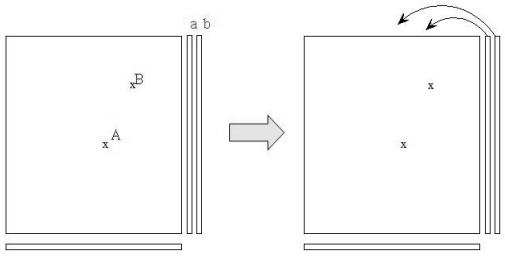

그림 1. 여유 셀로 치료하는 과정

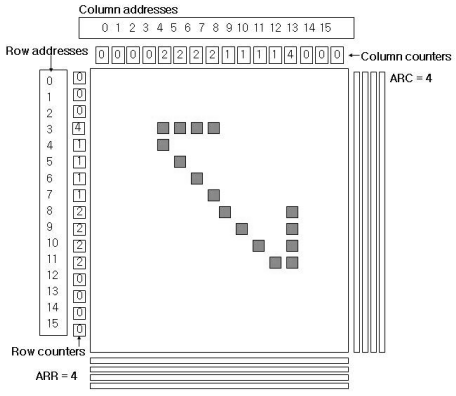

그림 2. Example of the occurrence of a defect

그림 1 행 쪽 여유 셀이 한 개, 열 쪽 여유 셀을 두 개 가진 디바이스의 경우에 결함(A,B)을 고치기 위해 여유 셀 (a,b)가 사용되고 있음을 보여 준다. 만약에 A, B 결함이 행 쪽으로 수평하게 발생하였다면 행 쪽 여유 셀 한 개로 치료하는 것이 열 쪽 여유 셀 두 개로 치료하는 것 보다 효율적이기 때문에 두개의 열 쪽 여유 셀로 치료하지 않고 행 쪽 여유 셀 한 개로 치료하였을 것이다. 참고로 결함이 한 개만 발생하였을 경우 행 쪽 여유 셀 한 개가 사용되거나 열 쪽 여유 셀 한 개가 사용되어도 마찬가지일 것이다. 이 경우 우선도 개념을 적용해서 대개 여유 셀이 많이 있는 행 또는 열 방향을 우선 사용하게 된다.

RRAM의 치료를 위해 많은 redundancy analysis 알고리즘이 있지만 가장 우선적으로 사용되어지는 것이 repair-most 알고리즘이다[4]. Repair-most 알고리즘은 먼저 각각의 행과 열의 카운터에 결함의 개수를 저장한다. 그림 2에서 행과 열의 카운터에 결함의 값이 저장되어 있음을 볼 수 있다. ARC (available-redundant-columns)와 ARR (available-redundant-rows)은 여유 셀의 개수를 표시하는 카운터이다. Repair-most 알고리즘은 그림 3에서처럼 가

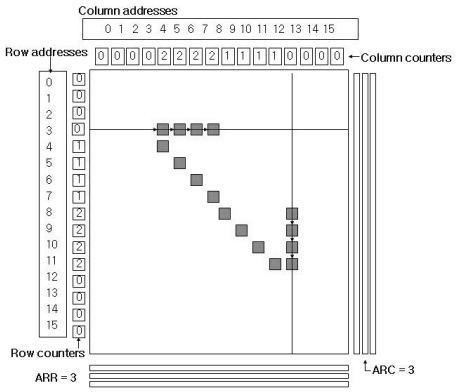

그림 3. Repairing a row a column with the highest value

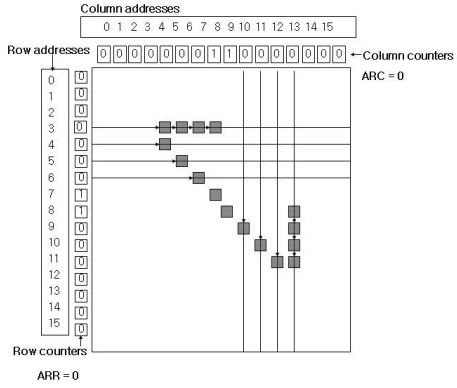

그림 4. Cells cannot be repaired even with the repair-most algorithm

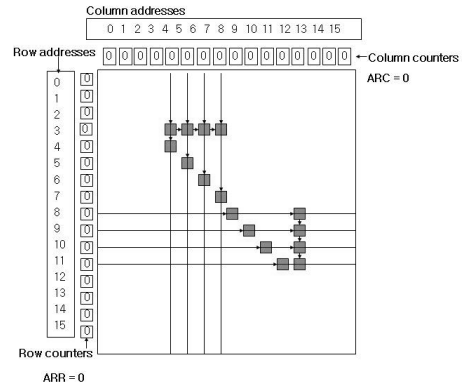

그림 5. Fault-driven algorithm

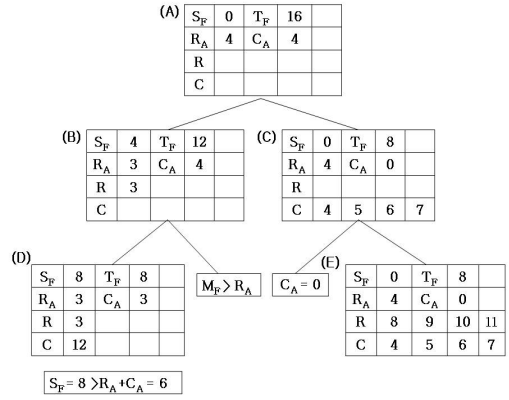

그림 6. Result of the FLCA algorithm

장 카운터 값이 높은 행과 열을 우선적으로 치료한다. 이 과정을 여유 셀을 다 사용하거나 결함이 모두 치료될 때까지 반복한다. 대개 repair-most알고리즘으로 일반적인 결함을 치료할 수 있으나 그림 4는 repair-most 알고리즘으로 결함을 모두 치료하지 못하는 경우를 보여준다. 이러한 경우 그림 5에서처럼 fault-driven 알고리즘을 사용하면 치료 할 수 있다.

Fault-Driven 알고리즘은 두 단계로 구성되어진다[5]. 첫 단계는 결함이 존재하는 방향과 같은 여유 셀로 대체되어야 하는 특정 행 또는 열을 결정하는 forced-repair 분석이다. 두 번째 단계는 forced-repair에 의해 사용되어지지 않은 여유 셀을 사용하여 forced-repair 단계이후 남아있는 결함에 대한 치료 방법을 결정하는 sparse-repair 분석이다.

Fault-Driven 알고리즘은 그림 5처럼 치료하기 위해 개의 레코드가 생성된다. FLCA(Fault Line Covering Approach) 알고리즘은 그림 6처럼 치료하는데 Fault-Driven 알고리즘의 레코드보다 적은 레코드만 생성될 뿐이다[6].

여기서 C_A는 ARC, R_A는 ARR, T_F는 fault RAM에 담겨 있는 전체 결함 수, S_F는 1개의 결함으로 이루어진 결함들의 수, M_F는 행과 열의 카운터에 담긴 결함 수 중에 가장 큰 값을 의미한다. FLCA는 forced repair analysis 후 남아있는 결함들과 fault RAM의 정보를 C_A, R_A, T_F, S_F로 분류한 후 큐에 저장한다.

FLCA는 단지 2번의 실행만으로 치료 해법을 얻을 수 있음을 그림 6에서 보여준다. 그림 6에서 부(parents) B와 C는 각각 오직 한 개의 자 (descendant)를 가지고 있음에 주목하여야 한다. ($M_F > R_A$이고 $C_A = 0$이므로) 그림 6에서 D는 $S_F > R_A + C_A$이므로 사용될 수 없다. 그러므로 치료 해법은 그림 6에서 parent E의 레코드로 주어진다.

3. 제안하는 CRA(Correlation Repair Algorithm)

3.1 상관 계수

패턴 X, Y의 상관 계수는 아래와 같이 표현된다.

COR(X,Y)라고 표시되는 random 변수 X와 Y의 correlation은 다음과 같다.

$$COR(X,Y)$$
$$=COV(Z_X,Z_Y) = \mathrm{E}\left[\left(\frac{X-\mu_X}{\sigma_X}\right)\left(\frac{Y-\mu_Y}{\sigma_Y}\right)\right]$$
$$=\frac{COV(X,Y)}{\sqrt{V(X)\,V(Y)}} = \frac{E(XY)-E(X)E(Y)}{\sqrt{V(X)\,V(Y)}}$$
$$\mu_X=\mathrm{E}(X),\ \sigma_X^2=V(X),\ Z_X=\frac{(X-\mu_X)}{\sigma_X},$$
$$Z_Y=\frac{(Y-\mu_Y)}{\sigma_Y}$$

COV(X,Y)라고 표시되는 random 변수 X와 Y의 covariance는 X와 Y의 의존적인 측정값이고 아래와 같이 표현된다.

$$COV(X,Y) = \mathrm{E}\left[(X-\mu_X)(Y-\mu_Y)\right] = \mathrm{E}(XY)-\mathrm{E}(X)\mathrm{E}(Y)$$

V(X)라고 표시되는 random 변수X의 분산은 평균에 대한 random 변수의 dispersion의 measure인데 아래와 같이 표현된다.

$$V(X) = \mathrm{E}[(X-\mu_X)^2] = \mathrm{E}(X^2)-\{\mathrm{E}(X)\}^2$$

3.2 대규모 DB에서의 속도 개선 방안

코릴레이션을 이용한 데이터 검색에서 속도 개선 방안은 입력패턴 X, 등록되어 있는 패턴에서의 Y1, Y2,· · · YN의 상관 계수는 다음과 같다.

$$COR(X,Y)$$
$$= \frac{\mathrm{E}\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sqrt{\mathrm{E}[(X-\mu_X)^2]\,\mathrm{E}[(Y-\mu_Y)^2]}}$$
$$= \frac{E(XY)-E(X)E(Y)}{\sqrt{[\mathrm{E}(X^2)-\{\mathrm{E}(X)\}^2][\mathrm{E}(Y^2)-\{\mathrm{E}(Y)\}^2]}}$$
$$= \frac{E(XY)-\mu_X\mu_Y}{\sqrt{[\mathrm{E}(X^2)-\mu_X{}^2][\mathrm{E}(Y^2)-\mu_Y{}^2]}}$$

계산 시간이 $\mathrm{E}[(X-\mu_X)^2]$ 보다 $\mathrm{E}(X^2)-\mu_X{}^2$ 이 빠르고 $Y-\mu_Y$는 float(4byte) 이지만 Y는 unsigned char(1byte) 라서 전처리 메모리 사이즈가 감소한다.

3.3 K-means Algorithm

패턴 유형 분류를 위한 이 알고리즘은 정확히 검색하는 것이 아니고 군집화를 이루는 작업이다. 식별해야할 전체 대상을 미리 군집화를 해두고 조금 더 빨리 검색하기 위한 수단이 된다. 따라서 최종적으로 코릴레이션 유사도를 측정하기 전에 샘플 패턴을 대상으로 군집화가 이루어져야하며 이것을 오프라인으로 학습을 한다. 데이터 이외에 클러스터의 수 K를 입력으로 하며 이때 K를 seed point 라고 한다. 하나의 샘플이 하나의 클러스터에 합류하자마자 곧 클러스터의 centroid 가 다시 계산되는 것이다. 따라서 K-means 알고리즘은 데이터집합에서 단지 두 번만의 패스가 이루어지며 그 과정은 다음과 같다.

- 1단계 : 처음에 K-클러스터로서 시작한다. 남아있는 n-k 샘플들에 대해서는 가장 가까이 있는 centroid를 찾는다. 이것에 가장 가까이 있는 centroid를 가지는 것이 확인된 클러스터에 샘플을 포함시킨다. 각각의 샘플들이 할당된 후에 할당된 클러스터의 centroid가 다시 계산된다.
- 2단계 : 그 데이터를 두 번 처리한다. 각 샘플에 대하여 가장 가까이 있는 centroid를 찾는다. 가장 가까이 있는 centroid를 가진 것으로 확인된 클러스터에 샘플을 위치시킨다. (이 단계에서는 어떤 centroid도 다시 계산하지 않는다.)

아래의 그림 7은 반복법을 통해 최적 해를 찾는 과정을 간략하게 보여준다.

(a) 입력벡터가 속하는 가장 가까운 클러스터를 찾음
(b) 할당된 클러스터의 centroid가 다시 계산
(c) 가장 가까이 있는 centroid를 가진 것으로 확인된 클러스터에 샘플을 위치시킴

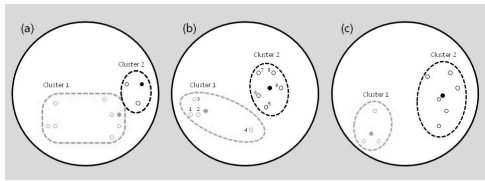

그림 7. K-means 알고리즘에서 최적 해를 찾는 과정

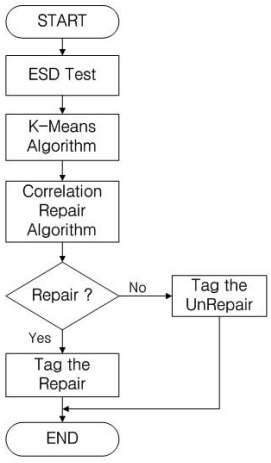

그림 8. Redundancy Analysis Simulation의 전체 Flow

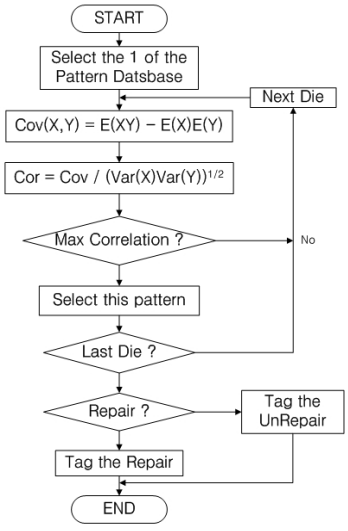

그림 9. CRA(Correlation Repair Algorithm) flow

본 연구에서는 샘플 패턴 DB를 10,000개로 하고 그 중에 대표 유형을 5, 10, 20가지로 랜덤하게 정하였다. 즉 K-Means 알고리즘 입력에 K = 5, 10, 20이 되고 5, 10, 20개의 클러스터가 출력이 된다. 입력 패턴 5, 10, 20개의 클러스터는 유사도 값이 5, 10, 20개가 나오고 그 중에 가장 유사한 클러스터를 입력 패턴의 유형으로 본다. 초기 값은 랜덤하게 하고 반복하면서 중심축이 이동한다. 수렴이 되면 대표 패턴이 생성된다. 클러스터링 하는 대상은 결함 패턴이 되고 초기 값도 그 중에 하나가 된다. 랜덤하게 선택된 5, 10, 20개의 결함패턴이 되는 것이다. 샘플 10,000개 중에서 10,000개를 자동으로 5, 10, 20개의 유

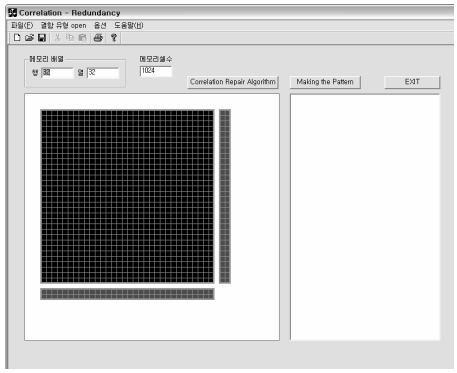

그림 10. CRA 인터페이스

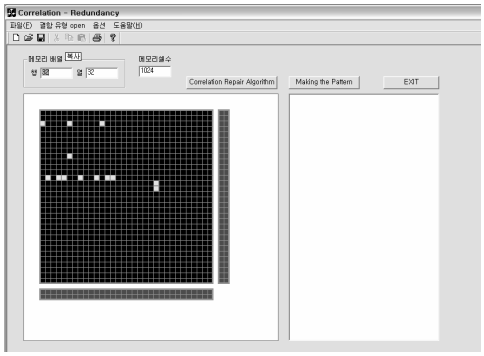

그림 11. 결함 패턴 유형의 로딩

형으로 분류하게 되는 것이다. 그림 8은 RA시뮬레이션의 전체 흐름도를 나타내고 그림 9는 CRA 흐름도를 나타낸다.

4. CRA(Correlation Repair Algorithm)

CRA시뮬레이션은 여유 셀로 치료할 수 있는 모든 경우의 수를 데이터베이스에 저장하고 비교 대상 디바이스의 테스트 후 분석된 결함 패턴의 결함 유형과 데이터베이스에 저장된 결함 유형을 비교하여 평균, covariance, 분산을 구한 후 correlation 값을 추출해 낸다. 시뮬레이션 과정은 다음 3단계로 이루어진다.

- 여유 셀로 치료할 수 있는 모든 경우의 수에 해당하는 정보를 생성 및 데이터베이스에 저장시킴.
- 테스트 후 얻어진 결함 패턴 정보를 가진 파일을 로딩
- 데이터베이스에 저장된 결함 유형과 결함 패턴 정보를 가진 파일의 결함 유형에 대한 CRA 진행

그림 10은 CRA 시뮬레이터 인터페이스를 나타낸다.

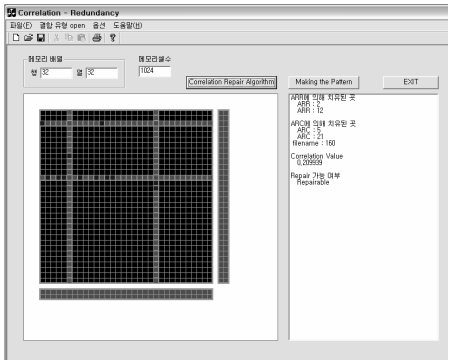

그림 12. CRA로 수리된 결과

데이터베이스에 저장된 결함 유형과 결함 패턴 유형을 비교하여 CRA를 진행한다. CRA는 수리가 가능하면서 correlation value를 추출한다. 그림 11와 그림 12에서 CRA을 통해 치료할 수 있으면서 유사도의 기준이 되는 최적의 correlation value를 산출한다. 이번 실험에서 디바이스 크기를 각각 다르게 가정하였기 때문에 결함 패턴 정보 파일은 행 방향으로 열 방향으로 이루어진 정사각형구조를 가진다.

5. 실험 및 고찰

5.1 실험의 목적

메모리의 용량이 급속도로 증가함에 따라 결함의 빈도 또한 높아지고 있으며 메모리 생산에 있어 중요한 요소 중 하나인 수율도 낮아지는 문제가 발생한다. 이 문제를 해결하고 수율을 높이기 위해 메모리를 수리하는 것이 중요한 역할을 차지하게 된다. 그리고 메모리의 수리는 메모리에서의 모든 고장을 검출하기 위한 테스트과정과 고장의 종류 및 위치를 결정하는 진단과정과 검출된 고장을 위해 여분을 배치하는 단계를 통해서 이루어 질 수 있다.

많은 결함이 있다면 어쩔 수 없겠지만 적은 결함이 발생한 경우에는 해당 다이를 사용안하는 것보다 수리해서 사용하는 것이 메모리 생산 업체 입장에서는 보다 효율적이고 원가 절감 차원에서 필수적이다. 이와 같은 이유로 Laser repair 라는 공정이 필요하고 Laser repair 공정의 정확한 타깃을 설정하기위해 RA가 필요하게 되었다. 이에 본 실험에서 사용한 CRA(Correlation Repair Algorithm)은 코럴레이션기법을 사용하여 수리하는 CRA알고리즘이다.

5.2 실험계획

지금까지 RA는 장비 개발 업체에서 제공하는 경우가 대부분 이었고 각 장비 업체별로 RA알고리즘을 개발하여 제공하여 왔기 때문에 동일한 결함 유형에 분석하는 RA시간이 각 장비 업체 별로 다른 경우가 대부분이었다. 이에 본 실험에서는 랜덤한 10,000개의 결함패턴을 K-means 알고리즘을 사용하여 5, 10, 20개의 군집유형으로 나누어 두고 CRA(Correlation Repair Algorithm)을 시뮬레이션 한다. 개발 환경으로는 Visual C++ / Window XP를 사용했다. CRA를 시뮬레이션하기위한 조건은 다음과 같다.

표 1. CRA(Correlation Repair Algorithm) 시뮬레이션 조건

Array Size	Number of Defective 셀s	Number of ARR	Number of ARC
32 × 32	13	2	2
64 × 64	13	4	4
128 × 128	28	4	4
256 ×256	30	5	5
512 × 512	45	10	10
1024 ×1024	60	20	20

- 샘플 패턴 : 10,000개의 메모리 어레이 사용
- 디바이스별 크기 (n × n) : 16 × 16, 32 × 32, 64 × 64, 128 × 128, 256 ×256, 512 × 512, 1024 × 1024 의 정사각형 어레이 사용.
- Line redundancy 개수(ARR=ARC) : 각각의 디바이스별로 정해 실험
- CRA(Correlation Repair Algorithm)의 입력요소 : 랜덤 결함 패턴

표 1은 CRA(Correlation Repair Algorithm)의 시뮬레이션 조건을 나타낸다.

5.3 실험 결과

EDS에서 테스트 시간을 비롯한 RA소요시간은 비용에 직결된다. 그러므로 본 실험결과는 각 디바이스별 크기와 패턴개수에 따라 10,000개의 결함 패턴을 랜덤하게 5, 10, 20개의 군집으로 나누어 CRA알고리즘으로 시뮬레이션 했을 때 수행시간을 실험하였다. 본 실험에서는 디바이스 사이즈가 커지면 데이터베이스 용량이 많아져 군집화 하는 방법을 썼다. 임의로 선택된 K개의 군집을 중심으로 시작하여 각 결함패턴은 가장 가까운 중심의 군집에 할당된다. 새로운 군집에 할당된 결함패턴들의 새로운 중심을 계산하며 결함패턴들이 다른 군집으로 재 할당 되지 않거나 SE (square-error) 값이 감소하지 않을 때까지 위의

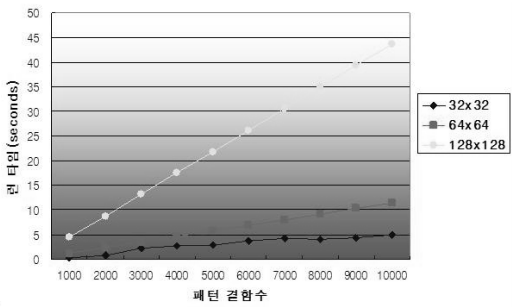

그림 13. 각 디바이스별 CRA 알고리즘 시뮬레이션

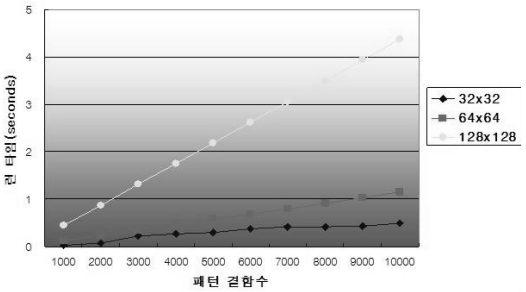

그림 14. 10개의 군집으로 나눈뒤 CRA알고리즘 시뮬레이션

과정 반복하여 선택과정에서 수렴에 많은 시간이 소요되는 것을 개선시킨다. 그림 13은 각 디바이스별 CRA 알고리즘 시뮬레이션 결과를 나타내고 그림 14는 10개의 군집으로 나눈 뒤 CRA알고리즘 시뮬레이션 결과를 나타낸다.

5.3.1 기존 알고리즘과 제안하는 CRA 성능평가

기존 RA 알고리즘의 수행속도를 보면 불량 셀의 개수에 의해 민감하게 영향을 받아 불량 셀의 수가 늘어나면 수행속도가 많이 걸리는 반면에 본 연구에서 제안하는 알고리즘은 단지 패턴의 수와 셀의 사이즈 크기에 따라 영향을 받기 때문에 CRA은 사이즈 크기를 i이라 했을때 O(i)의 시간 복잡도를 가지고, 또한 사이즈 크기를 j 이라 했을 때 O(j)의 시간 복잡도를 가지므로 제안된 CRA의 효율성을 확인 할 수 있었다. 또한 CRA은 모든 결함의 종류를 구별 하는 것이 가능하여 기존보다 효율적인 불량 분석 업무를 진행할 수 있으며 수율 및 품질 향상에 기초가 될 수 있을 것이다. 표 2는 기존 알고리즘과 제안하는 CRA 성능평가를 나타낸 것이다.

6. 결 론

반도체 제조 기술의 발달은 더욱 작고 복잡한 회로의 구현을 가능하게 하여, 단위 면적당 메모리 셀 수를 증가시켰다. 메모리의 집적도 또한 계속적으로 증가되고 있다. 그러나, 이러한 메모리 칩의 평면적 수직적 축소는 복잡하고 정밀한 제조 공정을 요구하게 되고, 이러한 공정을 통하여 완성된 메모리 제품의 신뢰도 및 품질 보장을 위하여 점점 더 복잡하고 정교하고 긴 시간을 요하는 테스트가 필요하게 되었다. 64M 이상의 메모리에서는 테스트가 전체의 생산비용의 40%이상을 차지하게 되어 반도체 메모리에 있어서 테스트 및 수리는 반도체의 가격 경쟁력을 결정하는 중요한 기술이 될 것 이다.

기존의 RA 알고리즘은 지수적인 복잡도를 갖거나 필요로 하는 기록인자가 많은 단점을 가지고 있다. 또한 최적의 RA결과를 산출하기위해 여러 번 결함 유형을 분석해야 최적의 해를 얻을 수 가 있었다. 하지만 CRA은 이미 결함 유형을 데이터베이스에 저장해 두고 군집화시켜 유사도가 높은 최적의 RA 결과를 산출하기 때문에 기존 RA 알고리즘의 보완 방법이 될 수 있다.

기존의 RA process는 최종적으로 치료할 수 없는 경우도 메인 셀의 결함 유형의 분석이 모두 끝난 후에 알 수 있었다. 그러나 CRA시뮬레이션은 이미 결함 유형의 분석이 끝나있기 때문에 코럴레이션 결과만 만족하면 바로 결함 유형 분석을 마칠 수 있다. 반도체 산업의 특성상 메모리의 용량은 갈수록 높아져 가고 상대적으로 용량당 단가는 낮아질 수밖에 없고 용량이 높아져 감에 따라 결함 유형도 더욱 다양해 질 수밖에 없기에 CRA 시뮬레이션이 RA소요시간 절감에 새로운 대안이 될 수 있다.

참 고 문 헌

1. Rei-Fu Huang, Jin-Fu Li, Jen-Chieh Yeh, and Cheng-Wen Wu, “A Simulator for Evaluating Redundancy Analysis Algorithms of Repairable Embedded Memories”, Proceedings of the Eighth IEEE International On-Line Testing Workshop (IOLTW’02)
2. C.-W. Wang, C.-F. Wu, J.-F. Li, C.-W. Wu, T. Teng, K. Chiu, and H.-P. Lin. A built-in self-test and selfdiagnosis scheme for embedded SRAM. In Proc. Ninth IEEE Asian Test Symp. (ATS), pages 45-50, Taipei, Dec. 2000.
3. C.-F. Wu, C.-T. Huang, C.-W. Wang, K.-L. Cheng, and C.-W. Wu. Error catch and analysis for semiconductor memories using March tests. In Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), pages 468–471, San Jose, Nov. 2000.
4. R. Rajsuman, “Design and test of large embedded memories”, an overview, IEEE Design & Test of Computers,

2001, pp. 16-27.
5. Chih-Tsun Huang, Chi-Feng Wu, Jin-Fu Li, Cheng-Wen Wu, "Built-In Redundancy Analysis for Memory Yield Improvement" IEEE TRANSACTIONS ON RELIABILITY, Vol. 52, No. 4, DECEMBER 2003.
6. Shyue-Kung Lu and Chih-Hsien Hsu, "Fault Tolerance Techniques for High Capacity RAM", IEEE TRANSACTIONS ON RELIABILITY, Vol. 55, No. 2, JUNE 2006
7. Kawagoe, T., et al.: 'A built-in self-repair analyzer (CRESTA) for embedded DRAMs'. Proc. 2000 Test Conf., ITC, Atlantic city, NJ, USA, 2000, pp. 567–573
8. Huang, C.-T., Wu, C.-F., Li, J.-F., and Wu, C.-W.: 'Built-in redundancy analysis for memory yield improvement', IEEE Trans. Reliab., 2003, 52, (4), pp. 386-399
9. Y. Zorian and S. Shoukourian, ''Embedded-Memory Test and Repair: Infrastructure IP for SoC Yield,'' IEEE Design & Test, Vol. 20, No. 3, May-June 2003, pp. 58-66.
10. R.-F. Huang et al., ''A Simulator for Evaluating Redundancy Analysis Algorithms of Repairable Embedded Memories,'' Proc. IEEE Int'l Workshop Memory Technology, Design and Testing (MTDT 02), IEEE CS Press, 2002, pp. 68-73.
11. C.-T. Huang et al., ''Built-in Redundancy Analysis for Memory Yield Improvement,'' IEEE Trans. Reliability, Vol. 52, No. 4, Dec. 2003, pp. 386-399.
12. C.-F. Wu et al., ''Fault Simulation and Test Algorithm Generation for Random Access Memories,'' IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 4, Apr. 2002, pp. 480-490.
13. T.J. Bergfeld, D. Niggemeyer, and E.M. Rudnick, ''Diagnostic Testing of Embedded Memories Using BIST,'' Proc. Design, Automation and Test in Europe Conf. (DATE 00), IEEE CS Press, 2000, pp. 305-309.

한 영 신 (hanys@skku.edu)
1997 이화여자대학교 전산정보학과(석사)
2004 성균관대학교 전기전자컴퓨터공학과(박사)
2004 이화여자대학교 컴퓨터그래픽스/가상현실 연구센터 박사후연구원
2005 성결대학교 멀티미디어학부 전임강사
2007 인하대학교 컴퓨터공학과 BK 계약교수
2008 아리조나대학교 ACIMS 센터 Visiting Scholar
현재 성균관대학교 반도체시스템공학과 연구교수

관심분야 : 모델링&시뮬레이션, 공장 자동화, 온톨로지, 데이터엔지니어링