논문 2010-47SD-12-4

불량 예비셀을 고려한 자체 내장 수리연산을 위한 분석 영역 가상화 방법

(An Analysis Region Virtualization Scheme for Built-in Redundancy Analysis Considering Faulty Spares)

정 우 식*, 강 우 헌*, 강 성 호**

(Woosik Jeong, Wooheon Kang, and Sungho Kang)

요 약

수율과 품질을 유지하기 위하여 불량 셀을 예비 셀로 수리하는 방법이 많이 사용되고 있다. 대부분의 메모리가 2차원 예비 셀 구조를 갖는 상황에서, 최근의 Giga 용량 메모리의 경우 대부분의 칩에서 예비 셀에도 불량이 존재 한다. 본 논문에서는 예비 셀에 불량이 있는 경우를 고려한 자체 내장 수리연산 시 기존의 모든 자체 내장 수리연산 회로에 적용이 가능한 분석 영역 가상화 방법을 제시하였다. 분석 영역 가상화 방법은 향후 메모리 고용량화에 따른 필수 해결 사항인 예비 셀 불량에 대한 효과적인 대처방안이 될 수 있을 것이다.

Abstract

In recent memories, repair is an unavoidable method to maintain its yield and quality. The probability of defect occurence on spare lines has been increased through the growth of the density of recent memories with 2 dimensional spare architecture. In this paper, a new analysis region virtualization scheme is proposed. the analysis region virtualization scheme can be applied with any BIRA (built-in redundancy analysis) algorithms without the loss of their repair rates. The analysis region virtualization scheme can be a viable solution for BIRA considering the faulty spare lines of the future high density memories.

Keywords: 메모리, 자체내장 수리 연산회로, 테스트, 분석 영역 가상화, 불량 예비 셀

I. 서 론

메모리 소자의 대용량화에 따라서, 수율과 품질을 동시에 만족시키기 위해 불량 셀 (faulty cell)을 정상적인 예비 셀 (redundant cell) 로 대체하여 수리 (repair)하는 방식이 많이 사용되고 있다. 대부분의 SoC (System on a Chip)의 내부에 사용되는 내장 메모리 (embedded memory)에 대해서는 외부의 비싼 테스트장치를 이용하여 테스트와 수리를 진행하는 것이 많은 비용이 요구되어짐에 따라, 자체 내장 테스트 회로 (Built-in Self Test)를 통한 불량 정보 취득과 자체 수리연산 회로 (Built-in Redundancy Analysis)를 이용한 수리 방법이 많이 사용되어 왔다. 또한 근 미래의 고용량 고속 동작 메모리를 구현하기 위해서 TSV (Through Silicon Via)를 이용하여 칩을 만드는 방안이 활발히 연구되어지고 있으며, 경우에 대해서는 기존의 칩을 packaging 할 때 필요한 bonding pad가 필요 없게 됨에 따라서 상용 메모리에 대해서도 자체 내장 테스트와 자체 수리연산 회로의 필요성이 대두되고 있다.

자체 수리연산 회로에 대한 많은 기존연구가 있었으나, 지금까지는 수리연산의 대상이 메인 셀 영역에 한정되었으며, 메인 셀 영역의 불량을 어떻게 예비 셀로 수리하는가 하는 알고리즘과 함께 적은 면적을 가지는

* 학생회원, ** 평생회원, 연세대학교 전기전자공학과
(Yonsie University)
접수일자: 2010년9월6일, 수정완료일: 2010년10월27일

회로 구현에 많이 치중되어 왔다[1~4]. 불량을 수리하기 위한 예비 셀은 메인 셀의 용량에 비하여 매우 적은 수를 가지게 된다. 메모리의 용량이 작은 경우나 보통의 SoC에서는, 메모리 칩 하나에 발생하는 불량의 수가 적으며, 그보다 적은 용량의 예비 셀에는 더욱 더 불량이 생기는 확률이 작기 때문에, 메모리 테스트 시 예비 셀 영역은 불량이 없다는 가정 하에 테스트조차 하지 않고도 수리를 하는 경우도 있었다. 하지만, 메모리의 용량이 커짐에 따라서 예비 셀에 발생하는 불량을 무시하지 못하게 되었으며, 이를 고려지 않고는 수율과 품질을 맞추기가 힘들어 졌다. 본 논문에서는 예비 셀에 불량이 있는 경우를 고려한 자체내장 수리연산방법을 제안하고자 한다.

Ⅱ. 2차원 예비셀 구조 메모리와 수리연산

그림 1은 2차원 예비 셀 구조를 갖는 8x8 메모리 블록에 각각 두 개의 예비 행 (spare row line)과 두 개의 예비 열 (spare column line)이 있는 경우의 예를 나타낸다. 예비 셀은 메인 셀의 불량을 대체하는 것이므로 동일한 형태의 셀로 구성되어야 한다. 예비 셀은 메인 셀과는 다른 address를 가지며, 정상동작 시와 구별되는 특별한 모드(mode)에서만 진입 가능하여 메모리 테스트와 수리 연산 시 접근(access)이 가능하다. 그림 1의 메모리 블록은 실제로는 10개의 행과 10개의 열로 이루어져 있지만, 이중에 8개의 행과 8개의 열은 메인 셀로 정의 되어 정상 동작 시에 access가 가능하다. 그림 1에서 8, 9번 행(열)은 예비 행(열)로 구별되어 사용된다. 2차원 예비 셀 구조에서는 하나의 불량에 대해서 행이나 열 단위로 수리가 가능 하도록 되어있다. 2차원 예비 셀 구조의 수리연산에서는 불량을 크게 단일셀 불량 (single fault), 라인 불량 (sparse fault), 확정 라인 불량 (must line fault)의 3가지로 구분할 수 있다[4]. 단일셀 불량은 해당 메모리 블록에 발생된 모든 불량에 대해서 행주소와 열주소 중 어느 것도 겹치는 다른 불량이 없는 경우를 말한다. 라인 불량은 두 개 이상의 불량이 동일 행 주소 또는 동일 열 주소를 갖는 경우를 말한다. 확정 라인 불량은 라인 불량 중에서 메모리 블록이 보유한 예비 행(열)의 개수보다 많은 불량 셀들이 동일한 행(열) 주소를 갖는 경우에 대해서 이를 수리가 능하기 위해서는 반드시 예비 열(행)으로 대체하여야만 하기 때문에 확정 라인 불량이라고 말한다. 그림 1에는 메인 셀 영역에 총 4개의 불량이 있으며, 두 개의 단일 불량과 하나의 라인 불량이 있다. 본 논문에서는 불량의 주소를 (행,열)로 표시하기로 하자. 단일 불량인 (3,1)과 (4,5)는 각각 하나의 예비 행이나 하나의 예비 열 중 어느 것으로도 수리가 가능하다. 라인 불량인 (2,4)와 (6,4)는 하나의 예비 열 4로 수리가 가능하며, 또한 두 개의 예비 행으로 수리하는 것도 가능하다. 만약 단일 불량인 (3,1)을 예비 행 8로 대체한다고 하면, 행주소가 3인 모든 셀 즉, (3,0)부터 (3,9)까지가 (8,0)에서 (8,9)로 대체된다. 일단 수리가 되면, 메모리의 불량이 있는 메인 셀에 해당되는 주소정보가 외부에서 들어올 때, 이 외부 주소에 해당하는 셀 대신에 대체된 예비 행이나 열 쪽의 셀이 대신 동작하도록 한다. 이러한 구조적인 이유로 인해서, 그림 1과 같이 불량 셀인 (3,1)을 예비 행 8로 대체할 경우에 예비 열 영역인 (3,8)과 (3,9)까지도 (8,8)과 (8,9)로 함께 대체가 된다. 또한 예비 행과 예비 열이 교차하는 부분을 교차 영역이라고 하면 그림 1의 예에서는 (8,8), (8,9), (9,8), (9,9)의 네 셀이 교차 영역에 해당한다. 그림 1의 예에서 메인 행 3을 예비 행 8로 대체하고, 메인 열 4를 예비 열 8로 대체하였다고 하자. 이 경우 일단 수리가 된 후에 메인 셀인 (3,4)에 정보를 저장하고자 한다면, 행 3과 열 4가 둘다 행 8과 열 8로 대체 되었으므로 결국 예비 행 8과 예비 열 8이 만나는 교차영역인 (8,8)에 정보가 저장된다. 즉 대체된 예비행과 예비 열이 교차하는 메인 셀은 사용된 예비 행과 예비 열의 교차영역의 셀로 대체가 되는 것이다. 이후 (4,5)의 불량 셀은 남아 있는 예비 행이나 예

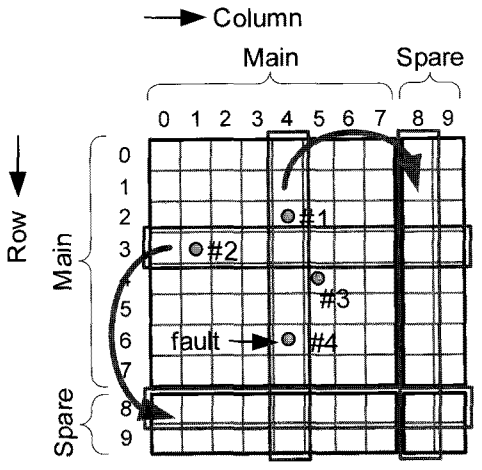


그림 1. 2차원 예비 셀 구조 메모리와 불량의 예
Fig. 1. Example of a memory block with 2-dimensional spare architecture and faults on it.

비 열 중 어느 것으로도 수리가 가능하다. 결과적으로 그림 1의 메모리 예는 수리가능하며, 모두 6가지의 수리연산해가 존재한다. 예비 행(열)을 R(C)로 표시하고 바로 뒤에 수리할 주소를 표시하면, 6가지의 수리 해는 각각 C4-R3-R4, C4-R3-C5, C4-C1-R4, R2-R3-C5-C4, R2-C1-R4-C4, R2-C1-C5-R6 이 된다.

III. 예비셀 불량과 수리연산

지금까지는 예비 셀에 불량이 없고 모든 예비 셀이 사용 가능하다는 전제하에 수리연산을 진행하였다. 예비 셀에 불량이 있는 경우의 수리연산에 있어서 가장 쉽게 생각할 수 있는 것은 불량이 있는 예비 행이나 예비 열을 사용하지 않도록 제외시키는 방법 (faulty spare exclusion) 을 통해 불량 예비 셀을 사용하지 않을 수 있다.

그림 2는 그림 1의 메모리 블록 예에 추가로 (8,2)에 불량이 발생한 경우이다. 그림 2에서는 예비 행 8에 불량이 있으므로 사용 가능한 예비 행의 수를 1개 줄인 상태에서 수리연산을 하는 것이다. 이제 사용 가능한 한 개의 예비 행과 두 개의 예비 열을 가지고 수리 해를 찾아보면, C4-C1-R4, C4-C5-R3 의 두 개의 해만이 존재함을 알 수 있다.

그림 3은 그림 1의 메모리 블록 예에 추가로 (8,8)에 불량이 발생한 경우이다. 그림 2와는 달리 예비 행과 예비 열의 교차영역에서 불량이 발생하였기 때문에 이러한 경우에서는 예비 행과 예비 열중에 어느 쪽을 제외시킬지를 결정하여야 한다. 예비 행 8을 제외시킨다면 그림 2와 같은 연산결과를 얻게 된다. 만약 예비 열 8을 제외시킨다면 수리 가능한 수리 해는 C4-R3-R4 단 하나기 존재하게 된다.

그림 3에서 예비 행과 예비 열의 어느 쪽을 제외시키는가에 따라서 수리가능해의 개수가 차이난다는 말은 수리효율의 차이로 이어지게 된다. 이는 불량이 존재하는 예비 라인을 무조건 제외시킨다는 제약사항에서 비롯된 것이다. 그림 3에서 (9,8)에 불량이 하나 더 발생하게 되는 경우에 예비 셀에 불량 발생 시 예비 행을 제외시키기로 했다면 예비 행8과 예비 행9를 모두 제외시키게 되어 수리 불가능이 되지만, 예비 열을 제외시킨다고 한다면 수리가능하게 된다. 또한 자체 내장 수리연산회로는 하드웨어로 구현해야 하는 상황에서 불량 상황에 따라 조건이 바뀌는 경우를 모두 고려하려면 이를 위한 회로가 많이 추가 되어야 하며 복잡도 또한 증가 되어 실제로 구현하기에는 어려운 점이 많다. 그림 4는 그림 1과 그림 2의 메모리 블록에 대하여 불량 예비 라인 제외 방법을 사용할 경우에 대한 이진검색 트리를 나타낸다. 기존의 자체내장 수리연산 알고리즘 중에서 CRESTA는 그림 4(a)와 같이 존재하는 모든 경우의 수를 하드웨어로 구현하여 불량 발생 때마다 비교하여 수리 가능해를 찾는 방법에서는 그림 2와 같이 예비 셀 영역에 불량이 발생하는 시점에서 바로 그림 4(b)와 같이 하드웨어 검색조건을 변경하여 비교하는 순서와

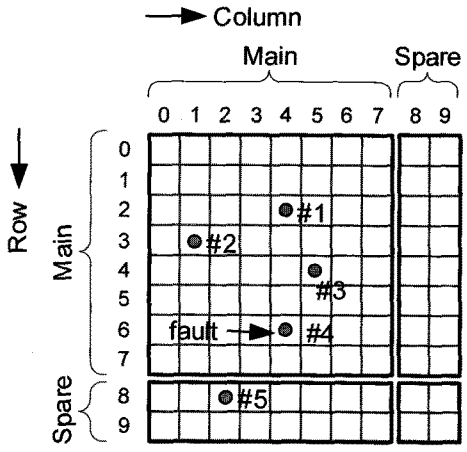


그림 2. 예비 행에 불량이 있는 메모리 블록의 예
Fig. 2. A memory example with a fault on a spare row line.

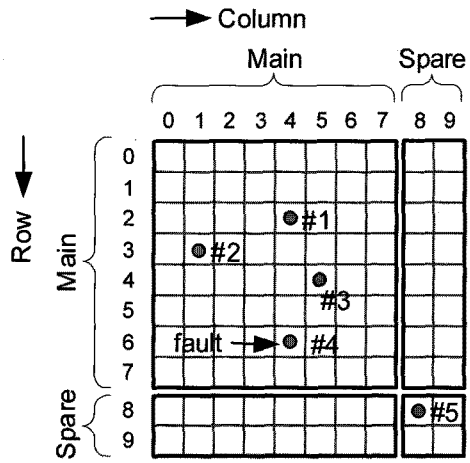
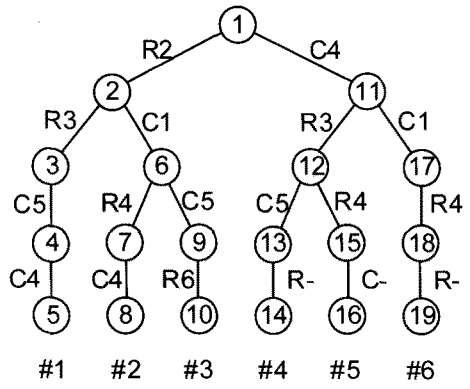
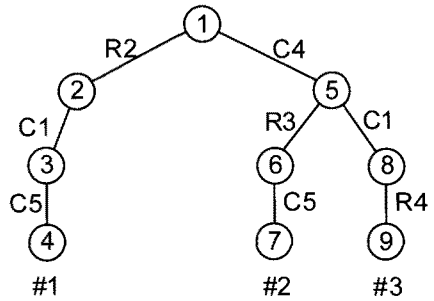


그림 3. 예비 행과 예비 열 교차영역에 불량이 있는 메모리 블록의 예
Fig. 3. A memory example with a fault on the cross point of a spare row line and a spare column line.



(a) Binary search tree for Fig . 1 with 2 spare rows and 2 spare columns



(b) Binary search tree for Fig . 1 with 1 spare row and 2 spare columns

그림 4. 예비 셀 영역의 불량에 의해 예비행의 수가 달라지는 경우에 그림 1의 메모리 블록에 대한 이진검색트리의 예

Fig. 4. Binary searcing tree for the memory example of Fig. 1 with different number of a spare row lines due to faults on spare area.

회로를 실시간으로 변경시키는 회로를 모든 예비 셀 불량을 가정하여 구현하는 것은 현실적으로 불가능하다.

IV. 분석 영역 가상화 방법

불량 예비라인을 연산에서 제외시키는 방법외에 해석영역을 예비 셀 영역까지 가상적으로 확장하고 모든 예비 셀들에 불량이 없다고 가정하여 연산을 하는 분석 영역 가상화 방법 (analysis region virtualization scheme)을 제시하고자 한다.

그림 5는 그림 2의 메모리 불량의 예에 대하여 분석 영역 가상화를 적용한 예이다. 원래 행 8과 9는 예비 행이었으나 분석 영역 가상화를 통해 메인 셀과 같이 취급한다. 마찬가지로 예비 열 8과 9에 대해서도 메인 셀과 같이 취급한다. 이와 함께 가상적인 예비 행과 예비 열이 추가적으로 있다고 생각하며, 이 예비 행(열)들은

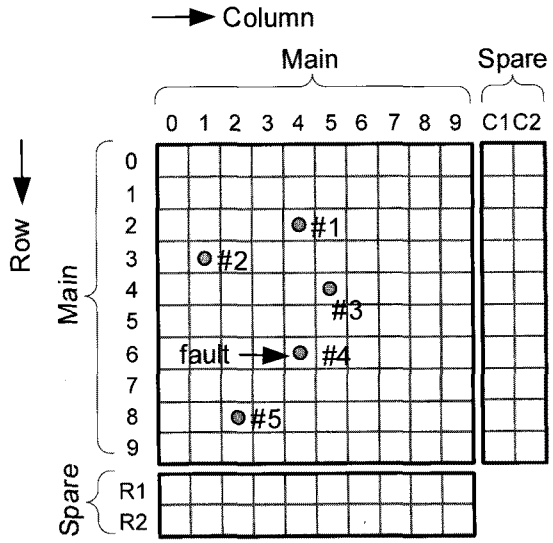


그림 5. 그림 2의 메모리 불량 예에 대한 분석 영역 가상화의 예

Fig. 5. An example of analysis region virtualization for a memory example of Fig. 2.

불량이 없이 모두 사용가능하다고 가정한다. 이렇게 분석 영역 가상화가 가능한 것은 이미 앞서 2장의 중간부에서 언급 했던 것처럼 하나의 예비 행(열)이 대체 될 때는 대상이 되는 메인 영역의 행(열)을 포함하여 이와 동일한 행을 가지는 예비 열(행)도 함께 대체되기 문이다. 다시 말하면 예비 행에 발생된 불량은 예비 열로 수리가 가능하고, 반대로 예비 열에 발생된 불량은 예비 행으로 수리가 가능하다는 것을 의미한다.

일단 그림 5와 같이 분석 영역의 가상화를 통해 수리연산을 수행하기 위해서는 기존의 자체내장 수리연산 회로에 다음의 사항들이 추가되어야 한다. 첫 번째는 불량발생 주소 저장을 위한 공간에 예비 셀 영역의 불량 주소도 저장이 가능하도록 한 비트(bit)씩 확장 하는 것이 필요하다. 그림 1에서는 메인 영역의 주소가 행과 열 모두 0부터 7까지 있으며 이를 표시하기 위해서 열 주소와 행주소가 각각 세 비트씩 필요하게 된다. 하지만, 분석 영역 가상화를 통해 예비 셀 불량도 표시하기 위해서는 행과 열 주소 모두 네 비트씩이 필요하게 되어 한 비트씩이 기존보다 더 추가된다. 두 번째는 수리연산해의 최종 출력 시에 수리해의 주소가 예비 셀 영역에 해당하는 주소인 경우 이를 무시하도록 해야 한다. 예비 셀에 해당하는 주소는 정상 동작 시 access를 하지 않으므로 굳이 대체 하지 않아도 되기 때문이다. 이를 위해서는 최종 출력 단에 주소의 최상위 비트가 1

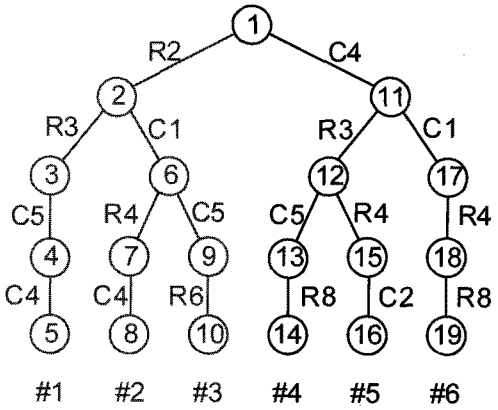


그림 6. 그림 2의 메모리 예에 대한 분석 영역 가상화 후의 이진검색트리

Fig. 6. A binary searching tree for a memory block example of Fig. 2 after analysis region virtualization.

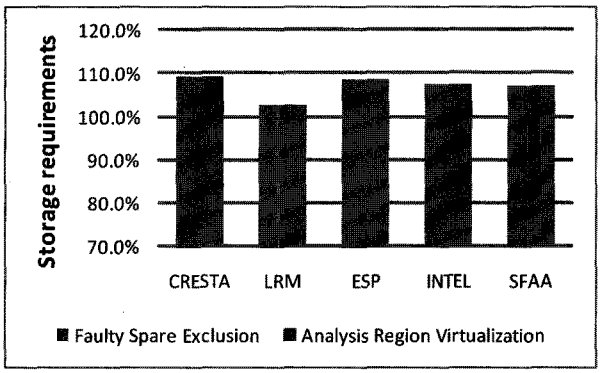


그림 7. 대표적인 자체내장수리 연산 회로들에 대한 불량 예비 라인 제거법과 분석 영역 가상화에 따른 추가 정보 저장공간의 비교 (1024x1024, 예비 행=5, 예비 열 =5)

Fig. 7. Comparison of additional storage requirements by applying faulty spare exclusion and analysis region virtualization for various BIRAs (1024 by 1024 memory with 5 spare rows and 5 spare columns).

인지 아닌지를 판단하는 AND gate하나만 추가하면 되므로 이에 따른 면적부담은 없다고 봐도 문제없는 수준이 된다. 결국 불량이 있는 예비 라인의 수를 줄이는 방법에 비해 분석 영역 가상화 방법이 면적 부담도 훨씬 적고 수리효율도 최적화 된 방법이라고 할 수 있다. 그림 6은 그림 2의 메모리 예에 대한 분석 영역 가상화 후의 이진검색트리를 나타낸 것이다. 결국 그림 5의 메모리 예에 대한 이진검색트리이며, 이 경우 처음의 세 개의 가지 (branch)는 수리 불가능이며 나머지 세 개는 수리 가능한 해가 된다. 세 개의 수리가능 해에 대해서 주소가 예비 셀 영역에 해당하는 수리해 만 제외하면 최종 수리해는 C4-R3-C5, C4-R3-R4-C2, C4-C1-R4와 같다. 결국 그림 2의 메모리 예에서 예비 행 8에 발생한 불량에 대해서 예비 행을 제외시킬 경우의 수리해와 예비 열을 제외시킬 경우의 수리해를 모두 포함하면서 수리가능한 모든 해를 구할 수 있게 된다.

V. 실 험

자체내장 수리연산회로의 예비 셀 영역에 불량이 발생하는 경우를 고려하여 불량 예비 라인 제외 방법과 분석 영역 가상화 방법을 제시하였다. 두 가지 방법의 장단점을 필요 정보저장 비트수와 수리 효율로 비교하였다.

자체내장 수리연산회로의 면적에 있어서, 정보저장에 필요한 비트수 만큼의 CAM (Content Addressible Memory)을 필요로 하며 이것이 면적의 대부분을 차지하게 되므로 많은 논문에서 면적비교를 이것으로 대신하여 왔다[2~5]. 그림 7은 1024행과 1024열을 가지며 5개의 예비 행과 5개의 예비 열로 구성된 메모리에 대하여 기존에 발표된 대표적인 자체내장 수리연산회로들의 정보저장 필요 비트수를 비교한 것이다. CRESTA[1], LRM[2], ESP[2], INTEL (IntelligentSolveFirst)[3], SFAA[4]의 원래 정보저장 필요 비트수를 각각 100%로 놓았을 때, 불량 예비라인 제거법은 필요 비트수의 증가가 없으므로 그대로 100% 이지만, 분석 영역 가상화를 위해서는 불량 주소를 저장할 때 열 주소와 행 주소 모두 한 비트씩이 추가 되어야 하므로 비교한 5개 수리연산 회로들에 대해 평균 약 7%정도의 면적이 추가로 필요하게 된다. 각 수리연산 회로들에 대한 면적 증가분이 서로 다른 것은 필요 비트수에 대한 주소 저장 비트가 차지하는 비율이 서로 다르기 때문이다. 하지만, 불량 예비 라인 제외법은 CRESTA와 같은 알고리즘에 대해서는 거의 적용이 불가능하며, 다른 알고리즘들에 대해서도 구현이 상당히 복잡하며, 이를 구현하기 위해서는 추가적인 회로가 많이 추가 되어야 하므로 반드시 불량 예비 라인 제외 방법이 분석 영역 가상화 방법보다 면적이 작다고는 보기는 힘들다..

그림 8은 불량 예비 셀을 고려한 알고리즘에 대하여 수리 효율을 비교한 것이다. 불량의 개수가 늘어남에 따라서 불량 예비라인 제거법의 수리 효율이 심각하게 떨어짐을 알 수 있는 반면, 분석 영역 가상화 방법은 수리

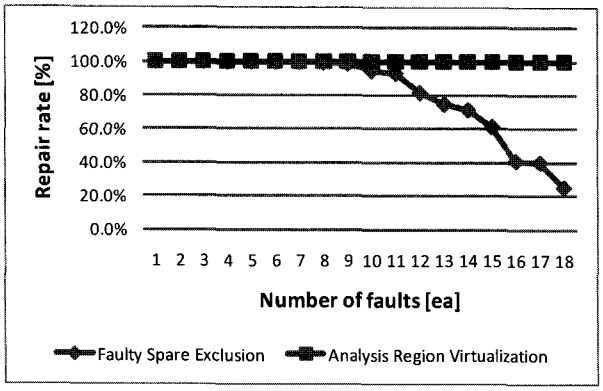


그림 8. 불량의 개수 증가에 따른 수리효율의 비교 (1024x1024, 예비행=5, 예비열=5)

Fig. 8. Comparison of repair rates with different number of faults for 1024 by 1024 memory with 5 spare rows and 5 spare columns.

효율의 저하 없이 완벽하게 수리해를 구할 수 있음을 보여준다.

실험 결과를 통해 분석 영역 가상화 방법이 약 7%정도의 추가저장 비트를 요구 하지만, 기존의 모든 자체 내장 수리 연산 회로에 쉽게 적용이 가능하며 동일한 수리효율을 가지는 것을 알 수 있다.

VI. 결 론

예비 셀 영역에 불량이 발생하였을 때를 고려하여 수리효율의 저하 없이 기존의 모든 자체 내장 수리연산 회로에 적용이 가능한 분석 영역 가상화 방법을 제안하였다. 이를 통해 향후 메모리 고 용량화에 따라 더욱 문제가 되는 예비 셀 영역 불량 발생에 대한 효율적인 대처 방안이 될 수 있을 것이다.

참 고 문 헌

[1] T. Kawagoe, J. Ohtani, M. Niiro, T. Ooishi, M. Hamada, and H. Hidaka, "A built-in self repair analyzer (CRESTA) for embedded DRAMs" in *Proc. International Test Conference*, pp.567-574, Oct. 2000.

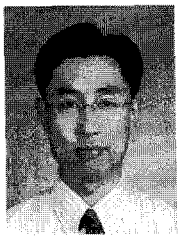
[2] C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, "Built-in Redundancy Analysis for Memory Yield Improvement," *IEEE Trans. Reliability*, vol. 52, pp.386-399, Dec.2003.

[3] P. Öhler, S. Hellebrand, and H.-J. Wunderlich, "An Integrated Built-in Test and Repair Approach for Memories with 2D Redundancy" in *Proc. European Test Symposium (ETS)*, pp.91-96, May 2007.

[4] W. Jeong, I. Kang, K. Jin, and S. Kang, "A Fast Built-in Redundancy Analysis for Memories With Optimal Repair Rate Using a Line-Based Search Tree" in *IEEE Trans. Very Large Scale Integration (VLSI) systems*, vol.17, pp. 1665-1678, Dec. 2009

[5] K. Pagiamtzis, and A. Sheikholeslami, "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no.3, pp.712-727, Mar. 2006.

저 자 소 개



정 우 식(학생회원)
1997년 고려대학교 제어계측 공학과 학사 졸업.
1999년 고려대학교 전자공학과 석사 졸업.
1999년~현재 하이닉스반도체 근무 중.
2010년 현재 연세대학교 전기전자공학과 박사과정.
<주관심분야 : 반도체, SoC 설계, 테스트>

강 우 헌(학생회원)
2009년 연세대학교 전기전자 공학과 학사 졸업.
2010년 현재 연세대학교 전기전자 공학과 석사과정.
<주관심분야 : 반도체, SoC 설계, SoC 테스트>

강 성 호(평생회원)
1986년 서울대학교 제어계측 공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터 공학과 석사 졸업.
1992년 The University of Texas, Austin 전기 및 컴퓨터 공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원.
1994년 Motorola Inc. 선임 연구원.
2010년 현재 연세대학교 전기전자공학과 교수.
<주관심분야 : SoC 설계, SoC 테스트>