논문 2012-49-9-26

비트맵 메모리 공유를 통해 면적을 크게 줄인 효율적인 수리 방법

(An Efficient Repair Method to Reduce Area Overhead by Sharing Bitmap Memory)

조 형 준*, 강 성 호**

(Hyungjun Cho and Sungho Kang)

요 약

최근의 시스템 온 칩 (SoC) 설계 기술의 발전에 따라, 수백개의 임베디드 메모리 코어들이 칩의 대부분의 면적을 차지하고 있다. 그러므로 시스템 온 칩의 수율은 임베디드 메모리 코어들의 수율에 따라 결정된다고 볼 수 있다. 최적의 수리 효율을 가지는 built-in self repair (BISR)을 모든 메모리들이 가지고 있게 된다면 면적의 부담이 너무 크다. 본 논문에서는 이와 같은 면적의 부담을 줄이기 위하여 메모리들을 그룹화 한 후에 비트맵 메모리를 공유하여 면적 부담을 크게 줄이는 방법을 제안한다. 제안하는 비트맵 메모리 공유방법은 built-in redundancy analysis (BIRA)의 면적을 크게 줄일 수 있다. 실험결과를 통해서 보면 제안하는 방법이 면적 부담을 대략 80%정도 줄이는 것을 확인 할 수 있다.

Abstract

In recent system-on-chip (SoC) designs, several hundred embedded memory cores have occupied the largest portion of the chip area. Therefore, the yield of SoCs is strongly dependent on the yield of the embedded memory cores. If all memories had built-in self repair (BISR) with optimal repair rates, the area overhead would be very large. A bit-map sharing method using a memory grouping is proposed to reduce the area overhead. Since the bit-map memory occupies the largest portion of the area of the built-in redundancy analysis (BIRA), the proposed bit-map sharing method can greatly reduce the area overhead of the BIRA. Based on the experimental results, the proposed method can reduce the area overhead by about 80%.

Keywords : Memory grouping, Built-in self repair (BISR), Built-in Redundancy Analysis (BIRA), Bitmap memory sharing

Ⅰ. 서 론

최근의 시스템 온 칩 디자인은 매우 많은 수의 임베디드 메모리 코어들을 포함하고 있고, 이러한 임베디드 메모리들은 칩 면적의 대부분을 차지하고 있다[1~2]. 이러한 메모리들의 용량과 집적도가 증가함에 따라, 고장의 수와 종류 또한 증가하게 되었다. 이러한 고장들은

* 정회원, ** 정회원-평생회원, 연세대학교 전기전자공학과
(Department of Electrical Electronic Engineering, Yonsei University)
※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2010-0024707).
접수일자: 2012년6월12일, 수정완료일: 2012년8월16일

메모리의 수율과 질을 크게 저하시키기 때문에 자가 수리를 할 수 있는 built-in self-repair (BISR)이 메모리를 테스트하고 수리하기 위해 설계되어 왔다[3~6]. 그러나 만약 이러한 BISR이 모든 메모리들에 포함이 된다면 그에 따른 면적 부담은 매우 커지게 된다.

이와 같은 면적 부담을 줄이기 위하여 셀프 테스트를 할 수 있는 기법인 built-in self test (BIST)를 공유하기 위한 메모리 그룹화 방법이 제안되었다[1]. 이러한 메모리 그룹화 방법은 직렬연결 방법과 병열연결 방법의 두 가지의 메모리 연결 방식으로 나뉘게 되는데, 이것은 회로의 크기와 파워 소비량에 따라 결정되게 된다. 그러나 이러한 방법은 BIST만을 공유하기 때문에 자가 수리에 대한 내용은 포함되어 있질 않으므로 자가 수리를 할 수 있는 설계 방법인 built-in redundancy analysis (BIRA) 방식은 따로 설계되고 고려되어야 한다.

시스템 온 칩의 메모리들을 동시에 수리하기 위한 방법 또한 개발이 되었다[7]. 이 방법은 칩의 공간을 줄이기 위하여 테스트와 수리를 할 수 있는 BISR 회로를 시스템 온 칩의 메모리를 평행하게 테스트하고 수리 할 수 있도록 하는 방법이다. 각각의 래퍼는 간단한 수리 알고리즘을 포함하고, 이러한 수리 방법과 분석을 진행할 수 있도록 제어하는 제어 회로를 가지고 있어서 평행하게 수리 방법을 찾을 수 있도록 하는 방식이다. 하지만 이 방법은 최적의 해결 방법을 찾을 수 없을 뿐만 아니라 BISR을 공유하지 않고 모든 메모리를 위하여 하나씩의 BISR을 포함하고 있기 때문에, 면적에 대한 부담을 줄일 수 없는 단점이 있다.

본 논문에서 제안하는 방법은 우선적으로 면적 부담을 줄이고 파워 소비를 줄일 수 있도록 메모리들을 그룹화 한다. 이 메모리 그룹화 방법은 앞서 언급한 것처럼 직렬 혹은 병렬연결 방법으로 나뉘게 되는데, 병렬연결 방법은 워드의 크기가 똑같은 메모리들을 그룹화 할 때 이용되게 되고, 직렬연결 방법은 같은 비트 크기를 가지는 메모리들을 그룹화 할 때 이용하게 된다. 병렬연결 방법의 장점은 연결된 메모리들을 동시에 테스트 할 수 있다는 장점이 있지만 그에 따라 파워 소비량이 매우 커지게 때문에, 직렬연결 방법과 병렬연결 방법을 동시에 이용하는 것이 면적 부담과 파워 소비를 줄이는데 가장 적합하다고 이야기 할 수 있다[1]. 또한 본 논문에서는 BIRA에 대한 면적 부담을 줄이기 위하여 BIRA의 면적에 가장 큰 부분을 차지하는 비트맵 메모리를 공유하는 방법을 제안한다. 이 방법은 많은 수의 메모리를 가지고 있는 칩들의 면적 부담을 크게 줄일 수 있고, 뿐만 아니라 최적의 해결책을 찾을 수 있는 BIRA 알고리즘[6]을 이용하기 때문에 최적의 수리 효율을 가질 수 있게 해준다.

Ⅱ. 본 론

1. 제안하는 BISR 공유 방법

메모리를 수리하기 위한 BIRA는 반드시 고장난 셀들의 주소를 저장할 수 있는 공간인 비트맵을 가지고 있어야 하는데, BIRA의 가장 큰 면적을 차지하는 것이 바로 이 비트맵 메모리이다. 제안하는 방법은 이러한 비트맵 메모리를 공유하여 하나의 비트맵이나 혹은 매우 적은 수의 비트맵으로 많은 수의 메모리들을 수리할 수 있도록 하는 방식이다. 이와 같이 비트맵을 공유하여 사용하기 때문에 제안하는 방식은 BIRA의 면적을 매우 크게 줄일 수 있게 되는 것이다. 또한 제안하는 방식은 이러한 하나의 혹은 적은 수의 비트맵뿐만 아니라 고장이 없다고 판명이 된 메모리나 비트맵을 이용하여

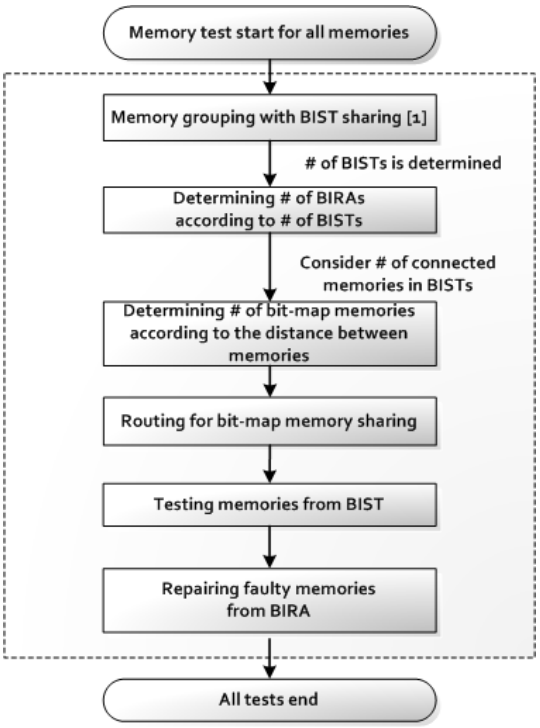

그림 1. 제안하는 방법의 순서도
Fig. 1. Flow of the proposed method.

미리 수리가 된 메모리들을 비트맵처럼 이용하여 BIRA의 면적을 더 크게 줄일 수 있도록 한다.

그림 1은 제안하는 방법의 순서도를 보여주는 그림이다. 제안하는 방법은 가장 먼저 BIST의 면적을 줄이기 위하여 BIST를 공유하는 방식인 [1]과 유사한 방법으로 메모리 그룹화를 수행한다. 그 다음에, 자가 수리를 할 수 있는 BIRA는 BIST가 메모리 테스트를 수행한 결과물을 이용하여 자가 수리 방법을 찾는 회로이기 때문에, BIST의 수에 맞게 BIRA의 수를 정하게 된다. 뿐만 아니라 일정한 거리 안에 있는 메모리들만이 비트맵의 공유가 가능하기 때문에, 메모리들끼리의 거리에 따라서 비트맵의 수를 결정한다. 이렇게 비트맵의 수를 결정한 후에는 메모리들의 거리에 따라 비트맵을 공유할 수 있도록 라우팅을 수행하게 된다. 이러한 라우팅 방법은 후에 자세히 설명하도록 하겠다. 이렇게 라우팅을 수행한 이후에는 BIST에 의해 연결된 메모리들을 테스트하고 BIRA에서 이러한 정보를 토대로 수리를 위한 솔루션을 찾아서 고장난 메모리들을 수리 할 수 있게 된다.

앞서 이야기한 바와 같이, BIST는 자가 테스트를 통하여 고장에 대한 정보를 BIRA의 비트맵에 저장하여

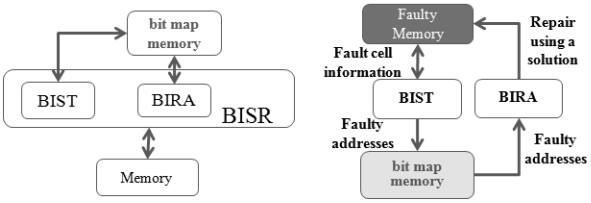

그림 2. 제안하는 방법의 BISR 구조
Fig. 2. Proposed BISR structure.

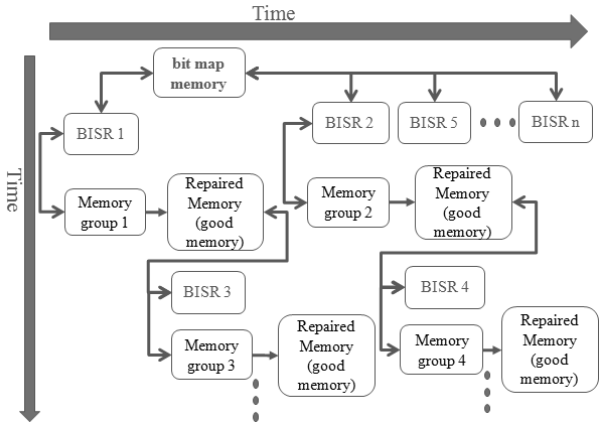

그림 3. 비트맵을 공유 하는 방법
Fig. 3. Bit-map memory sharing method.

이렇게 저장된 고장주소들을 이용하여 최적의 수리방법을 찾게 된다. 제안하는 방법은 그림 2에서 볼 수 있는 것과 같이, BIST의 테스트 결과물인 고장에 대한 정보를 담기 위하여 비트맵을 BIRA에서 분리하여 공유를 하게 된다. 만약 비트맵을 공유하지 않게 된다면 모든 메모리들은 크기가 큰 비트맵을 모두 가지고 있어야 하기 때문에, 자가 수리를 위한 면적 부담이 너무 크게 되므로 이와 같이 비트맵을 따로 생각하여 공유를 하게 되어 면적을 크게 줄일 수 있게 된다.

그림 3과 같이, 고장이 발생한 메모리들을 수리하기 위하여 제안하는 방법은 하나의 비트맵 메모리나 혹은 고장이 없다고 판명 난 메모리 혹은 이미 수리가 끝난 메모리들을 비트맵으로 이용하게 된다. 그림에서 볼 수 있는 것과 같이 우선 BISR1을 이용하여 메모리1을 테스트 및 수리를 하고, 이렇게 수리가 된 메모리1과 비트맵을 이용하여 다른 두 개의 메모리를 테스트하고 수리하는 방식이다.

그림 4는 비트맵을 공유하는 방법에 대한 하나의 예제이다. 그림의 M1부터 M7은 각각 테스트 및 수리를 하기 위한 메모리들을 의미한다. 그림의 그룹들은 [1]의 방식과 동일한 방법으로 메모리 공유 방식을 통하여 메모리 그룹이 4개로 분리된 것을 의미한다. 이와 같이 분류된 4개의 메모리 그룹들을 위하여 하나의 비트맵이 필요하고 이러한 하나의 비트맵 메모리를 통하여 7개의 메모리를 순차적으로 모두 테스트하고 수리를 할 수 있게 된다. 가장 먼저 M1이 BIST를 통하여 전달된 고장난 셀의 주소를 비트맵에 저장하고 이를 이용하여 수리를 한다. 이와같이 수리가 끝난 M1 메모리와 하나의 비트맵 메모리를 이용하여 이번에는 M2의 고장난 셀의

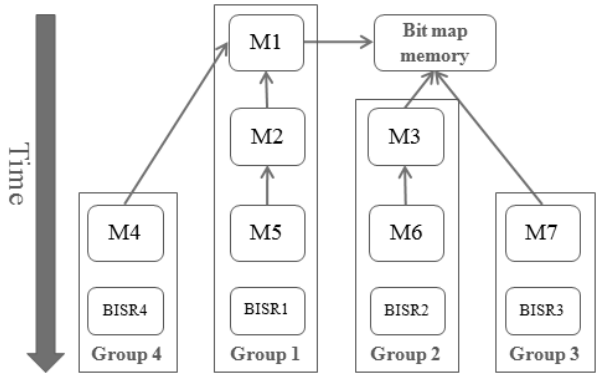

그림 4. 비트맵 메모리를 공유하는 방법의 예
Fig. 4. Example of sharing bit-map memory.

정보를 BIST를 통하여 M1에 저장하고, M3의 고장난 셀의 정보는 비트맵 메모리에 저장을 한다. 이와 같이 저장된 각 메모리의 고장난 셀의 주소에 대한 정보를 각각 이용하여 또다시 고장 난 셀들을 여분의 셀들을 이용하여 수리한 후에, 다시 M4부터 M7의 고장 정보들을 앞서 수리가 끝난 메모리인 M1부터 M3와 하나의 비트맵에 각각 저장하고 이를 이용하여 다시 수리를 진행한다. 이 예제에서 보는 바와 같이 제안하는 방법은 오로지 하나의 비트맵을 이용하여 모든 메모리를 테스트 하고 수리 할 수 있기 때문에 고장을 수리하기 위한 BIRA의 면적을 크게 줄일 수 있게 된다.

사실상 병렬연결 방법은 동시에 모든 메모리를 테스트하고 수리를 하여야 하기 때문에, 모든 메모리가 개별적인 비트맵 메모리를 가지고 있어야 한다. 이러한 이유 때문에, 제안하는 방법은 병렬연결보다는 직렬연결에 최적화되어 이용이 가능하다. 하지만 앞서 이야기한 바와 같이 병렬연결만을 이용하여 메모리를 그룹화 한다면 면적의 부담이 너무 커지고 동시에 소비하는 파워 소비량 또한 크게 증가하기 때문에, 직렬연결과 병렬연결을 동시에 수행하여야 하며, 이렇게 동시에 이용하게 된다면 제안하는 방법은 하나 혹은 적은 수의 비트맵을 이용하여 테스트 및 수리가 가능하게 된다. 하지만 이렇게 직렬연결과 병렬연결을 동시에 이용하기 때문에 비트맵의 수를 최소화하기 위하여 무조건적으로 가장 적은 수의 비트맵을 이용하게 되면, 때에 따라서는 추가적인 시간이 소요가 되게 된다. 그러므로 제안하는 방법은 이러한 추가 시간의 소요를 막기 위해서 하나의 BIST에 직렬 연결된 메모리들의 수에 따라서 비트맵의 수를 미리 정하여, 추가 시간의 소요를 막는 방식으로 진행을 하게 된다. 예를 들어, 만약 하나의 BIST에 직렬 연결된 메모리의 수가 가장 많은 BIST의 수가 두 개라면 비트맵의 수 또한 두 개로 늘려서 한꺼번에 수리를 진행할 수 있도록 하여 추가로 발생하는 시간을 사전에 예방하는 것이다.

이와 같이 비트맵을 공유하기 위해서는 비트맵을 공유하여 테스트할 메모리의 거리와 주파수 또한 고려해야한다. 수리할 메모리와 비트맵의 거리가 너무 멀다면 서로 공유할 수 없기 때문에 특정 거리안의 메모리만을 공유하게 되고, 또한 주파수가 다른 메모리는 공유를 할 수 없기 때문에 같은 주파수를 가지는 메모리만을 대상으로 비트맵을 공유하게 된다.

2. 거리에 따른 비트맵 공유 방법

[1]의 BIST를 공유하는 방법과는 달리 제안하는 방법의 비트맵 공유방식은 같은 고장 그룹의 메모리만을 이용할 필요가 없이, 다른 그룹의 메모리도 고장정보를 저장하기 위해서 이용하게 된다. 하지만 이렇게 공유를 하기 위해서는 반드시 고장정보를 저장할 메모리와 고장 수리를 진행할 메모리와의 거리가 특정거리 이하에 있어야만 한다. 때문에 그림 4의 방법과 같이 무조건적으로 그룹안의 메모리를 이용하지 않고, 다른 그룹의 메모리를 때때로 이용할 수 있도록 라우팅을 해주어야만 한다. 이처럼 특정 거리 안에 있도록 하게 하는 거리는 설계 환경에 따라 다르게 설정이 될 것이다.

예를 들어 테스트하고 수리해야할 메모리 그룹이 V = {m1, m2, m3, ..., mn}라고 한다면, 메모리와 고장 정보를 저장할 메모리와의 거리 D는 $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ 와 같이 정의 될 수 있다. 여기서 xi, yi는 각각 mi의 x좌표와 y좌표이다.

이와 같이 만약 고장 정보를 저장한 메모리와 고장을 수리할 메모리의 거리가 D보다 크다면, 그 메모리가 아닌 다른 메모리를 이용하여 고장을 수리하여야만 한다. 그러므로 라우팅을 통해 메모리의 거리를 계산하여 고장 정보를 저장할 메모리와 저장된 정보를 이용할 메모리의 거리를 D 이하로 모두 매칭을 시켜야만 한다. 만약 메모리 중에 수리 불가능한 메모리가 발생한다면, 그 메모리는 사용할 수 없기 때문에 존재하지 않는 메모리로 간주하고 라우팅을 진행하게 된다.

그림 5는 그림 4와 같은 메모리 그룹의 예제로써, 비

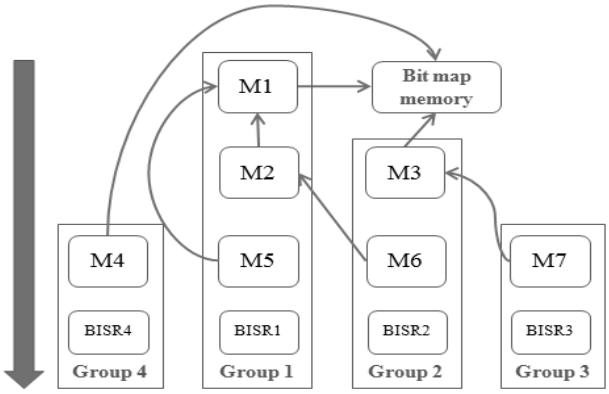

그림 5. 거리에 따라 비트맵을 공유하기 위한 라우팅 방법

Fig. 5. Routing method for bit-map sharing according the distance.

트맵 메모리를 M0이라고 지칭하고 M0부터 M7까지 각각의 메모리의 숫자가 증가할 때마다 거리가 1씩 증가한다고 가정하자. 그러면 M0과 M1의 거리는 1이 되고, M0와 M7의 거리는 7이 된다. 설계 상황을 고려하여 거리 D가 4라고 정의하면 모든 메모리는 고장정보를 저장할 메모리와의 거리가 4가 넘으면 안 된다. 그러므로 그림 4의 예에서 M7는 고장 정보를 저장할 메모리로써 M0를 이용하게 되는데 이렇게 되면 거리가 7이 되어 D인 4를 넘게되어 이용할 수 없게 된다. 그러므로 그림 5와같이 라우팅을 통하여 모든 메모리의 간격이 4 이하로 맞춰주게 되어 비트맵 메모리 하나만을 이용하여도 모든 메모리를 테스트하고 수리 할 수 있게 된다. 비록 라우팅을 위한 오버헤드가 추가되겠지만, 비트맵의 공유를 통한 하드웨어 오버헤드의 축소가 훨씬 크기 때문에 실제로 하드웨어 오버헤드는 줄어들게 된다.

3. 면적을 계산 하는 방법

본 논문에서 BIST는 [1]에서 제안하는 방식대로 공유를 하였기 때문에 BIST는 [1]과 같은 방식으로 면적을 계산하였다. 그 계산 방식은 [1]에 명시되어 있으므로 생략하도록 하겠다.

제안하는 방식은 BIRA의 면적을 줄이는 것으로서, BIRA에서는 비트맵의 면적이 대부분을 차지하기 때문에 본 논문에서는 이와 같은 비트맵의 크기에 따라서 면적을 계산하고 비교하였다. 비트맵 크기의 경우에는 알고리즘에 따라 비트맵의 면적 계산이 다르게 된다. 그래서 본 논문에서는 100% 수리 효율을 보이면서 면적이 가장 작은 BRANCH[6] 알고리즘에 따라서 BIRA의 면적을 계산하였다. 그 계산 방법은 다음 수식 (1)과 같다. 수식 (1)에서 RS과 CS는 리던던시 열과 리던던시 행의 수를 의미하고, M과 N은 메모리 열과 메모리 행의 비트수를 의미한다.

$$A_{BRANCH} = (R_S + C_S) \times (\log_2 M + \log_2 N + 3) + (R_S + (C_S - 1) + C_S \times (R_S - 1)) \times (\log_2(\max(M,N)) + \log_2(R_S + C_S) + 2) \quad (1)$$

위와 같은 수식들에 따라서 비트맵을 공유하지 않은 BIRA와 비트맵을 공유한 BIRA의 면적을 비교하였다.

Ⅲ. 실 험

앞서 설명한 바와 같이 BIST 면적의 크기는 [1]에서 사용한 메모리 공유법에 따라서 계산하였으므로, 본 논문에서는 생략하고 BIRA가 차지하는 면적에 대한 실험만을 진행하였다. BIRA의 면적을 계산하기 위해서 본 논문에서는 표 1에서 볼 수 있는 메모리들을 이용하였다. 첫 번째 열은 메모리의 번호를 의미하고, 두 번째 행은 각 메모리의 비트크기, 세 번째 행은 워드의 크기, 다섯 번째 행은 메모리 공유를 위해 같아야하는 주파수의 크기, 그리고 네 번째 행은 용량에 따른 BIRA의 크

표 1. 실험을 위해 이용한 메모리들의 정보
Table 1. Memory information.

No.	비트 크기	워드의 크기	주파수 (MHz)	메모리 용량(Mb)	블록의 수	위치	
						X	Y
1	16	128	133	10	10	10	10, 20, 30, 40, 50
2	16	128	133	10	10	20	
3	16	128	266	10	10	30	
4	16	128	266	10	10	40	
5	16	256	133	20	20	50	
6	16	256	133	20	20	60	
7	16	256	133	20	20	70	
8	16	256	133	20	20	80	
9	32	512	133	40	40	90	
10	32	512	133	40	40	100	

표 2. BIRA의 면적 비교
Table 2. Area comparison of BIRA.

메모리 수	그룹의 수	공유하지 않은 BIRA (#bit)	공유한 BIRA (#bit)	비트맵 수	비율 (%)
3	2	202301	134867	2	33.3
4	2	269734	134867	2	50.0
5	2	404602	202301	2	50.0
6	3	539469	202301	2	62.5
7	3	674336	202301	2	70.0
8	3	809203	269734	3	66.7
9	4	1078937	269734	3	75.0
10	4	1348672	269734	3	80.0
11	5	1416105	269734	3	81.0
12	5	1483539	269734	3	81.8
13	6	1550973	337168	4	78.3
14	6	1618406	337168	4	79.2
15	6	1753273	472035	5	73.1
20	8	2697344	539469	6	80.0
30	12	4046015	809203	9	80.0
40	16	5394687	1078937	12	80.0
50	20	6743359	1348672	15	80.0

기를 계산하기 위한 메모리 용량, 그리고 그에 따른 블록의 수는 여섯 번째 열에 그리고 마지막 두 열은 각 메모리의 위치를 변화시키기 위한 메모리 위치를 나타낸다. 각 메모리의 X좌표는 표에 나타난 바와 같고, Y의 수치는 메모리의 숫자가 늘어날 때마다 10씩 늘어나고 50이 넘으면 다시 10으로 돌아가 다시 차례대로 늘어나는 형식으로 변화시켰다. 그리고 본 논문에서는 메모리 블록의 고장을 수리하기 위해, 모든 메모리에 동등하게 5개씩의 여분의 row와 column을 가진 것으로 가정하였다. 표 1에서의 각기 다른 종류의 메모리 10종류를 가지고 메모리의 개수를 늘려가면서 실험하였다. 표 2는 그에 따른 결과를 나타낸 표이다.

표 2에서 볼 수 있는 것과 같이 메모리의 수는 3개에서 50개까지 표 1의 메모리들을 이용하여 늘려가면서 실험하였다. 두 번째 열에서 볼 수 있는 것은 메모리의 수를 늘려감에 따라 늘어나게 되는 그룹의 수이고, 세 번째 열은 비트맵을 공유하지 않았을 때 BIRA의 면적이고, 네 번째 열은 비트맵을 공유하였을 때 BIRA의 면적이다. 다섯 번째 열은 공유할 때 이용한 비트맵의 수를 보여준다. 본 논문에서의 모든 실험은 비트맵을 공유하는데 있어서, 거리 D는 50으로 정하고 각 메모리가 비트맵을 공유하는 데 있어서 거리가 50이 넘어가면 공유하지 못하도록 세팅하고 실험하였으며, 같은 주파수를 갖지 않는 메모리끼리는 비트맵을 공유하지 않았다. 그렇기 때문에 다섯 번째 열에서 볼 수 있는 것과 같이 공유하고 이용한 비트맵의 수가 그에 따라 달라지는 것을 확인 할 수 있다. 본 논문에서 제안하는 것과 같이 비트맵을 공유하였을 때 그에 따른 BIRA 면적의 줄어든 비율이 마지막 열에 나타난다. 마지막 표에서 볼 수 있는 것과 같이 비트맵을 공유하였을 때, BIRA의 면적은 개수가 늘어날수록 줄어드는 비율이 늘어나게 되고 20개 이상에서는 80%로 동일한 것을 볼 수 있다. 그러므로 본 논문에서 제안하는 비트맵 공유방법을 이용하면 대략적으로 BIRA의 면적을 80%정도 줄어들게 한다고 이야기할 수 있다. 기존의 논문들은 BIST의 공유만을 제안하였지만, 이와같이 BIST의 면적뿐만 아니라 BIRA의 면적 또한 줄임으로써 전체적인 하드웨어 오버헤드를 크게 줄일 수 있다.

Ⅳ. 결 론

시스템 온 칩 기술 개발이 발달함에 따라 집적도가 증가함에 따라 고장의 수 또한 증가하기 때문에 메모리의 고장을 테스트하고 수리하는 것은 필수 사항이 되었다. 본 연구에서는 수많은 임베디드 메모리를 테스트하고 수리하기 위해서 이용되는 BISR를 공유해서 테스트 및 수리를 위한 면적을 줄이는 방법을 제안하였다. BIRA에서 가장 큰 면적을 차지하는 비트맵을 공유함으로써, 직렬 또는 병렬로 연결된 메모리들을 테스트하고 수리해서, 면적을 크게 80%까지 줄일 수 있다는 것을 실험을 통하여 확인할 수 있었다.

참 고 문 헌

[1] M. Miyazaki, T. Yoneda, H. Fujiwara, “A memory grouping method for sharing memory BIST logic”, in Proc. of Asia and South Pacific Conf. 2006.

[2] P. Bernardi, L. Ciganda, “An Adaptive Low-Cost Tester Architecture Supporting Embedded Memory Volume Diagnosis”, IEEE Trans. on Instrumentation and Measurement, vol. 61, Issue 4, pp. 1002-1018, April, 2012.

[3] 김기철, 강성호, “내장된 자체 테스트를 위한 저전력 테스트 패턴 생성기 구조”, 전자공학회 논문지, 제47권 SD편, 제8호, 29-35쪽, 2010년 8월.

[4] H. Cho, W. Kang, and S. Kang, “A Built-in Redundancy Analysis with a Minimized Binary Search Tree”, ETRI Journal, vol. 32, no. 4, pp. 638-641, Aug, 2010.

[5] 정우식, 강우헌, 강성호, “불량 예비셀을 고려한 자체 내장 수리연산을 위한 분석 영역 가상화 방법”, 전자공학회 논문지, 제47권 SD편, 제11호 24-30쪽, 2010년 12월.

[6] W. Jeong, J. Lee, T. Han, K. Lee, S. Kang, “An Advanced BIRA for Memories With an Optimal Repair Rate and Fast Analysis Speed by Using a Branch Analyzer”, IEEE Trans. on Computer-Aided Design, Vol. 29, Issue 12, pp. 2014-2026, Dec., 2010.

[7] T.-W. Tseng, J.-F. Li, and C.-S. Hou, “A Built-in Method to Repair SoC RAMs in Parallel”, IEEE Trans. on Design & Test of Computers, vol. 27, issue 6, pp.46-57, Dec., 2010.

저 자 소 개

조 형 준(정회원)

2005년 연세대학교 전기전자공학과 학사 졸업.
2007년 연세대학교 전기전자공학과 석사 졸업.
2012년 현재 연세대학교 전기전자공학과 박사 과정.

<주관심분야 : Diagnosis, DFT, SoC 테스트>

강 성 호(정회원)

1986년 2월 서울대학교 제어계측공학과 학사 졸업.
1988년 6월 The University of Texas, Austin 전기 및 컴퓨터공학과 석사 졸업.
1992년 6월 The University of Texas, Austin 전기 및 컴퓨터공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원
1994년 Motorola Inc. 선임 연구원
2012년 현재 연세대학교 전기전자공학과 교수

<주관심분야 : SoC 설계 및 SoC 테스트>