고속 정적 RAM 명령어 캐시를 위한 방사선 소프트오류 검출 기법

준회원 권 순 규*, 최 현 석*, 정회원 박 종 강*, 김 종 태*

Radiation-Induced Soft Error Detection Method for High Speed SRAM Instruction Cache

Soongyu Kwon*, Hyun Suk Choi* *Associate Members*,
Jong Kang Park*, Jong Tae Kim* *Regular Members*

요 약

본 논문에서는 슈퍼스칼라 구조를 가진 시스템의 명령어 캐시에서 효율적으로 소프트오류를 검출할 수 있는 기법을 제안한다. 명령어 캐시로 주로 사용되는 고속 정적 RAM(Random Access Memory)에 적용할 수 있으며 1D 패리티와 인터리빙을 통해 기존 기법들과 비교하여 더 적은 메모리 오버헤드로 연집오류를 검출할 수 있다. 정적 RAM에서는 소프트오류의 발생만을 확인하고 검출된 소프트오류의 정정은 명령어 캐시의 캐시 미스와 같이 처리하여 하위 메모리로부터 명령어들을 다시 인출하는 방식이다. 이를 통해 명령어 캐시의 성능에 영향을 주지 않으면서 연집오류를 검출하고 정정할 수 있으며 최대 4×4의 윈도우 내에서 발생된 연집오류를 검출 할 수 있다. 제안된 방식을 이용하면 256비트 × 256비트 크기의 메모리에서 기존의 4-way 인터리빙 기법에서 검출에 필요한 패리티 크기의 25%만으로도 동일한 4비트의 연집오류를 검출 할 수 있다.

Key Words : Soft Error, Instruction Cache, Static Ram, Interleaving, Burst Error

ABSTRACT

In this paper, we propose multi-bit soft error detection method which can use an instruction cache of superscalar CPU architecture. Proposed method is applied to high-speed static RAM for instruction cache. Using 1D parity and interleaving, it has less memory overhead and detects more multi-bit errors comparing with other methods. It only detects occurrence of soft errors in static RAM. Error correction is treated like a cache miss situation. When soft errors are occurred, it is detected by 1D parity. Instruction cache just fetch the words from lower-level memory to correct errors. This method can detect multi-bit errors in maximum 4x4 window.

Ⅰ. 서 론

중성자 방사선에 의해 발생된 고에너지 중성자나 알파 입자들은 반도체 메모리 소자에 전하 축적을 유도하여 오류를 발생시킬 수 있다. 이와 같은 오류를 소프트오류라고 부르며 하드오류와는 달리 영구적인 손상을 주지 않는다[1]. 반도체의 p-n 접합에서 주로 발생되는 소프트오류는 메모리 소자에 저장된

※ 본 연구는 2007년 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2007-521-D00350).
* 성균관대학교 정보통신공학부 (jtkim@skku.edu)
논문번호 : KICS2010-05-212, 접수일자 : 2010년 5월 4일, 최종논문접수일자 : 2010년 6월 7일

값을 변경시키고 이로 인해 시스템 전체 운용에 치명적인 위험을 초래할 수 있다. 근래 VLSI(Very Large Scale Integration)설계에서는 나노 크기의 소자들이 고속·고집적화 되고 동작전압이 낮아짐에 따라 소프트오류의 발생 확률이 높아지고 있다[2]. 메모리 소자 중 동적 RAM의 경우 낮은 동작전압을 가지며 전하를 커패시터에 저장하므로 소프트오류가 발생할 확률이 높아 비교적 초기부터 소프트오류를 보정하기 위한 기법 개발과 구조 개선이 이루어졌다. 따라서 공정기술의 발전으로 공급 전압이 낮아지고 동작 속도가 증가하더라도 동적 RAM에서의 소프트오류 발생 빈도는 과거와 비슷한 수준으로 유지할 수 있었다[1]. 반면 시스템의 캐시로 주로 사용되는 정적 RAM의 경우 상대적으로 높은 동작전압을 사용하고 전하를 안정적으로 저장할 수 있었기 때문에 초기에는 소프트오류에 안정적인 특성을 보였다. 그러나 공정기술이 발전함에 따라 소프트오류에 취약하게 되었으며 특히 단일 입자로 인한 2비트 이상의 연집 오류 발생 비중이 점차 커지고 있다[3][4][5]. 반도체 공정 기술이 발전 할수록 이러한 현상이 더욱 심화될 것으로 예측되고 있다. 이러한 문제를 해결하기 위한 방법으로 시스템의 신뢰성을 높이기 위해 소프트오류를 검출하고 정정하는 기법이 많이 연구되고 있다. 본 논문에서는 메모리 소자 중 캐시 메모리에 널리 사용되는 정적 RAM에 대하여 연집오류에 강한 소프트오류 검출 기법을 제안하였다. 본 논문은 총 6장으로 구성된다. 2장에서는 소프트오류와 연집오류에 대해 소개하고, 소프트오류의 검출과 정정 기법에 관한 기존 연구와 한계를 3장에서 설명한다. 4장에서는 제안한 소프트오류 검출 기법에 대해 설명하고 5장에서 제안한 검출 기법을 정량적으로 분석한 후 6장에서 결론을 맺는다.

Ⅱ. 소프트오류

고 에너지 중성자나 알파 입자에 의한 소프트오류는 반도체의 p-n 접합에서 주로 발생한다. 고 에너지 입자가 입사되면 한시적으로 여기된 전류에 의해 저장된 값이 변화하게 된다. 이러한 현상은 근래의 VLSI 설계가 고속, 고집적화 되고 나노 크기의 소자가 사용되면서 발생 확률이 더욱 커지고 있다. 특히 단일 입자의 입사로 인해 공간적으로 인접한 다수의 비트가 동시에 오류를 일으키는 연집오류가 많이 나타나고 있다[3][4][5]. 이런 연집오류의 발생 패턴에는 여러 형태가 있는데 이 중 영문자 L의 형태를 가진 3L 패턴이 가장 많이 나타난다[3]. 다시 말해 연집오류는 행, 열이나 대각 방향으로 연속해 발생하기 보다는 뭉쳐진 정방형에 가까운 형태로 발생한다는 것이다. 이와 같은 형태의 연집 오류를 검출하기 위한 방법으로는 패리티를 이용한 방식과 SECDED(Single Error Correction Double Error Detection)을 주로 사용하고 있다[6][7][8]. 중성자에 의한 소프트오류의 연집오류는 행, 열 방향이 아닌 즉 N×1이나 1×N 형태의 연집오류보다 3L 패턴과 같은 M×N 형태의 2차원적인 연집오류를 검출하고 정정하는 기법이 필요하다.

Ⅲ. 소프트오류 검출 및 정정 기법

명령어 캐시에 사용되는 정적 RAM에서의 오류 검출 및 정정은 홀수 패리티, SECDED, hsio 부호 기법 등을 주로 사용하고 있다[9]. 특히 상용으로 사용되는 정적 RAM의 경우 SECDED기법을 사용한다[7][8]. 지금까지 정적 RAM은 상대적으로 동적 RAM에 비해 소프트오류에 강인한 특성을 가졌기 때문에 이와 관련된 초기 연구는 SECDED 기법 자체를 수정 보완하여 속도를 개선하거나 하드웨어 오버헤드를 줄이는 방향으로 이루어졌다. 하지만 공정이 발전함에 따라 정적 RAM에서의 소프트오류 발생 가능성이 높아지고 그 중 연집오류가 차지하는 비중이 커지면서 근래에는 새로운 소프트오류 검출 및 정정 기법에 관한 연구가 많이 진행되고 있다. 연집오류를 검출하기 위해서 인터리빙 기법이 많이 사용된다. 기본적으로 메모리는 데이터를 워드 단위로 저장하기 때문에 인접한 여러 비트에 걸쳐 연집오류가 발생하게 되면 오류 검출 및 정정 능력의 한계를 넘어설 수 있다. 인터리빙 기법은 데이터 워드를 비트 단위로 분산시킴으로써 공간적으로 인접한 여러 비트에 연집오류가 발생하더라도 실제 하나의 워드에서 발생한 오류의 개수를 감소시킬 수 있는 방법이다.[6,10,11] 그림 1은 4-way 인터리빙 기법으로 각 워드의 비트들이 4 비트씩 떨어져 배치되어 있어 4 비트의 연집오류가 일어나도 각 워드는 1 비트의 오류만을 갖게 되어 1 비트 패리티로 각 워드의 오류를 검출할 수 있다. 2D 패리티 기법은 수평 패리티를 이용해 데이터 워드의 오류를 검출하고 수직 패리티를 이용해 오류를 정정한다. 따라서 오류를 검출하고 정정하기 위해서 전체 메모리를 읽고 정정된 값을 써야 하므로 N개의 메

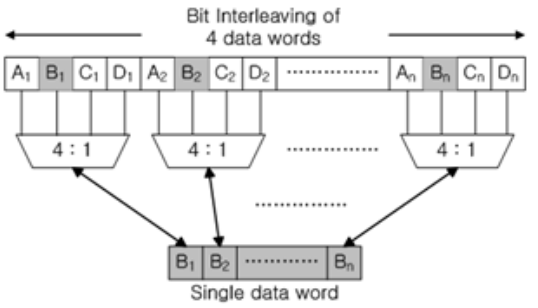

그림 1. 4-way 인터리빙 기법

모리 주소가 있을 경우 총 N+1 싸이클이 필요하게 된다. 하지만 XOR 연산만을 이용하므로 하드웨어 오버헤드는 작다고 할 수 있다. 여기에 인터리빙을 적용하면 연집오류도 검출하고 정정할 수 있다. SECDED는 해밍코드를 기반으로 만들어졌다. 해밍 코드에 1비트가 추가된 형태로 두 개의 오류를 검출할 수 있고 하나의 오류를 정정할 수 있다. SECDED는 2D 패리티 기법에 비해 더 많은 하드웨어 오버헤드를 가지고 느린 레이턴시를 가지지만 오류 정정이 쉬운 장점이 있다. 그 밖에 SECDED와 2D 패리티를 복합적으로 이용한 기법도 제안되었는데 이 방법은 인터리빙이 적용된 SECDED에 2D 패리티의 수직 패리티를 추가하여 정정 가능 범위를 늘리는 기법이다[6]. 그림 2는 4-way 인터리빙 기법을 적용한 256비트 × 256비트 크기의 일반적인 메모리 구조이다. 64비트 당 하나의 패러티 비트를 부가하고 64비트의 워드에 속한 각 비트는 4비트씩 떨어져 있으므로 각 행 마다 4비트까지의

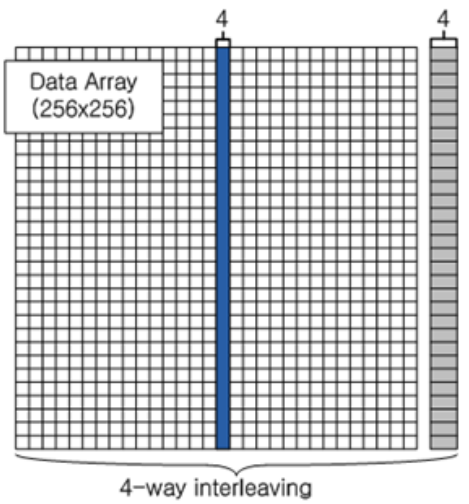

그림 2. 4-way 인터리빙 기법을 적용한 통상적인 메모리 구조

연집오류를 검출할 수 있다.

Ⅳ. 제안하는 소프트오류 검출 기법

본 논문에서 제안하는 소프트오류 검출 기법은 명령어 캐시에서 사용되는 고속 정적 RAM에 적용할 수 있는 기법이다. 소프트오류 검출 기법은 1 워드가 16비트인 슈퍼스칼라 구조의 시스템이라는 가정 하에서 1D 패리티와 인터리빙을 기반으로 하고 있다. 그림 3과 같이 한 라인에 네 개의 워드의 4 비트가 인터리빙 되어 있고 하나의 패리티 비트가 추가되어 있다. 그림에서 숫자는 워드 번호를 나타내고 네 라인에 걸친 4×4 윈도우는 같은 위치의 비트를 의미한다. 그림에서 p로 표시된 비트는 홀수 패리티이다. 네 라인에 걸쳐 총 16개의 워드의 4 비트가 분산되어 형태이다. 또 다시 네 라인에 걸쳐 16 워드의 4 비트가 분산 되어 있는데 이때는 1 비트만큼 왼쪽으로 순환 이동된 형태이다. 따라서 16 워드를 저장하기 위해서 총 16 라인이 필요하며 이를 클러스터라 부른다. 네 라인 단위로 패리티 비트를 제외하고 1 비트씩 왼쪽으로 순환 이동된 형태를 가지고 있다. 패리티 비트는 각 워드에 대한 패리티로 역시 인터리빙 되어 배치되어 있다.

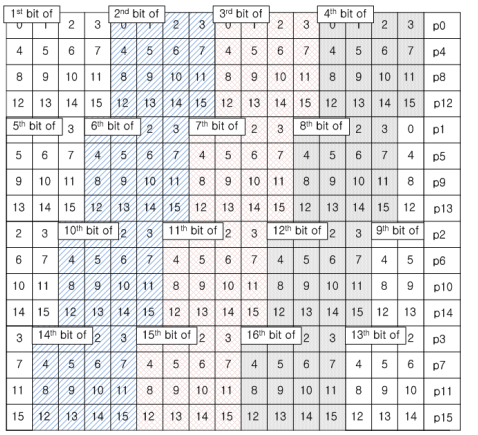

그림 3. 제안된 소프트오류 검출 기법

Ⅴ. 제안한 기법에 대한 분석 및 정량적 결과

블록 B_i는 $N_{burst} \times N_{burst}$의 비트 행렬로 구성되며, 하나의 클러스터(cluster) 당 $N_{row} \times N_{col}$ 개의 블록이 존재한다. 하나의 클러스터 내 블록 B_i $(0 \le i < N_{burst}^2)$는 다음과 같다.

$$B_i' = \{b'_{i,j} | \, i \; is \; bit \; number \text{ and } j \; is \; word \; number, 0 \le j < N_{burst}^2\}$$

(1) $0 \le i < N_{col}$ 인 경우

$$B_i = \{b_{i,j} | b_{i,j} = b'_{i,j}, 0 \le j < N_{burst}^2\} \quad (1)$$

(2) $p \cdot N_{col} \le i < (p+1) \cdot N_{col}$ 인 경우
(단, $1 \le p < N_{burst}$인 정수)

$$B_i = \left\{ b_{i,j} \left| \begin{array}{ll} b_{i,j} = b'_{i,j+p}, & (q-1)N_{burst} \le j < qN_{burst} - p \\ b_{i,j} = b'_{i+1,j-N_{burst}+p}, & qN_{burst} - p \le j < qN_{burst}, \\ & i+1 < (p+1) \cdot N_{col} \\ b_{i,j} = b'_{pN_{col},j-N_{burst}+p}, & qN_{burst} - p \le j < qN_{burst}, \\ & i+1 = (p+1) \cdot N_{col} \end{array} \right. \right\} \quad (2)$$

(단, $1 \le q \le N_{burst}$인 정수)

여기에서, $b_{i,j}$는 블록 B_i의 j번째 비트를 의미한다. B_i의 가로 길이와 세로 길이는 모두 N_{burst}이다. 1 클러스터에 존재하는 워드 번호와 비트 번호는 모두 N_{burst}^2이며, 모두 N_{burst}^4의 비트로 구성된다. 즉, 식(2)에서 B_i는 B_i'로 구성된 메모리의 한 라인을 p만큼 왼쪽으로 순환 이동한 상태에서 구성된 $N_{burst} \times N_{burst}$의 단위 블록과 동일하다. 여기에 메모리의 각 라인당 1개의 패리티를 마지막 우측열의 오른쪽에 위치시킨다. 각 라인의 패리티는 가장 왼쪽 열의 워드 번호와 일치하는 워드의 홀수 패리티로 정의된다. 즉, 왼쪽 열의 비트가 b_{i,w_0}, $i = pN_{col}$ 라면, 패리티는 N_{burst}^2 비트를 가지는 w_0 워드의 홀수 패리티로 생성된다.

식 (1), (2)에 따라 인터리빙된 메모리에서 메모리 클러스터 L_p에 대한 bit집합은 다음과 같이 정의된다.

(1) $0 \le i < N_{col}$ 인 경우

$$\begin{aligned} L_p &= \bigcup \left[\bigcup_{i=pN_{col}}^{(p+1)N_{col}} B_i, \; OP_{(q-1)N_{burst}} \right] \\ &= \bigcup \left[\bigcup_{i=pN_{col}}^{(p+1)N_{col}} \left\{ \bigcup_{n=(q-1)N_{burst}}^{qN_{burst}-1} b'_{i,n} \right\}, \; OP_{(q-1)N_{burst}} \right] \end{aligned} \quad (3)$$

(단, $0 \le p < N_{burst}$, $1 \le q \le N_{burst}$, p, q는 정수)

(2) $p \cdot N_{col} \le i < (p+1) \cdot N_{col}$,
$1 \le p < N_{burst}$, (p는 정수) 인 경우

$$\begin{aligned} L_p &= \bigcup \left[\bigcup_{i=pN_{col}}^{(p+1)N_{col}-1} B_i, \; OP_{(q-1)N_{burst}} \right] \\ &= \left[\begin{array}{l} \bigcup_{i=pN_{col}}^{(p+1)N_{col}-1} \left\{ \bigcup_{n=(q-1)N_{burst}}^{qN_{burst}-p-1} b'_{i,n+p}, \bigcup_{n=qN_{burst}-p}^{qN_{burst}-1} b'_{i+1,n-N_{burst}+p} \right\} \cup \\ \left\{ \bigcup_{n=(q-1)N_{burst}}^{qN_{burst}-p-1} b'_{(p+1)N_{col}-1,n+p}, \bigcup_{n=qN_{burst}-p}^{qN_{burst}-1} b'_{(p+1)N_{col},n-N_{burst}+p} \right\} \cup OP_{(q-1)N_{burst}+p} \end{array} \right| \end{aligned} \quad (4)$$

(단, $0 \le p < N_{burst}$, $1 \le q \le N_{burst}$, p, q는 정수)

여기에서 OP_n은 n 워드의 홀수 패리티를 의미한다. 식 (3)에서 알 수 있듯이, 각 블록 내의 비트 $b'_{i,n}$은 패리티를 포함하여 $(q-1)N_{burst} \le n \le qN_{burst}-1$의 서로 다른 워드로부터 반복적으로 구성된다. 따라서 N_{burst} 비트 마다 동일한 워드의 비트가 존재하게 되며, $N_{burst}-1$ 비트 내에는 동일한 워드의 비트가 위치하지 않는다. 식 (4)에서 패리티를 포함한 비트 라인은 p 순환 이동에 따라 각각 $(q-1)N_{burst}+p \le n \le qN_{burst}-1$, $(q-1)N_{burst} \le n \le (q-1)N_{burst}+p-1$의 두 가지 워드 번호로 구성된다. 두 범위를 합치면, $(q-1)N_{burst} \le n \le qN_{burst}-1$가 되며, 식 (3)의 경우와 마찬가지로 $N_{burst}-1$ bit 내에는 동일한 워드의 비트가 위치하지 않는다. 그러므로 임의의 $b_{i,j}$에 대하여, 가로 방향으로 N_{burst} 비트 내에 동일한 워드의 비트가 존재하지 않는다. 또한 임의의 $b_{i,j}$에 대하여, 세로 방향으로 N_{burst} 라인 내에 동일한 워드의 비트가 존재하지 않으므로 제안된 메모리 구조는 최대 $N_{burst} \times N_{burst}$의 연집오류를 검출할 수 있다.

제안하는 기법은 명령어 캐시로 사용된 정적 RAM에 적용할 수 있다. 캐시는 캐시 미스가 발생되면 하위 메모리에서 해당하는 데이터를 인출하게 된다. 소프트오류의 정정에서도 이와 같은 방법을 사용할 수 있다. 소프트오류의 발생확률은 0.0001 ~ 0.01 FIT/bit 정도로 1 FIT(Failure in Time)은 10^9 시간에 소프트오류가 한번 발생할 확률을 의미한다[2][12]. 시스템 전체 동작 시간과 비교하여 소프트오류가 발생하고 이를 정정하는데 드는 시간은 큰 비중을 차지하지 않는다는 것이다. 따라서 정적 RAM이 캐시로 사용될 때에는 소프트오류를 검출하고 그것을 정정하기 위한 기법을 적용하여 캐시 성능을 저하시키기 보다는 연집오류를 검출할 수 있는 기법만을 적용하여 오류 발생 유무만 확인하고 오류가 발생하면 캐시 미스가 발생한 것과 같이 하위 메모리에서 오류가 발생한 부분의 데이터를 다시 인출한다면 적은 하드웨어 오버헤드로도 소프트오류

에 강인한 정적 RAM을 구현할 수 있다.

제안된 기법은 4개의 라인에 4개의 워드가 분산되어 있다. 따라서 하나의 워드를 읽기 위해서는 4번의 캐시 메모리 접근이 필요하다. 하지만 앞선 가정에서와 같이 정적 RAM이 슈퍼스칼라 구조에서 명령어 캐시로 사용된다면 이러한 레이턴시 오버헤드는 무시 할 수 있다. 일반적으로 명령어는 연속적이므로 슈퍼스칼라 구조와 같이 복수의 명령어를 동시에 읽는 시스템에서는 4 싸이클 동안 캐시의 4라인을 읽어와 총 4개의 워드를 완성할 수 있으므로 이로 인한 오버헤드는 무시 할 수 있다. 표 1은 대표적인 소프트 오류 검출 및 정정 기법을 256비트 × 256비트 크기의 메모리에 적용했을 때 필요한 패리티 비트의 크기이다. 제안한 소프트오류 검출 방식은 4비트의 연집오류를 검출하기 위해서 256비트의 패리티 비트가 필요하다. 1,024비트가 필요한 기존의 4-way 인터리빙 기법에 비해 25% 크기로 같은 4비트 연집오류를 검출할 수 있다.

표 1. 부가 패리티 크기 (256×256 비트 메모리)

	패리티크기 (bit)	기능
4-way 2D 인터리빙 기법	2,048	검출 및 정정
SEC-DED	8,192	검출 및 정정
4-way 인터리빙 기법	1,024	검출
제안된 방식	256	검출

Ⅵ. 결 론

본 논문에서는 고속 정적 RAM에서 효율적으로 소프트오류를 검출할 수 있는 기법을 제안하였다. 제안된 기법은 슈퍼스칼라 구조의 명령어 캐시로 사용되는 정적 RAM에 적용할 수 있으며 1D 패리티와 인터리빙을 통해 기존 기법들과 비교하여 더 적은 메모리 오버헤드로 연집오류를 검출할 수 있다. 공정 기술 발전에 따라 소프트오류 발생 확률이 증가하고 그 중 2 비트 이상의 연집오류 발생 비중이 커지고 있지만 전체 시스템의 동작 시간에 비해 연집오류의 발생과 정정에 필요한 시간은 매우 짧다고 할 수 있다. 따라서 정적 RAM을 사용한 명령어 캐시에서는 소프트오류의 발생만을 확인하고 검출된 오류의 정정은 캐시 미스와 같이 처리하여 하위 메모리로부터 명령어들을 다시 인출한다면 캐시의 성능에 영향을 주지 않으면서 연집오류를 검출하고 정정할 수 있다. 인터리빙으로 인한 레이턴시 오버헤드는 슈퍼스칼라 구조 하에서 무시될 수 있으며 최대 4×4 내에서 어떤 형태의 연집오류가 발생하더라도 소프트오류가 발생한 워드를 검출할 수 있다.

참 고 문 헌

〔1〕 R. C. Baumann, "Soft Errors in Commercial Integrated Circuits," Int. J. of High Speed Electronics and Systems, Vol.14, No.2 pp.299-309, 2004.

〔2〕 T. Granlund, B. Granbom, N. Olsson, "Soft Error Rate Increase for New Generations of SRAMs", IEEE Trans. Nuc. Sci., 50, 6, pp. 2065-2068, Dec. 2003.

〔3〕 J. Marz, S. Hareland, K. Zhang and P. Armstrong, "Characterization of Multi-bit Soft Error events in advanced SRAMs," IEEE Int. Electron Devices Meeting 2003, pp.21..4.1-21.4.4, 2003.

〔4〕 S. Buchner et. al., "Investigation of single-ion multiple-bit upsets in memories on board a space experiment," IEEE Trans. on Nuclear Science, Vol.47, No.3, pp.705-711, 2000.

〔5〕 F.X. Ruckerbauer, G. Georgakos, "Soft Error Rates in 65nm SRAMs--Analysis of new Phenomena," IEEE int. conf. on IOLTS'07, pp.203-204, 2007.

〔6〕 Jangwoo Kim, et al., "Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding", MICRO 2007,40th Annual IEEE/ACM International Symposium on 1-5 Dec.pp.197-209, 2007.

〔7〕 J. Borkenhagen and S. Storino. 5th Generation 64bit PowerPCcompatible, Commercial Processor Design. IBM white paper,1999.

〔8〕 W. Bryg and J. Alababo. "The UltraSPARC T1 Processor-reliability, availability, and serviceability". SUN white paper, Dec 2005.

〔9〕 M.Hsiao, "A Class of Optimal Minimum

Odd-weight-column SEC-DED codes", IBM J. Res Dev.14, pp.395-401, July 1970.
〔10〕 Ritesh Mastipuram and Edwin C Wee, "Soft errors impact on system reliability", 2004, september.30, EDN
〔11〕 L. D. Huang, M. Goshima and S. Sakai, "Zigzag-HVP: A Cost Effective Technique to Mitigate Soft Errors in Caches with Word-based Access," IPSJ, Vol.2, pp.748-758, 2006.
〔12〕 R. Baumann, "The Impact of Technology Scaling on soft Error Rate Performance and Limits to the Efficacy of Error Correction", IEDM Tech. Dig., pp.329- 332, 2002.

권 순 규 (Soongyu Kwon) 준회원
한국통신학회 논문지 제 34권 5호 참조
현재 성균관대학교 휴대폰학과 박사 과정

최 현 석 (Hyun Suk Choi) 준회원

2007년 2월 성균관대학교 정보통신공학부
2009년 2월 성균관대학교 전자전기컴퓨터공학과 석사
<관심분야> 유/무선 네트워크, 데이터 통신

박 종 강 (Jong Kang Park) 정회원

2001년 2월 성균관대학교 전기전자 및 컴퓨터공학부
2003년 2월 성균관대학교 전기전자 및 컴퓨터공학과 석사
2008년 8월 성균관대학교 전자전기공학과 박사
<관심분야> 임베디드 시스템, 소프트오류

김 종 태 (Jong Tae Kim) 정회원
한국통신학회 논문지 제 34권 5호 참조
현재 성균관대학교 정보통신공학부 교수