메모리 소자의 소프트 에러 극복을 위한 최적 스크러빙 방안

류상문*

*군산대학교

An Optimal Scrubbing Scheme for Protection of Memory Devices against Soft Errors

Sang-moon Ryu*

*Kunsan National University

E-mail : smryu@kunsan.ac.kr

요 약

우주 방사선은 메모리 시스템에 소프트 에러를 야기할 수 있다. 소프트 에러는 오류 검출 및 정정 코드를 이용하여 극복될 수 있으며, 소프트 에러의 누적을 방지하기 위하여 스크러빙 작업이 병행되어야 한다. 본 논문은 CPU의 쓰기 동작 없이 소프트 에러를 정정할 수 있는 자가 오류 검출 및 정정 회로가 적용된 메모리 시스템에 적용할 수 있는 최적 스크러빙 수행 방안을 제안한다. 제안된 스크러빙 방안은 시스템의 가용한 스크러빙 로드와 시스템에서 실행되는 태스크의 주기적 메모리 접근을 함께 고려하여 최대의 신뢰도를 성취할 수 있도록 하여준다.

ABSTRACT

Error detection and correcting codes are typically used to protect against soft errors. In addition, scrubbing is applied which is a fundamental technique to avoid the accumulation of soft errors. This paper introduces an optimal scrubbing scheme, which is suitable for a system with auto error detection and correction logic. An auto error detection and correction logic can correct soft errors without CPU's writing operation. The proposed scrubbing scheme leads to maximum reliability by considering both allowable scrubbing load and the periodic accesses to memory by the tasks running in the system.

키워드

소프트 에러 극복, 최적 스크러빙, 신뢰도 개선

Ⅰ. 서 론

컴퓨터 메모리 시스템에서 발생하는 소프트 에러는 컴퓨터 시스템의 신뢰성에 가장 큰 악영향을 미친다. 메모리 소자의 집적도가 높아지고 저전력 설계로 인해 동작 전압이 낮아지면서 소프트 에러의 영향이 지속적으로 증가하고 있다[1][3][4].

메모리의 소프트 에러는 오류 검출 및 정정(EDAC: Error Detection & Correction) 회로[5]를 이용하여 정정할 수 있다. CPU가 메모리에 정보를 저장할 때는 추가의 정보(패리티)가 함께 저장되고, 정보를 읽을 때에는 이 추가 정보를 함께 읽어 들여 소프트 에러 발생 여부를 검출하고 일정 범위 이내에서 소프트 에러를 정정할 수 있다.

그리고 소프트 에러의 발생 정도가 적용된 오류 검출 및 정정 회로의 정정 범위 이내에서 유지되도록 메모리의 모든 워드에 대해 스크러빙(Scrubbing)[1][6][7][8]을 수행해 준다. 스크러빙의 주기가 짧을수록 정정 범위를 벗어나는 소프트

에러의 누적을 방지할 수 있다. 하지만 빈번한 스크러빙은 CPU 및 버스 자원을 많이 사용하여 시스템의 성능을 떨어뜨리게 된다. 따라서 주어진 오류 검출 및 정정 회로에 대해 시스템의 성능을 고려하면서 신뢰도를 최대로 할 수 있는 스크러빙 주기를 적용하여야 하며 이와 관련된 연구가 진행되었다.

그림 1처럼 CPU가 메모리의 정보를 읽을 때 소프트 오류가 검출되면 CPU의 개입 없이 오류 검출 및 정정 회로 스스로 오류가 제거된 정보를 메모리에 기입하도록 구성할 수 있다. 본 논문에서는 이런 기능이 있는 회로를 '자가 복구 오류 검출 및 정정 회로(Auto Error Detection & Correction logic)라고 부르기로 한다.

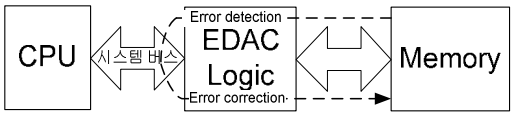

그림 1. 자가 복구 오류 검출 및 정정회로

자가 복구 오류 검출 및 정정 회로가 적용되면 오류가 제거된 정보를 메모리에 기입하기 위해 CPU가 메모리를 대상으로 쓰기 동작을 수행할 필요가 없기 때문에 CPU의 스크러빙 로드(Scrubbing load)를 경감할 수 있다. 특히 메모리 영역 중에서 항상 읽기 동작만 일어나는 프로그램 코드 영역이나 상수 데이터 영역 등에서는 그 경감 효과가 클 것이다.

그리고 주기적으로 실행되는 태스크에 할당된 메모리 영역은 CPU가 해당 태스크를 수행할 때 자동으로 스크러빙이 이루어지게 된다. 따라서 시스템 내에서 가용한 스크러빙 로드를 비주기적인 태스크의 메모리 영역이나 평소에는 접근되지 않는 메모리 영역의 스크러빙에 사용하면 시스템의 메모리 신뢰도를 높일 수 있을 것이다. 본 논문은 자가 복구 오류 검출 및 정정 회로로 보호되는 메모리를 갖는 컴퓨터 시스템에 적용할 수 있는 메모리 신뢰성을 고려한 최적 스크러빙 기법을 제안한다.

II. 신뢰도 성능 해석

각각 N_i 메모리 워드를 점유하고 t_i의 주기를 갖는 태스크 $\tau_i(i=1,2,...,n)$ 가 실행되는 컴퓨터 메모리 시스템을 가정한다. 각 메모리 워드의 크기는 정보 저장을 위한 w비트와 패리티 저장을 위한 c비트로 구성되어 있고, 평균값 λ를 갖는 포아송 분포에 따라 발생하는 소프트 에러의 영향을 받는다고 가정한다. 각 워드당 1비트의 소프트 에러가 정정될 수 있는 오류 검출 및 정정 코드 성능을 가정한다.

$\lambda \ll 1$인 경우 하나의 워드에 대한 신뢰도 함수는 식(1)과 같다.

$$r_o(t) = e^{-\lambda(w+c)t} + (w+c)(1-e^{-\lambda t})e^{-\lambda(w+c-1)t}$$
$$\approx 1 - \frac{1}{2}(w+c)(w+c-1)\lambda^2 t^2,\ t \geq 0 \quad (1)$$

편의를 위하여 $\alpha = \frac{1}{2}(w+c)(w+c-1)\lambda^2$

를 정의하면 식(1)은 다음과 같이 된다.

$$r_o(t) \approx 1 - \alpha t^2,\ t \geq 0 \quad (2)$$

식 (2)로부터 태스크 τ_i가 사용하는 메모리 영역에 대한 신뢰도 함수는 다음과 같다.

$$R_{o,i}(t) = (r_o(t))^{N_i} \approx 1 - \alpha N_i t^2,\ t \geq 0 \quad (3)$$

태스크 τ_i가 매 t_i마다 주기적으로 실행되면 t_i마다 스크러빙이 수행되는 것으로 간주할 수 있다. 스크러빙이 적용되면 t_i마다 정정 가능한 범위의 소프트 오류들이 정정되므로 신뢰도 함수는 매 스크러빙 작업이 완료되는 시점에서 모든 오류가 정정되고, 최근 스크러빙 작업 이후에 정정 불가능한 오류가 발생하지 않을 확률과 같다. 따라서 스크러빙 효과를 고려한 태스크 τ_i의 메모리에 대한 신뢰도 함수는 다음과 같다.

$$R_{t,i}(t) = [R_{o,i}(t_i)]^l R_{o,i}(\tau) \quad (4)$$
$$\approx (1-\alpha N_i t_i^2)(1-\alpha N_i \tau^2)$$
$$\approx (1-\alpha N_i t_i^2)^{\frac{t}{t_i}},\ t \geq 0$$

여기서 $t = t_i l + \tau$ 이고 τ는 $0 \leq \tau < t_i$ 를 만족하는 실수이며 l은 $l \geq 0$를 만족하는 정수이다.

전체 메모리에 대한 MTTF는 그 정의에 따라 식(4)를 이용하여 구하면 다음과 같다.

$$MTTF_t = \int_0^\infty \prod_{i=1}^{n} R_{t,i}(t)\,dt \quad (5)$$
$$\approx -\frac{1}{\sum_{i=1}^{n} \frac{1}{t_i} ln(1-\alpha N_i t_i^2)}$$

III. 최적 스크러빙 방안

메모리 스크러빙에 따른 부담은 스크러빙 대상 메모리의총 워드 수에 비례하고 스크러빙 주기에 반비례한다. 따라서 시스템에 요구되는 모든 작업을 수행하면서 전체 메모리를 T_s의 주기로 스크

러빙할 수 있다면 가용한 스크러빙 로드를 다음과 같이 표현할 수 있다.

$$\frac{1}{T_s}\sum_{i=1}^{n} N_i \tag{6}$$

식 (6)로 표현된 시스템의 가용한 스크러빙 로드를 이용해 각 태스크에 할당된 메모리 영역을 추가로 스크러빙하면 전체 메모리의 신뢰도를 개선할 수 있을 것이다. 그리고 가용한 스크러빙 로드를 유지하면서 전체 메모리의 신뢰도를 최대화하는 각 태스크 영역별 최적 스크러빙 주기가 존재할 것이다. 추가의 스크러빙이 적용된 각 태스크별 스크러빙 주기를 $T_i(\le t_i, i=1,2,...,n)$ 라고 하면 이를 위해 다음과 같은 최적화 문제를 구성할 수 있다.

$$\max imize - \frac{1}{\sum_{i=1}^{n}\frac{1}{T_i} ln(1-\alpha N_i T_i^2)} \tag{7}$$

$$such\ that \sum_{i=1}^{n}\frac{N_i}{T_i} = \sum_{i=1}^{n}\frac{N_i}{t_i} + \frac{1}{T_s}\sum_{i=1}^{n} N_i \tag{8}$$

식 (7)는 식 (5)의 MTTF에서 t_i를 T_i로 대체한 것으로 각 태스크의 스크러빙 주기 T_i가 전체 메모리의 신뢰도를 최대화하는 것을 최적화 목적 함수로 설정한 것이다. 식 (8)의 좌변은 각 태스크를 스크러빙 주기 T_i로 스크러빙하는 총 스크러빙 로드를 의미하여, 우변은 각 태스크가 각자의 실행 주기 마다 실행되어 자동으로 수행되는 스크러빙 양과 식 (6)의 가용한 스크러빙 로드의 합을 의미한다.

$0 < x \ll 1$ 을 만족하는 x에 대해 $\ln(1-x) \approx -x$ 이므로 식 (7)의 목적 함수는 다음과 같이 바꿀 수 있다.

$$\min imize \sum_{i=1}^{n} N_i T_i \tag{9}$$

식 (8)과 (9)로 이루어진 최적화 문제를 Lagrange 정리를 적용하여 풀면 다음의 연립 방정식이 얻어진다.

$$N_j + \kappa\left(\sum_{i=1, i\neq j}^{n}\frac{N_i}{T_i} - \sum_{i=1}^{n}\frac{N_i}{t_i} - \frac{1}{T_s}\sum_{i=1}^{n} N_i\right) = 0,$$
$$j = 1, 2, \cdots, n \tag{10}$$

$$\sum_{i=1}^{n}\frac{N_i}{T_i} - \sum_{i=1}^{n}\frac{N_i}{t_i} - \frac{1}{T_s}\sum_{i=1}^{n} N_i = 0 \tag{11}$$

여기서 κ는 Lagrange multiplier이다.

식 (10)과 (11)를 만족하는 $T_i(>0, i=1,2,\cdots,n)$ 를 구하는 것이 가용한 스크러빙 로드를 유지하면서 전체 메모리의 신뢰도를 최대화하는 각 태스크 영역별 최적 스크러빙 주기를 구하는 것이다. 그런데 구해진 태스크 영역별 스크러빙 주기 T_i 중에 $T_i \ge t_i$를 만족하는 태스크 τ_i가 있을 수 있다. 구해진 최적 스크러빙 주기가 태스크의 실행 주기보다 크거나 같다는 것은 이 태스크에 대해 추가적인 스크러빙이 필요 없음을 의미한다. 따라서 만일 식 (10)과 (11)의 해 중 $T_i \ge t_i$를 만족하는 것이 존재하면 T_i를 $T_i = t_i$로 대입하여 식 (10)과 (11)를 다시 구성하고, 해 T_i중 $T_i \ge t_i$를 만족하는 것이 없을 때까지 이 과정을 반복한다.

$T_i < t_i$를 만족하는 태스크 τ_i에 대해서는 메모리 시스템의 신뢰도 향상을 위해서 추가의 스크러빙이 필요하다는 것을 의미한다. 태스크 τ_i를 위한 추가 스크러빙 수행 주기를 e_i라고 하면, T_i와 t_i 그리고 e_i의 관계는 식 (12)로 표현되며, 이로부터 e_i는 식 (13)과 같이 구할 수 있다.

$$\frac{N_i}{T_i} = \frac{N_i}{t_i} + \frac{N_i}{e_i} \tag{12}$$

$$e_i = \frac{T_i t_i}{t_i - T_i} \tag{13}$$

최적 스크러빙이 적용되는 메모리 시스템의 MTTF는 각각 식 (5)에서 t_i를 T_i로 대체한 것과 같다.

Ⅳ. 성능 평가

표 1과 같은 실행 주기가 1, 5, 10인 3개의 주기적 태스크와 1 개의 비주기적 태스크가 실행되는 상황을 가정한다. 비주기적인 태스크 4의 주기는 ∞로 표현한다. 실행 주기와 스크러빙 주기 설정에 따른 효과를 비교하기 위해서 태스크에 할당된 메모리 워드의 양는 동일하게 10^5로 가정한다.

표 1. 성능 평가를 위해 가정된 시스템

Task #	Period [sec]	# of Words
1	1	10^5
2	5	10^5
3	10	10^5
4	∞	10^5

각 워드의 크기는 7 비트 ($w=4$, $c=3$)이며, 워드에서 발생하는 1 비트 오류를 정정할 수 있는 자가 오류 검출 및 정정 회로를 가정한다. 소

프트 에러 발생률 $\lambda = 2 \times 10^{-8}$ [bit/sec]를 가정하고, 시스템에서는 모든 워드를 10초($T_s = 10$)의 주기로 스크러빙할 수 있는 성능 여유가 있다고 가정한다.

스크러빙 주기에 따른 효과 비교를 위하여 2가지의 경우를 고려한다.

Case 1: 가용한 스크러빙 로드의 여유, 즉 10초 주기로 모든 워드를 스크러빙할 수 있는 여력을 모두 비주기적인 태스크 4 영역을 스크러빙하는데 사용한다. 4×10^5 워드를 10초 주기로 스크러빙하는 것은 10^5 워드를 2.5초 주기로 스크러빙하는 것과 동일하기 때문에 태스크 4의 스크러빙 주기는 2.5가 된다. 나머지 태스크들은 추가의 스크러빙을 수행하지 않으므로 태스크의 실행 주기가 스크러빙 주기가 된다. 각 태스크의 스크러빙 주기와 추가 스크러빙 주기가 표 2에 있다.

Case 2: 본 논문에서 제안한 최적화 방안을 적용한 경우로 태스크 1, 2, 3, 4의 영역을 각각 1, 4.3, 4.3, 4.3초로 스크러빙한다. 태스크 1은 추가 스크러빙이 필요 없으며, 4×10^5 워드를 10초 주기로 스크러빙하는 로드를 태스크 2, 3, 4을 각각 30.7, 7.5, 4.3초 추가 스크러빙하는데 할당한다. 각 태스크의 스크러빙 주기와 추가 스크러빙 주기가 표 3에 있다.

표 2. Case 1: 가용한 스크러빙 로드가 태스크 4의 스크러빙에 할당된 경우

Task #	Scrubbing period [sec]	Extra scrubbing period [sec]
1	1	0
2	5	0
3	10	0
4	2.5	2.5

표 3. Case 2: 제안된 최적 스크러빙 방안 적용

Task #	Scrubbing period [sec]	Extra scrubbing period [sec]
1	1	0
2	4.3	30.7
3	4.3	7.5
4	4.3	4.3

표 4는 두 경우에 대한 메모리 시스템의 MTTF를 보여준다. MTTF가 33% 개선되는 것을 알 수 있다.

표 4. MTTF 비교

	Case 1	Case 2
MTTF [day]	745	991

Ⅴ. 결 론

본 논문은 CPU가 메모리의 정보를 읽을 때 소프트 오류가 검출되면 CPU의 개입 없이 오류를 정정할 수 있는 자가 복구 오류 검출 및 정정 회로에 의해 보호되는 메모리 시스템에 적용할 수 있는 최적 스크러빙 방안을 제안하였다. 제안된 방안은 메모리를 점유하는 태스크의 실행 주기를 고려하여 메모리 시스템의 신뢰도가 최대가 되도록 가용한 스크러빙 로드를 태스크 영역별로 할당하는 것이다. 비주기적인 태스크에 의해 점유되는 메모리 영역이나 주기적인 태스크에 의해 점유되지만 주기적으로 접근되지 않는 메모리 영역은 그 주기를 ∞라고 간주하고 제안된 방안을 적용할 수 있다.

참고문헌

[1] S. Karp and B.K. Gilbert, “Digital system design in the presence of single event upsets,” IEEE Trans. Aerospace and Electronic Systems, vol.29, no. 2, pp. 310-316, Apr., 1993.

[2] R. Harboe-Sorensen, E. Daly, F. Teston, H. Schweitzer, R. Nartallo, P. Perol, F. Vandenbussche, H. Dzitko, and J. Cretolle, “Observation and Analysis of Single Event Effects On-Board the SOHO Satellite,” IEEE Trans. Nuclear Science, vol. 49, no. 3, pp. 1345-1350, Jun., 2002.

[3] A. Taber and E. Normand, “Single event upset in avionics,” IEEE Trans. Nuclear Science, vol. 40, no. 2, pp. 120-126, Apr., 1993.

[4] E. Normand, “Single event upset at ground level,” IEEE Trans. Nuclear Science, vol. 43, no. 6, pp. 2742-2750, Dec., 1996.

[5] R. Morelos-Zaragoza, The Art of Error Correcting Coding, Wiley, 2002.

[6] A.M. Saleh, J.J. Serrano, and J.H. Patel, “Reliability of scrubbing recovery-techniques for RAMs,” IEEE Trans. Reliability, vol. 39, no.1, pp. 114-122, Apr., 1990.

[7] G.C. Yang, “Reliability of semiconductor RAMs with softerror scrubbing techniques,” IEE Proc. in Computers and Digital Techniques, vol.142, pp. 337-344, Sep., 1995.

[8] R.M. Goodman and M. Sayano, “The reliability of semiconductor RAM memories with on-chip error-correction coding,” IEEE Trans. Information Theory, vol.37, no. 3, pp. 884-896, May., 1991.