하드웨어 메모리 스크러버 설계

김대영*, 조창범**, 강석주***, 채태병****

Design of Hardware Memory Scrubber

Day-Young Kim*, Chang-Burm Cho**, Seok-Ju Kang***, Tae-Byung Chae****

Abstract

Usual satellite design adopts hardware Error Detection and Correction (EDAC) circuitary for memory elements to endure proper operation in space radiation environment and periodic read-back (scrubbing) scheme to remove errors occurred and to prevent further accumulation of errors, in parallel, But lack of detail radiation test data, upset rates of KOMPSAT-2 mass storage was estimated very worse compared to that of KOMPSAT-1, which was evaluated from very precise radiation test. Although upset rates were evaluated enough low to accommodate by KOMPSAT-2 Flight Software, hardware scrubbing scheme is studied to shorten scrubbing time as well. This paper describes hardware scrubbing architecture having minimum 1.88 minutes scrubbing interval over 1 Gbits memory.

초 록

대부분의 위성 설계에서 우주 방사선에 의한 메모리 데이터 보호를 위해 오류정정회로를 내장하며, 동시에 오류의 누적을 방지하기 위해 주기적으로 메모리 내용을 읽는 알고리즘을 적용하고 있다. 소프트웨어에 의한 읽기 알고리즘을 적용하는 KOMPSAT 2호기의 경우 메모리 소자에 대한 방사능 영향 시험을 수행하지 않아 1호기에 비해 다소 큰 오류 가능성이 예측되었다. 소프트웨어 알고리즘 변경으로 읽기 작업을 하도록 결정하였으나 하드웨어에 의해 더 빠른 속도로 오류를 정정하도록 하는 방법도 연구되었다. 본 논문은 이러한 연구 결과로서, 최소 1.88분 정도의 주기로 1 Gbits의 메모리 영역을 읽음으로서 하드웨어만으로 메모리 내용을 보존할 수 있는 방법에 대하여 논의하였다.

키워드 : 오류정정회로(EDAC), 메모리(memory), 스크러빙(scrubbing)

1. 서 론

지상에서 사용되는 상용 전자 장비와는 달리 높은 고도에서 동작하는 항공기나 저궤도/정지궤도 등의 우주에서 동작하는 위성에 사용되는 전자 장비들은 다소 정도의 차이는 있으나 우주

* 위성전자연구그룹/dykim@kari.re.kr
** 과학기술위성그룹/choreno@kari.re.kr
*** 한국항공산업(주)/asura@kari.re.kr
**** 위성전자연구그룹/tbchae@kari.re.kr

방사선에 의한 영향을 고려하여 설계를 하여야 한다. 더욱이 더 작은 전력을 소모하면서 복잡한 임무를 위성 자체적으로 처리하도록 하는 요구가 증가함에 따라 최신의 저전력/고집적/고성능 특성을 갖는 전자 소자 사용이 늘어나고, 이로 인해 우주 방사능으로 인한 영향도 커지고 있다.

SRAM, DRAM 등 메모리 소자의 경우 태양활동에 의해 직접적으로 유기되거나 지구 부근의 전자기장에 포획된 이온 입자와 먼 우주에서 날아오는 중성자 등에 의해 저장된 값이 변하는 현상 (Upset)이 오래 전부터 관측되었다. 특히 고집적 메모리 소자나 DRAM 소자의 경우 그 영향이 상대적으로 더 큰 것으로 관측되고 있다. 메모리 값의 변경은 소프트웨어 코드의 변경, 데이터 값의 변경 등으로 표현되어 정상적인 임무 수행에 많은 지장을 초래하고 있다. [1]

위성의 전자 장치 설계 과정에서 메모리 소자에 대한 우주 방사능 영향을 배제할 수 없으므로 오류가 발생할 경우를 대비하여 일반적으로 하드웨어로 구현된 다양한 오류 검출 및 정정 (EDAC, Error Detection & Correction) 회로를 사용한다. 또한 발생한 오류가 누적되어 수정되지 못하는 상황을 방지하기 위해 소프트웨어 알고리즘에 의해 주기적으로 메모리의 내용을 읽어 하드웨어에 의해 자동적으로 오류가 정정될 수 있도록 하고 있다.

다목적실용위성(KOMPSAT, KOrea Multi-Purpose SATellite)의 경우 빠른 소프트웨어 동작을 위해 설계된 SRAM 소자에 대해서는 4-bit 단위로 단일 오류 정정 및 다중 오류 검출 (SECDED, Single Error Correction & Double Error Detection) 회로를 적용하고 있으며, 위성의 상태 정보 및 위성 운영을 위해 필요한 Engineering 데이터 저장을 위해 DRAM 소자로 구성된 대용량 메모리(Mass Memory)에 대해서는 16-bit 단위로 SECDED 회로를 적용하고 있다. 두 종류의 메모리 모두에 대해 임무 수행 중 남는 시간에 소프트웨어에 의해 내용을 읽어 (Scrub) 오류가 누적되지 않도록 하고 있다.[2,3]

오류 누적을 방지하기 위한 소프트웨어 알고리즘 설계에 있어, 읽기 주기(Scrubbing Rate)는 매우 중요한 변수로서 메모리 소자에 대한 우주 방사선 영향 분석으로 결정된다. KOMPSAT 2호기, 특히 대용량 메모리에 대한 분석 결과 약 9분 정도의 주기가 필요한 것으로 평가되었으나 약 6일 정도의 주기로 평가된 1호기 결과와 비교해 볼 때 매우 짧은 것으로 나타났다. 이에 따라 세부적인 영향 분석이 수행되었으며, 소프트웨어 알고리즘 변경으로 영향을 흡수하도록 하였으나 더 대용량의 메모리 혹은 우주 방사능에 더 취약한 메모리 소자를 사용할 경우를 고려하여 하드웨어에 의한 대처 방안이 필요할 것으로 판단되었다.

본 논문에서는 다목적실용위성의 대용량 메모리 구조를 소개하고, 1 Giga-bit의 용량을 가진 다목적실용위성 대용량 메모리를 기준으로 소프트웨어 동작에 영향을 주지 않으면서, 하드웨어에 의해 자동적으로 읽기 동작을 수행, 현재 설계로는 약 3분이며, 개선할 경우 최소 1.88분의 읽기 주기를 갖는 Hardware Memory Scrubber 설계 결과를 소개하고자 한다.

2. 본 론

2.1 다목적실용위성 대용량 메모리 구조

다목적실용위성에 사용되는 대용량 메모리(Mass Memory)는 4Mbits DRAM을 4개씩 적층, 4Mx4-bit 구조를 갖는 소자를 12개 사용하여 구성된 512Mbit 용량의 보드 2장으로 구성되어 있다. 각각의 메모리 모듈은 그림 1에 제시된 바와 같이 독립적인 메모리 제어기(Memory Controller)와 32Mx16-bit 크기의 Memory Bank 두 개를 보유하고 있으며, 잉여성을 갖는 두 개의 CPU 보드에서 동시에 접근이 가능하도록 하고 있다.

각 Memory Bank는 6개의 4Mx4 메모리 모듈로 구성되어 있으며, 4개의 모듈은 데이터 저장 영역으로 사용되고 있으며 2개의 모듈은

EDAC 코드를 저장하도록 하고 있다. 메모리 제어기는 총 512Mbit 메모리를 32 Kword (64 Kbyte) 크기의 Page 단위로 분할한 후 각 페이지 내에서 16-bit 단위로 실제 메모리에 대한 읽기 혹은 쓰기가 가능하도록 하고 있다. 또한 (22, 16) SECDED EDAC 회로를 내장, 쓰기 과정에서 데이터 이외에 오류 정정을 위한 코드를 자동으로 기록하게 되며, 읽기 과정에서 오류를 검출한 경우 수정할 수 있다고 판단되는 경우 자동적으로 동일 영역에 재 기록하도록 하며, 수정이 불가능하다고 판단되는 경우 Double-bit Error Counter를 증가시켜 오류가 발생하였음을 표시하는 동시에 소프트웨어에서 해당 페이지를 더 이상 사용하지 않도록 하는 정보를 제공한다. 이 정보를 바탕으로 소프트웨어는 해당 페이지에 더 이상의 데이터를 기록하지 않음으로서 혹시나 있을지도 모를 데이터의 손실을 막도록 하고 있다.

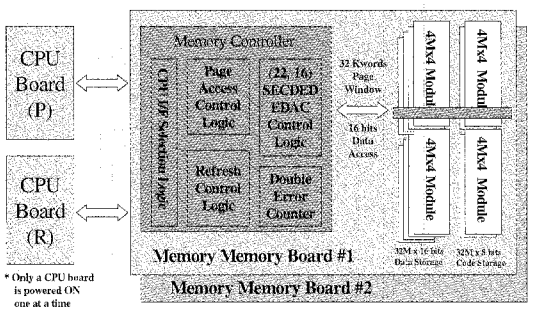

그림 1. 다목적실용위성의 대용량 메모리 구조

메모리 제어기는 DRAM Refersh를 위한 제어 회로를 내장하고 있으며, 소프트웨어가 512 Mbit의 메모리 모듈에 대한 접근을 하지 않을 경우 refresh cycle을 수행하도록 한다. KOMPSAT에 사용되는 DRAM 소자은 최소 64ms의 refresh time을 요구하고 있으나 4Mx4-bit 단위의 메모리 모듈에 대해 동시에 refresh를 하도록 하며, 모듈 내의 개별 메모리 소자에 대해 순차적으로 refersh를 하도록 함으로서 최대 43 ms 이내에 모든 메모리에 대한 refresh가 완료될 수 있도록 설계되어 있다. [4]

2.2 Hardware Scrubber 설계 타당성 분석

Memory Controller를 통해 CPU가 대용량 메모리를 사용하는 경우, read cycle일 때 최악 조건이 되며, 12MHz CPU 클럭 기준으로 최대 11clock이 소요된다. 최대로 빈번히 접근하는 경우는 Playback 동작을 수행하는 경우로, 이 경우 약 40 클럭마다 1회 read를 하는 것으로 분석되었다. 즉, 64ms 기간 중 약 17.6ms 정도만 CPU service를 위해 사용되며, 나머지 46ms 정도는 refresh를 위해 사용이 가능하므로 DRAM에 대한 refresh 동작에 문제가 없는 것으로 확인되었다. 그러나 대부분의 경우 매 초당 최대 8개의 220 바이트 데이터를 기록하며, Background Ttask로서 메모리 관리를 하므로 초당 약 3.7ms 정도의 시간, 다시 말해 64ms 기준으로는 약 0.23ms 정도의 기간 동안 CPU service를 하는 것으로 충분할 것으로 분석되었다.

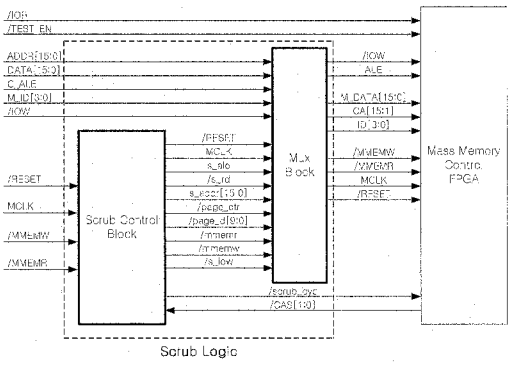

그림 2. Mass Memory Scrub Logic Block

하드웨어에 의한 Scrubber 설계는 그림 2에 제시된 바와 같이 Memory Controller(MC)와 CPU interface 사이에 Scrub Logic을 두어 CPU service가 없는 기간 동안 가상의 read-cycle을 발생시켜 읽기 동작을 수행하도록 하는 것이다. 이때 Scrub Logic은 CPU access를 계속 확인하여 CPU에 대한 service를 우선적으로 처리하도록 함으로서 정상적인 임무 수행에 지장이 없도록 하였다. 그러나 빈번한 메모리 접근으로 인해 MC에서 정상적인 refresh cycle이 이루어지

지 않을 경우를 고려, MC에서 Refresh 중임을 표시하는 CAS 제어 신호와 Scrub Logic 내의 16-bit Counter를 이용, CPU access 기간을 제외한 43 ms 이후에 가상의 scrub cycle이 동작함을 표시하도록 Scrub Cycle Enable (/scrub_cyc) 제어 신호를 전달하여 정상적인 refresh가 완료된 후에 Scrub 과정을 진행할 수 있도록 하였다.

동작의 안정성을 보장하기 위해 최악 조건하에서 최대 burst access 시간보다 긴 4ms를 여유분으로 두어 43ms는 refresh cycle이, 그리고 17ms는 가상의 scrubbing cycle이 이루어지도록 하였다. 그러나 4ms 동안 이루어지는 CPU Service 역시 전 메모리를 읽어 전송하는 S/W Scrub 과정과 같으므로 무시하여도 상관없다. 그림 3은 64ms의 refresh 주기에서 Scrub 회로가 동작할 수 있는 기간을 표시하였다.

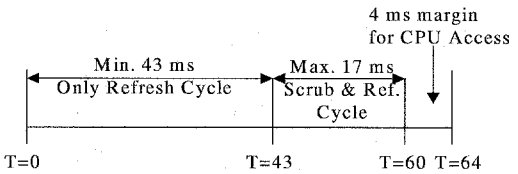

그림 3. 개념적인 Mass Memory 사용 시간 할당표

실제적인 회로에서는 11 clock의 read-cycle인 경우에도 2.5cycle 기간의 RAS/CAS address 설정 시간 동안 2cycle이 소요되는 refresh cycle이 수행될 수 있으므로 17ms 기간 중에도 refresh 동작이 이루어질 수 있다.

어떠한 경우에도 CPU access가 가능하여야 하므로 각 cycle은 다음과 같이 정의된다.

- Only Refresh Cycle : /Scrub_Cyc 신호가 Disable 상태로, CPU access 시 /CPU_MEMR 신호가 inactive 구간에서 refresh가 가능함. 최소 43 ms 동안 유지됨.
- Scrub & Refresh Cycle : .Scrub_Cyc 신호가 Enable 상태로, CPU access시 scrub 동작은 멈추고, CPU access가 끝난 후부터 다시 scrub을 시작함. 최대 17 ms 동안 유지됨. Scrub cycle은 일종의 pseudo-CPU DRAM read cycle이므로 scrub cycle중 앞쪽 2.5 cycle 시간 동안 refresh가 가능함. (CPU Access가 빈번한 경우 0 sec가 될 수도 있음.)

2.3 Hardware Scrubber 상세 설계

그림 4에는 KOMPSAT-2의 상세한 DRAM 제어 인터페이스 구성과 Hardware Scrubber를 위한 개략적인 Timing diagram을 표시하였다.

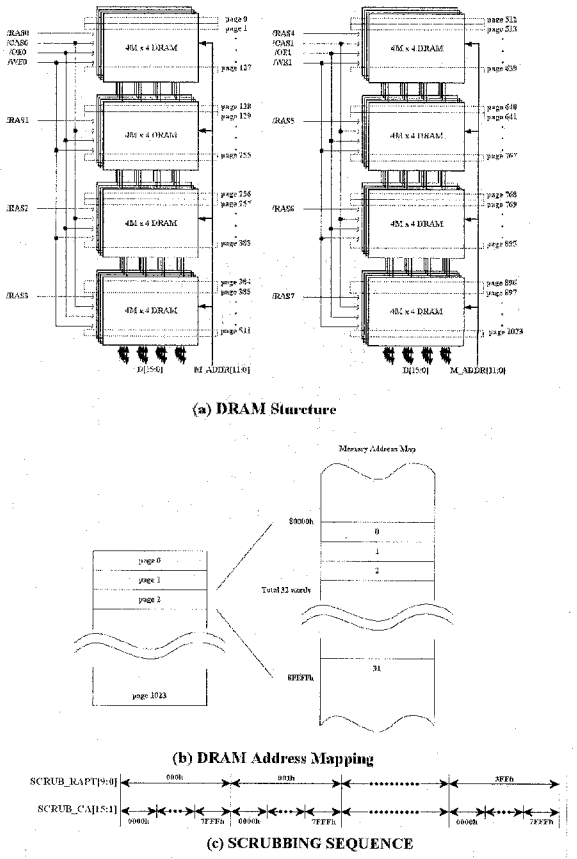

그림 4. 상세한 DRAM 제어 인터페이스 구조

그림 4에서 보는 바와 같이, 0.5Gbit DRAM modules는 총 1023 page를 가지고 있으며, 각

page는 64kbytes의 용량을 가지고 있다.

각 page는 80000h-8FFFFh의 memory address 영역에 mapping된다. Scrub의 진행은 우선 최초 'page 0'부터 시작한다. 'page 0'에 할당되어 있는 64kbytes를 우선 scrub하며, 완료되면 'page 1'에 대한 64kbytes 데이터 scrub이 이루어진다. 각 page 구별은 SCRUB_ RAPT[9:0]로 나타내며, 64kbytes 데이터 선택은 address 신호인 SCRUB_CA[15:1]에 의해 이루어진다. 즉, SCRUB _CA[15:1]가 0000h에서 7FFFFh까지 데이터를 access하면 1개page에 대한 scrub이 완료된다. 이를 다시 1024번 반복하면 전체 1024개 page에 대한 scrub이 완료되는 것이다. 이러한 동작은 그림 2에 제시한 Scrub Logic 중 Scrub Control Block에서 제어된다. Scrub Control Block에 대한 개략 구성은 그림 5와 같다. [5]

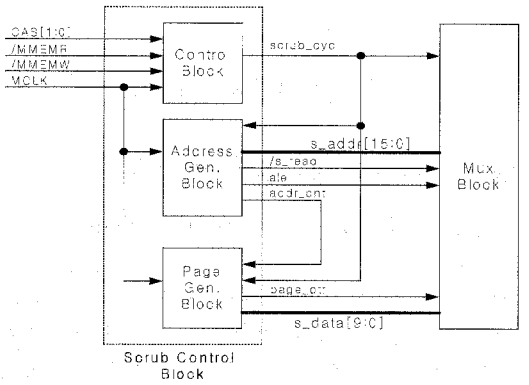

그림 5. Scrub Control Block 개념도

2.3.1 Control Logic

기존의 Mass Memory Control FPGA의 Refresh Block에서 'CAS[1:0]' 신호를 입력받아 16-bit Down Counter의 클럭으로 사용, CPU Access에 영향을 받지 않고 전 메모리에 대한 Refresh가 완료되는 43ms의 시간을 측정하며, Count값이 0이 되었을 때, Scrub Cycle이 시작될 수 있도록 'scrub_cyc' 제어신호를 발생시킨다.

또한 더 빈번한 CPU Access를 고려, 17ms보다 더 단축된 최대 10ms 동안만 Scrub Cycle이 수행되도록 제어하기 위하여 17-bit Down Counter를 사용, 절대 시간 10ms가 되면 'scrub_cyc' 신호를 Disable 시킨다. 그리고 '/MMEMR' 및 '/MMEMW' 신호를 입력받아 언제든지 CPU Access가 발생할 경우 제어 우선순위를 CPU 쪽으로 전환하며, 이 경우 scrub_cyc' 신호는 Disable 상태가 된다. 상세도를 그림 6에 제시하였다.

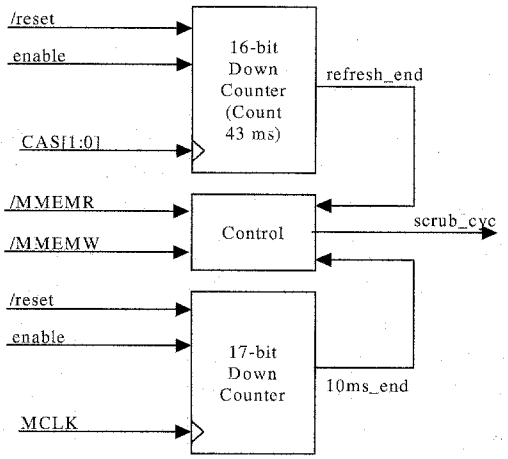

그림 6. Control Logic Block Diagram

2.3.2 Page Gen. Block

Mass Memory Control FPGA가 CPU Access 및 Refresh Cycle이 아닐 경우, 즉 Scrub Cycle에서 Mass Memory Page를 선택하기 위한 입출력 주소로서 2N10 제어신호를 발생시키고, 10-bit Counter를 사용하여 3FF까지의 Page Data를 발생시켜 Mux Block으로 전송하는 기능을 수행한다. 3FF 까지 모두 Count되었을 때, Module을 변경할 수 있도록 제어신호를 만들어 Mass Memory 내의 2048 Page를 모두 Access 할 수 있도록 제어한다. 도중에 CPU Access가 발생할 경우 현재의 Page count를 Latch 시키고, 나중에 다시 Scrub Cycle로 들어왔을 때 Latch된 Page 부터 다시 순차적으로 Page를 증가시켜 Scrub이 될 수 있도록 한다. 그림 7에 상세 블록도를 나타내었다.

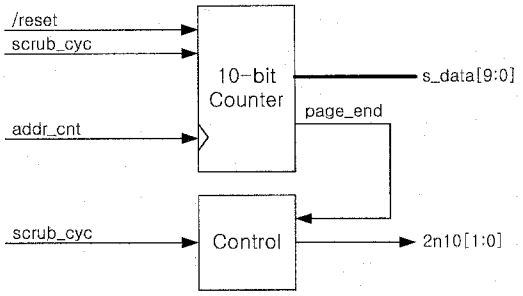

그림 7. Page Gen. Block Diagram

2.3.3 Address Gen. Block

Scrub Cycle일 경우, 16-bit Counter를 사용하여 Scrub에 필요한 Address를 발생시키고, CPU Access가 발생할 경우 현재의 Address를 Latch 시킨다. CPU Access가 끝난 후에는 Refresh 기간이 아닌 경우 Latch된 Address부터 다시 순차적으로 증가시켜 Scrub할 수 있도록 한다. 'FFFF'까지 Address Count가 끝났을 때 Page를 변경할 수 있도록 제어신호를 Page Select Block으로 전송한다. 또한 'ALE' 신호 및 '/Read' 신호를 모사하여 만들어 Mux Block으로 전송한다. 그림 8에 Address Gen. Block 상세 블럭도를 나타내었다.

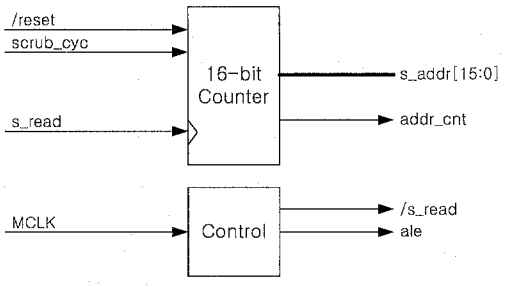

그림 8. Address Gen. Block Diagram

2.3.4 MUX Block

CPU 및 Scrub Logic Block으로부터 입력되는 Address, Data 그리고 기타 제어신호를 입력받아 'scrub_cyc' 제어신호에 따라 신호경로를 선택하여 Mass Memory Control FPGA Block으로 전송하는 기능을 수행한다. 그림 9에 상세도를 나타내었다.

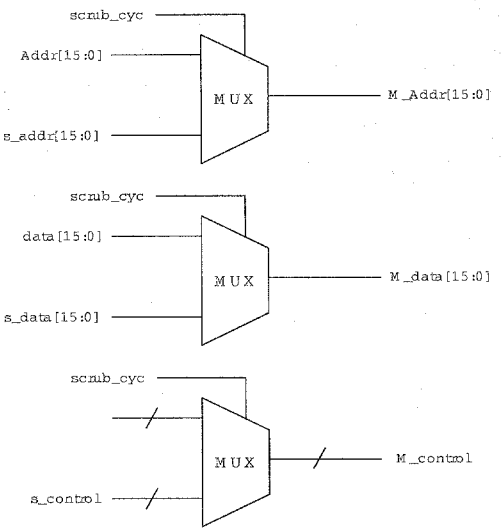

그림 9. MUX Block Diagram

2.4 Hardware Scrubber 성능 분석

설계된 Hardware Scrubber는 60ms의 시간 중 최대 10ms의 시간을 사용, 가상의 CPU cycle을 발생하여 memory 영역에 대한 read cycle이 가능하도록 하였다. 이 경우 16-bit 데이터를 읽기 위해 소요되는 기간은 11clock (916.67ns)가 소요되므로, 10ms 시간 동안 약 10,909 word에 대한 읽기 제어가 가능하다. 1초 시간동안 약 16.67번의 반복하므로 약 181,817 word(A)를 읽을 수 있다. 각 메모리 모듈은 512 Mbit 크기로서 33,554,432 word(B)이므로 한 모듈 내의 모든 영역을 읽기 위해 소요되는 시간은 (B)/(A)로서 약 184 second (약 3.07분)이 소요된다.

그러나 앞서 설명한 바와 같이 최소 64ms의 refresh interval을 60ms로 단축하였으며, 최대 21ms 정도의 CPU access 가용 시간을 17ms 혹은 10 ms로 더 축소하였다. 실제 동작 시의 CPU Access time이 매 64ms 동안 약 0.24ms

정도이며, 최대 3.7ms 정도인 것을 감안하면, 매우 많은 곳에서 여유를 갖고 있음을 알 수 있다. 64ms 주기로 여유분을 최소화한 상태에서 동작할 경우 전체 메모리를 모두 읽기 위해 필요한 시간은 최대 2.3분, 최소 1.88분이면 충분할 것으로 예측된다.

또한 대용량 메모리는 완전한 회로를 가진 512 Mbit 크기의 모듈로서 구성되므로 보드 개수를 증가시켜도 동일한 시간에 Scrub 과정을 완료할 수 있는 장점을 가지게 된다. 더 큰 용량의 메모리 소자를 사용하는 경우에도 대부분의 DRAM Refresh time이 64ms 이내이므로 동일한 결과를 가질 수 있을 것으로 판단된다.

4. 결 론

본 논문에서는 메모리에 대한 우주 방사선 영향을 완화하기 위한 방안으로 기존의 다목적 실용위성 대용량 메모리 회로에 적용할 수 있는 Hardware Scrubber 설계 결과를 제시하였다.

설계 결과, 약 3.07분 이내에 전 영역에 대한 읽기 동작이 가능함을 분석하였으며, 회로 동작을 위해 적용한 여유를 제거할 경우 최대 1.88분 이내에 동일 기능을 완료할 수 있을 것으로 평가되었다. 또한 메모리 보드의 개수를 늘리거나 더 큰 용량의 DRAM 소자를 사용, 용량을 1 Gbit 이상으로 확장할 경우에도 동일한 성능을 가질 수 있을 것으로 판단된다.

이들 이외에도 Hardware Scrubber 회로는 기존의 Memroy Controller FPGA 내에 추가적으로 회로를 구현할 수 있으므로 기존 보드 내에서 설계 변경이 가능할 것으로 판단된다.

본 설계 결과는 비록 시간적으로 안정성 검토가 충분하지 않아 다목적실용위성 2호기에는 적용되지 못하였으나 향후 국내에서의 위성 설계에 있어 소프트웨어에 대한 부담을 줄이고, 더 대용량의 메모리를 운영할 수 있도록 하여 더 복잡한 임무를 수행할 수 있는 여건을 조성할 수 있을 것으로 예상된다.

참 고 문 헌

1. Llyod Condra, PDF, "Integrated Aerospace Parts Acquisition Strategy", Boeing, Sep. 11. 2002.
2. 김대영, "K2 MM Upset Rate Estimation", KOMPSAT-2 IOC K2-02-480-004, 2002.
3. 김대영, "K2 MM Scrubbing Scheme", KOMPSAT-2 IOC K2-02-480-011, 2002.
4. KAI, "Electrical Components CDA, OBC/ECU/SAR/PCU/DDC", Nov. 29, 2001.
5. 강석주, "H/W Scrub Logic Design Last Report", 2003. 6. 10.