ECC를 적용한 DRAM의 초기화 시간 최소화 방법

Minimizing method of initial time for ECC DRAM

노종성, 김종태
Roh Jong-sung, Kim Jong-Tae

Abstract - DRAM with ECC is used widely and the size of DRAM increases. According to this, DRAM initial time, especially the time to make the whole area typical value, 0, increases. This paper introduces the method that without any additional hardware, using characteristic of DRAM and DRAM controller, minimize the memory initial time. Conservative reordering - it eliminates DRAM read time and makes write buffer used - reduces initial time to make the whole DRAM area 0, by 95.36% for DDR DRAM, 93.41% for Rambus DRAM

Key Words : ECC, DRAM, initial time, buffer

1. 장 서 론

임베디드 시스템에서 메모리는 성능의 주요한 요인으로 작용한다. 최근의 임베디드 시스템에서 쓰이는 DRAM들은 용량도 커지고 속도가 빨라지면서, 고장에 대한 염려 또한 증가하고 있다. 이에 따라 ECC를 적용한 DRAM이 다양한 시스템에서 사용하고 있다. 현재 주로 사용되는 ECC의 동작은 메모리에 값을 쓸 때 정상적인 ECC 값을 같이 쓰고, 메모리에서 값을 읽을 때 ECC 값을 같이 읽어서 정상인지 검사하여 1bit 에러이면 정정하고 2bit 이상 에러이면 인터럽트를 발생하여 별도 처리를 요청한다. ECC를 적용한 DRAM의 경우, 한번도 쓰지 않은 영역에 대해 읽는 경우 초기값에 의한 ECC 고장이 발생하기 때문에, 부팅 과정에서 전체 영역에 대해 특정 값(0)으로 쓰는 과정을 거치게 된다. 초기값을 원하는 값으로 넣어 초기 ECC 고장을 방지하는 방식(일본 특허, JP4213130)이 있지만 이는 비용상의 문제로 거의 사용되지 않는다.

2. 장 연구 동기

512MB(Mega Byte) ECC DRAM을 사용하는 한 시스템에서 부팅시간을 측정한 결과 총 20초의 시간이 걸렸다. 이 중 ECC DRAM 중 511MB를 특정 값(0)으로 쓰는 과정에서 걸리는 시간이 약 15초 걸렸다(1MB는 부팅 코드가 차지하는 공간이므로 제외한다). 임베디드 시스템에서 부팅 시간이 중요한 의미를 갖기 때문에, 부팅 시간 중 가장 큰 시간을 차지하는 이 것을 줄이는 방법을 찾기 위해 연구를 시작하였다.

3. 장 최소 시간

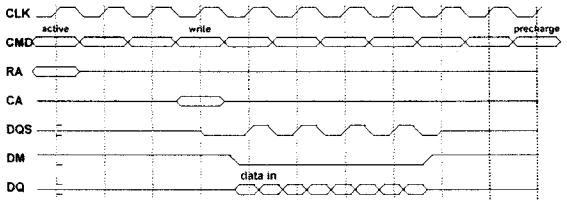
RAM의 내용을 모두 0으로 바꾸는 과정에서 가장 많은 시간이 걸리는 구간을 찾아보면 DRAM의 Bandwidth이다. 시험에 사용된 IXDP2400(IXP2400 Network Processor evaluation system)의 경우 peak bandwidth가 2.4Gbps(Giga Byte per second)이다. 단순히 511MB를 2.4Gbps로 나누면 0.2129초가 된다. 하지만 DRAM의 peak bandwidth를 가정하기 보다는 좀더 실제적인 속도를 계산하는 것이 실용적이므로 다시 계산하겠다. 사용한 DRAM은 "SAMSUNG M381L6423DTM-CB3"로써 ECC를 지원하는 DDR DRAM이다. DDR DRAM 클럭은 300MHz(150MHz X 2(DDR))를 사용하였다. 쓰기 동작만 일어난다고 가정하고 계산한다. 아래 그림 1을 참조하면, 메모리 클럭이 150MHz이므로 한 클럭당 6.67msec이고, 8burst 쓰기에 총 걸리는 시간은, 13clock, 8 burst로 쓰는 데이터 양은 8 X 64bit = 64byte이다. 따라서 Bandwidth를 계산하면,

$$\frac{64byte}{13clock \times 6.67nsec/clock} = 0.7381\,GBps$$

0.7381 GBps가 된다. 따라서 511 MB를 쓰는데 걸리는 최소 시간은 0.6923초가 된다. 읽기 동작이 일어난다면 더 많은 시간이 걸리게 된다.

최소시간 0.6923초와 실제 측정 값 15.08초를 비교하면 약 20배의 시간이 더 걸린다.

저자 소개
* 노종성 : 성균관大學 전자전기공學科 碩士課程
** 김종태 : 성균관大學 전자전기공學科 敎授 · 工博



(그림 1) 8 burst(4 byte X 8) 쓰기 - DDR DRAM

4. 장 기존 방법

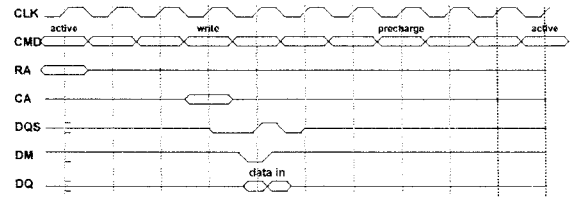
이런 차이가 발생하는 이유를 알기 위해 원인을 분석하였다.기존 코드의 초기화 순서를 간단히 나열하면 아래와 같다.

1. 리셋
2. 클럭 세팅
3. I/O 세팅
4. DRAM 제어기 초기화
5. DRAM 내용 0으로 초기화
6. 캐쉬 가능
7. MMU(Memory Management Unit)가능
8. 스택 설정
9. C 코드 실행시작

여기서 MMU 설정(7번항목) 이전에 DRAM 내용 0으로 초기화(5번항목)을 수행한다는 사실에 관심을 두었다. MMU의 설정이 끝나야 버퍼를 사용할 수 있기 때문이다. 쓰기 동작에서 버퍼를 사용할 수 없는 경우 어떤 동작이 일어나는지 분석하면 다음과 같다. 아래 그림 2을 참조하시오. 4 byte(32 bit CPU이므로)당 10클럭이 필요하게 된다. 따라서 64byte를 쓰기 위해서는 $64byte \times \frac{10clock}{4clock/byte}$, 즉 160 클럭이 걸리게 된다. 그러므로 Bandwidth를 계산하면,

$$\frac{64byte}{160clock} \times 6.67nsec/clock = 59.97Mbps$$

59.97Mbps가 된다. 따라서 511MB를 쓰는데 걸리는 최소 시간은 8.52초가 된다. 이는 실제 측정값 15.08초와 상당한 차이를 보인다. 그 이유 중 하나는 캐쉬를 사용하지 않으므로 DRAM 읽기 동작을 수행하는데 걸리는 시간이다. 이에 대한 정밀한 분석은 별도의 연구를 통해 진행할 계획이다.



(그림2) 한 번(4 byte) 쓰기 - DDR DRAM

5. 장 수정한 코드를 DDR DRAM에 적용한 경우

버퍼와 캐쉬를 사용하기 위해 기존 코드의 순서를 아래와 같이 바꾸었다.

1. 리셋
2. 클럭 세팅
3. I/O 세팅
4. DRAM 제어기 초기화
5. 캐쉬 가능
6. MMU(Memory Management Unit)가능
7. DRAM 내용 0으로 초기화
8. 스택 설정
9. C 코드 실행 시작

캐쉬 가능과 MMU 가능 동작에서 초기화되지 않은 메모리 영역을 읽지 않기 때문에 ECC 에러가 발생하지 않는다. 기대시간에서 가정한 쓰기 동작만 발생하는 상황을 만들기 위해 캐쉬 라인에 맞추어 코드를 배열하였고, 명령어 캐쉬를 가능하도록 설정하였다. 캐쉬 라인 크기와 쓰기 버퍼의 크기를 고려하여, 아래와 같이 명령을 배열하였다. 이와 같이하여 명령어를 가져오기 위한 메모리 접근을 막았고, 버퍼의 모아쓰기 기능을 최대한 사용할 수 있도록 하였다. 4개의 명령을 반복해서 사용한 이유는 분기에 의한 시간 낭비를 최소한으로 줄이기 위함이다. R0부터 R7까지 레지스터에는 모두 0이 들어있고 R8은 시작지점인 0x100000, R11은 끝나는 지점인 0x20000000이 들어있다.

```
1 :
        STMIA R8, {R0 - R7}
        STMIA R8, {R0 - R7}
        STMIA R8, {R0 - R7}
        STMIA R8, {R0 - R7}
        TST R8, R11
        BNE 1b
```

실험은 모두 4가지 상태에서 수행하였다. 데이터 캐쉬와 쓰기 버퍼를 각각 가능하게 또는 가능하지 않게 설정한 상태에서 순수하게 전체 영역을 0으로 만드는 시간을 측정하였다. 측정 방법은 50MHz 내부 카운터를 사용하였다. 내부 카운터가 사용하는 클럭은 외부에서 들어간 100MHz 클럭을 분주한 것이다. 외부에서 들어가는 100MHz 클럭은 순도가 50ppm(pulse per million, 백만번에 50번 이하의 오차 발생, 즉 1초에 99995000번에서 100005000번 뛰는 클럭이다)) 이하이므로측정 시간(유효 숫자 4자리)의 오차는 고려 대상에서 제외하였다.

실험에 사용한 4가지 상태 모두 MMU를 가능하게 한 상태에서 측정하였는데, 이것은 코드의 변경에서 오는 실수를 최소화 하면서 쓰기 버퍼가 DRAM 초기화 시간에 어떤 영향을 주는지 확인하기 위한 것이다. 각각 10회 실험하였다. 각 시험의 결과에 평균값만을 나타내었는데 이는 각 시험 결과 분산의 최대값이 9.379E-13 로써 차이가 거의 없기 때문이다.

data cache	disable	enable	disable	enable
write buffer	disable	disable	enable	enable
aver. time(sec)	15.08	0.8658	0.8658	0.8658

(표 1) DDR DRAM 511 MB를 0으로 쓰는데 걸리는 시간

표 1의 결과를 보면 캐쉬나 쓰기 버퍼 중 하나만 가능하게 하면 시간이 똑같이 걸리는 것을 알 수 있다. 이것은 모두 쓰기 버퍼(모아쓰기)가 가능하기 때문이다. ARM Architecture Reference Manual B5-8을 참조하면, 세 번째 경우는 write-through 캐쉬로 동작하면서 쓰기 버퍼를 사용하게 된다. 두 번째, 네 번째 경우는 write-back 캐쉬로 동작하면서 쓰기 버퍼를 사용하게 된다.

버퍼를 사용하는 경우 걸리는 시간은 0.8658초다. 기존에 사용하던 방식을 사용하면 걸리는 시간 15.08초에 비교하면 95.36% 감소한 시간이다. 기대시간의 최소값 0.6923초와 비교하면 아직 0.1735초(23.06%)의 과부하가 걸리는 것을 알 수 있다. 각 명령어를 수행하는데 1 CPU 클럭씩 총 6 CPU 클럭이 걸리고, 분기를 하면서 걸리는 최대 시간에 4 CPU 클럭이라고 가정하면, CPU가 600 MHz로 동작하고 128 byte를 쓸 때마다 분기가 일어나므로

$$\frac{511Mbyte}{128byte} \times 10clock \times \frac{1}{600}sec/clock = 0.06654sec$$

0.06654sec가 CPU에서 소요되는 시간이라고 할 수 있다. 이 외에 CPU에서 쓰기 버퍼까지 오고 가는 시간과, DRAM의 refresh같은 다른 동작을 고려하면 최대 성능에 상당히 근접했다고 판단된다.

6. 장 수정한 코드를 Rambus DRAM에 적용한 경우

이런 특성이 다른 DRAM에서도 적용 가능한지 판단하기 위해 Rambus DRAM에서도 같은 방법으로 측정하였다. 256 MB 크기에 1066MHz 클럭을 사용하는 MN(P)18R1624(8)EF0 Nexmod RAMBUS MODULE을 사용하였다. CPU 클럭은 700MHz이며, 시간 측정방법은 DDR과 동일하게 50MHz 내부 클럭을 사용하였다. 254 MB 크기를 0으로 쓰는 시간을 측정하였다. 사용한 시스템은 IXDP2800(IXP2800 Network Processor evaulation system)이다.

data cache	disable	enable	disable	enable
write buffer	disable	disable	enable	enable
aver. time(sec)	3.882	0.2558	0.2558	0.2558

(표. 2) Rambus DRAM 254MB를 0으로 쓰는데 걸리는 시간

측정 시간 결과를 보면 DDR DRAM과 유사하게, 93.41%가 줄어든 것을 볼 수 있다. 1.6GBps peak bandwidth를 고려하면 254MB를 0으로 쓰는데 걸리는 시간은

$$\frac{254Mbyte}{1600Mbyte/sec} = 1588sec$$

0.1588 sec이다. 여기에 다른 overhead(CPU 시간, Row Cas 지연, precharge 등)을 고려하면 이것도 상당히 최적 값에 접근했다고 판단 가능하다.

7. 장 결 론

본 논문은 ECC를 사용하는 DRAM 메모리를 초기화하는데 걸리는 시간을 최소화하기 위한 방법을 제안하였다. 최근의 memory controller에서 DRAM Bandwidth를 최대한으로 이용하여 자동으로 초기값을 써넣을 수 있도록 지원하는 경우도 있고 통상적인 Computer System에서는 점점 대중화하는 것으로 보이지만, 이 연구는 그렇지 않은 경우가 많은 임베디드 시스템과 오래된 서버에도 하드웨어 변경 없이 소프트웨어(부트로더)의 변경만으로 적용할 수 있다는 장점이 있다.

참고로 parity를 사용하는 경우에도 이 연구가 적용될 수 있으나, 통상적으로 parity를 사용하는 memory의 경우는 ECC를 사용하는 경우보다 용량이 적고 모아 쓰는 이득이 적으므로 부팅시간 전체로 보면 시간상 이득이 적다고 할 수 있다.

참 고 문 헌

[1] ARM, ARM Architecture Reference Manual, B5-8, June, 2000.

[2] Samsung, "DDR DRAM data sheet," http:///www.samsung.com/Products/Semiconductor/DRAM/DDRSDRAM/DDRSDRAMmodule/UnbufferedDIMM/M368L1624DTM/ds_m368_3811624dtm_rev11.pdf.

[3] Samsung, "Device Operation Direct RDRAM," http://www.samsung.com/Products/Semiconductor/DRAM/TechnicalInfo/2004_device_operation.pdf, 2004.

[4] Samsung, "based on 288Mb RDRAM.(E-die, 32s banks) NexModTM Module SPD Specification," http://www.samsung.com/Products/Semiconductor/DRAM/RDRAM/RDRAMmodule/NexMod/MN18R1624EF0/spd_nexmod_288mb_e_die.pdf, 2004

[5] www.ece.neu.edu/students/dmorano/classes/class041206.pdf, 2004