ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 13-02-13
80

Journal of IKEEE.Vol.17,No.2,168~175,June 2013
http://dx.doi.org/10.7471/ikeee.2013.17.2.168

# 고속 메모리의 전송선 지연시간을 적응적으로 반영하는 메모리 제어기 구조

# Memory Controller Architecture with Adaptive Interconnection Delay Estimation for High Speed Memory

이찬호*, 구교철*★

Chanho Lee*, Kyochul Koo*★

**Abstract**

The delay times due to the propagating of data on PCB depend on the shape and length of interconnection lines when memory controllers and high speed memories are soldered on the PCB. The dependency on the placement and routing on the PCB requires redesign of I/O logic or reconfiguration of the memory controller after the delay time is measured if the controller is programmable. In this paper, we propose architecture of configuring logic for the delay time estimation by writing and reading test patterns while initializing the memories. The configuration logic writes test patterns to the memory and reads them by changing timing until the correct patterns are read. The timing information is stored and the configuration logic configures the memory controller at the end of initialization. The proposed method enables easy design of systems using PCB by solving the problem of the mismatching caused by the variation of placement and routing of components including memories and memory controllers. The proposed method can be applied to high speed SRAM, DRAM, and flash memory.

**요 약**

고속의 동작 주파수를 갖는 메모리 제어기를 설계하여 PCB에서 고속 메모리와 통신을 할 경우 연결선의 길이와 배치에 따라 데이터가 전달되는 시간이 달라진다. 따라서 메모리 제어기를 설계한 뒤 PCB 상에서 메모리와 연결하여 동작시킬 때마다 이러한 지연시간이 달라져 제어기의 입출력 회로를 다시 설계하거나 초기화시 내부 설정을 바꾸어 주어야 한다. 본 논문에서는 이러한 문제를 해결하기 위해 제어기 내부에 초기화 단계에서 메모리에 테스트 패턴을 쓰고 읽으며 지연시간을 측정하고 적응적으로 지연시간을 고려한 입출력 회로를 구성하는 학습 방법을 제안한다. 제안한 학습 방법에서는 테스트 패턴을 쓰고 최소 시간 단위로 데이터를 읽는 타이밍을 바꾸어 가며 차례로 읽기를 시도하여 테스트 패턴이 정확히 읽히는 타이밍을 기억하여 초기화가 끝난 뒤 정상 동작을 시작하였을 때 학습 결과를 반영하여 메모리 접근을 시도한다. 제안한 학습 방법을 이용하면 PCB에 새로운 시스템을 구성하여도 초기화시 지연시간을 새로 설정하므로 제어기와 메모리의 통신 지연 문제를 해결할 수 있다. 제안한 방식은 고속의 SRAM, DRAM, 플래시 메모리 등에 사용 가능하다.

*Key words : Memory controller; memory; memory access timing; adaptive timing control; DRAM*

## Ⅰ. 서론

전세계적인 스마트폰과 태블릿 PC 열풍은 다양한 AP(Application Processor) 개발을 유도하였고 여러 분야의 SoC 들이 AP와 유사한 플랫폼 구조를 갖게 하였다. 이러한 SoC의 특징은 내부에 여러 개의 프로세서 또는 매스터를 가지고 있고 이들이 하나의 메모리를 공유한다는 점이다. 특히 동영상이나 음악 등의 멀티미디어 데이터를 처리하고 인식, 인증 등 많은

*School of Electronic Engineering, Soongsil University (chlee@soongsil.ac.kr, 02-820-0710)
★Dept. of Electronic Engineering, Soongsil University. (bykoo51@gmail.com, 02-825-8108)
※This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MOE) (No. 2010-0021225). The EDA tools were supported by IDEC.
Manuscript received May. 22, 2013; revised Jun. 14, 2013 ; accepted Jun 17. 2013

데이터를 고속으로 처리하는 시스템은 메모리 접근에 의해 시스템 성능이 제한되는 경우가 많다. 즉 메모리가 충분한 대역폭을 제공하지 못하면 프로세서나 하드웨어 가속기 등의 매스터에서 데이터를 기다리며 성능 저하를 경험하게 된다. 이러한 대역폭의 증가를 위해 DDR2나 DDR3 SDRAM과 같은 높은 동작 주파수를 이용하는 고속의 메모리를 사용하는 추세이다 [1-3].

DRAM은 온칩화하기 어려워 별도의 패키지로 PCB(Printed Circuit Board)상에서 메모리 제어기와 구리 등의 금속선으로 연결되거나 스마트폰에 많이 이용되는 AP(Application Processor)와 같이 멀티칩 패키지로 연결된다. PCB 상의 연결선은 칩 내부의 연결선에 비해 그 길이가 훨씬 길어 연결선을 통한 전송시간 지연을 무시할 수 없다. 멀티칩 패키지의 경우에는 PCB에 비해서는 전송시간 지연이 훨씬 작지만 온칩에 비해서는 여전히 크다. 일반 SDRAM이나 DDR(double data rate) SDRAM은 100 ~ 166MHz의 동작 주파수를 이용하므로 어느 정도의 전송 지연시간은 수용할 수 있어 PCB의 설계와 배선 변경이 일정한 범위내에서 가능하다.

그러나 DDR2 이상의 SDRAM과 같은 고속의 메모리는 200 ~ 1066MHz의 높은 동작 주파수를 갖는 클록을 사용하여 한 클록 주기가 0.938 ~ 5ns에 불과하다. 특히 클록의 한 주기가 5ns 이하로 내려가면 배선의 모양과 길이에 따라 큰 영향을 미쳐 배선을 신중하게 해야 한다. 일반적으로 고속의 메모리 제어기 설계시에는 PCB의 배선 상태에 따라 제어기의 입출력 회로에 대한 튜닝이 필요하다. 따라서 상용화된 메모리 제어기를 사용하기 위해서는 제어기 제조사에서 제공하는 표준(reference) 보드를 사용해야 동작이 보장된다. 시스템 업체에서 시스템 보드 개선이나 기능 변경을 위해서 설계 변경을 할 경우 상당한 노력과 비용을 필요로 한다. PCB 기판의 크기가 작은 휴대용 기기보다는 기판 크기가 큰 데스크탑 PC와 서버, 산업용기기와 고속의 메모리가 필요한 그래픽 어댑터 등이 이러한 문제가 더욱 심각하다.

기존의 메모리 제어기 관련 논문은 제어기 로직의 개선이나 입출력 신호의 아날로그 신호 개선에 중점을 두었고 [4-9] PCB 배선에 따른 지연시간을 제어기에서 보정하여 입출력 성능을 개선하려는 노력은 Uniquify에서 메모리 제어기의 PHY에 구현한 SCL(Self Calibrating Logic)에서 찾아볼 수 있다 [10]. 구조를 공개하지 않아 자세한 방식은 알 수 없으나 메모리 제어기의 PHY에 구현하여 제어기와 메모리 사이의 실제 지연 시간을 측정하는 것으로 보이며 그에 필요한 시간은 1us 이내이다. 상당히 정밀한 방식으로 온도와 전압의 변화에 따른 지연시간 변화까지 고려하고 있다. 그러나 이 방식은 PHY에서 구현되어 해당 메모리 제어기를 구입해야 하는 문제가 있다.

본 논문에서는 고속 메모리를 접근하기 위한 메모리 제어기에서 PCB상의 배선이 달라져도 제어기와 메모리 사이의 통신이 가능한 인터페이스 로직을 제안한다. 제안한 인터페이스 로직에서는 초기화시에 각 배선마다 일정한 패턴의 데이터를 메모리에 쓰고 이를 다시 읽는 과정에서 메모리의 사양에 따라 예상되는 기준 타이밍을 중심으로 클록의 위상을 변경시키면서 패턴이 일치하는 타이밍을 찾는다. 이 과정을 여러 번 반복하여 가장 안정적인 타이밍을 찾아 이를 기억하고 이후 실제의 메모리 접근 동작에 이를 반영한다. 이를 통해 시스템 보드 설계와 성능 개선 및 변경을 쉽게 하고 안정적인 시스템 동작을 가능하게 한다. 또한 PHY가 아닌 인터페이스 로직에 포함되므로 HDL code 형태로 구현이 가능하여 적용이 쉽다. 제안한 로직 구조에 따라 인터페이스 로직을 설계하여 DDR2 메모리 제어기에 적용하여 그 동작을 확인하였다.

## II. SDRAM의 동작

### 1. SDRAM의 구조

DRAM은 비트당 가장 저렴한 비용으로 대부분의 IT 기기의 주메모리로 이용된다. 그림 1에 나타난 바와 같이 DRAM은 제조비용을 줄이기 위해 I/O 핀 수를 줄여 접근하고자하는 실제 필요한 데이터에 접근하기 위해 2회에 걸쳐 주소지를 제공하는 방식을 사용한다. 내부의 메모리 배열은 다수의 뱅크로 되어 있고 각 뱅크는 독립적으로 제어가 가능하다. 이러한 특성들은 DRAM의 동작 속도 저하와 복잡한 제어기 구조의 원인이 되었다. 그러나 SDRAM (Synchronous DRAM)의 도입으로 메모리의 동작을 클럭과 동기화하고 DDRx(x=1,2,3,4) 방식의 도입으로 인터페이스를 수정하여 동작 주파수와 데이터 전달 속도를 개선하였다. 또한 독립적으로 제어가 가능한 뱅크 구조의 활용 등으로 성능을 개선하여 널리 이용되고 있다.

이론적인 SDRAM의 최대 대역폭, BWmax는 식 (1)과 같다.

$$BW_{\max} = \frac{f \times W_{DQ}}{8\,bit}\,(bytes/s) \qquad (1)$$

여기서 WDQ는 SDRAM의 데이터 핀의 비트 수를 의미한다.

### 2. SDRAM의 동작주파수

위에서 언급한 이유로 SDRAM의 전송 대역폭을 증가시키기 위해서는 동작주파수를 증가시키는 것이 가장 효과적이다. DDR 방식에서는 동작주파수를 증가시키지 않고 클록의 상승과 하강 에지를 동시에 사용하는 방식을 택했고 DDR2 이후는 동작주파수가 2배씩 증가한다. SDRAM의 동작주파수와 하나의 데이터를 보내거나 받는데 사용할 수 있는 시간 간격이 표1에 나타나 있다. DDR SDRAM의 경우 동작주파수는 변화가 없지만 상승 및 하강 에지를 동시에 사용하므로 하나의 데이터를 처리하는 시간은 반으로 줄어든다.

Table 1. Operation frequencies and data duration times for various SDRAMs.
표 1. SDRAM의 종류에 따른 동작주파수와 데이터 전송 간격

[단위: MHz/ns]

| SDR | DDR | DDR2 | DDR3 |
|---|---|---|---|
| 100/10 | 100/5 | 200/2.5 | 400/1.25 |
| 133/7.52 | 133/3.76 | 266/1.88 | 533/0.94 |
| 166/6.02 | 166/3.01 | 333/1.5 | 667/0.75 |
| | 200/2.5 | 400/1.25 | 800/0.63 |
| | | | 933/0.54 |
| | | | 1066/0.47 |

일반적으로 SDRAM 메모리는 별도의 패키지 형태로 메모리 제어기와는 PCB의 연결선을 통해 데이터를 주고받는다. PCB 상의 연결선은 칩내부의 연결선에 비해 캐퍼시턴스가 커서 신호전달 지연시간이 온칩 통신에 비해 크다. 메모리 제어기에서 메모리로 신호가 전달될 때는 클록과 함께 전달되므로 데이터 및 제어신호의 전달에 문제가 없지만 반대의 경우에는 제어기의 클록을 사용하여 데이터를 캡처하는데 이 데이터는 메모리 디바이스 내부의 지연시간에 PCB 연결선상의 지연시간이 포함되어 있다. 즉 그림 1에 나타난 바와 같이 (1)은 제어기의 클록이고 (1‘)은 메모리 디바이스에 도달한 클록이다. (1’)은 (1)에 비해 지연 시간 td가 발생했지만 (1)과 데이터 또는 명령어 (2)의 상대적인 관계는 (1‘)과 (2’)에서 그대로 유지된다. 그러나 읽기의 경우 기준 클록은 (1)이므로 메모리 디바이스에서 나온 데이터 (3)은 데이터 연결선에 의한 지연시간이 양방향 모두 같다고 할 때 2td 만큼의 지연시간이 더해진다. 일반적으로 메모리 제어기에서 데이터를 읽을 때 지연시간은 메모리 제어기에서 메모리로 데이터가 갈 때와 올 때의 지연시간의 합에 해당한다. 따라서 데이터를 읽을 때 메모리 내부에서 필요한 처리 시간외에 지연시간이 고려되어야 한다.

이러한 지연시간은 PCB의 배선 길이와 모양에 따라 달라진다. 칩셋 개발시 표준 보드를 만들어 공급하는 경우 제조사에서 이러한 지연시간을 고려하여 메모리 제어기를 설계하고 설정한다. 따라서 표준 보드를 그대로 시용하는 경우에는 메모리 사용에 문제가 없으나 독자적인 보드를 사용하는 경우에는 메모리에서 데이터를 읽고 쓸 때 PCB 상의 배선에 따라 문제가 발생할 수 있다. 즉 지연시간의 변화가 허용범위를 벗어나면 읽기나 쓰기 오류가 발생한다. 이러한 문제는 데이터가 유지되는 시간이 작을수록 더욱 발생하기 쉽다. 표 1에서 보인 바와 같이 DDR2나 DDR3에서 이러한 문제가 더욱 심각하다는 것을 알 수 있다. AP와 같이 같은 패키지 내부에서 SoC와 메모리가 연결된 경우에는 전송시간 지연 문제가 훨씬 완화되지만 온칩 지연시간에 비해서는 여전히 더 크므로 완전히 사라지지는 않는다.

위에서 언급한 문제가 발생하면 데이터를 읽을 때 예상보다 늦게 신호가 도착한다. 따라서 데이터를 예상시간보다 뒤에 일정 시간 간격을 두고 읽어 확인하면 예상된 값을 읽을 수 있다. 쓰기의 경우에는 읽기의 지연시간의 절반에 해당하는 영향이 있으나 클록이 같이 전달되므로 실제 메모리에 저장되는 시간은 제어기 기준으로는 지연이 발생해도 다음 명령이 동일한 지연시간 뒤에 메모리에 도달하므로 문제가 되지 않는다. 이러한 작업을 메모리 제어기에서 초기화 직후 스스로 시행하게 하면 보드 제작시 설계 및 검증 시간을 크게 줄이고 메모리 접근 동작의 안정성도 확보할 수 있다.

## III. 제안하는 메모리 제어기의 지연시간 조정 기능

### 1. SDRAM 제어기

위에서 언급한 바와 같이 메모리 제어기에서 PCB 배선에 따른 지연시간을 고려한 메모리 접근을 하면 PCB 설계에 따른 시간과 비용을 줄일 수 있다. 메모리 제어기는 시스템에 전원이 가해진 후 메모리 초기화 작업을 진행한다. 메모리 제어기의 FSM(Finite

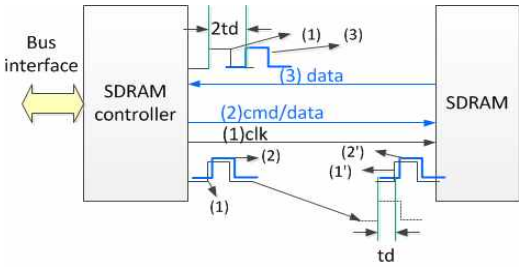



Fig. 1 Data transfer between controller and memory device
그림 1 제어기와 메모리 디바이스 사이의 데이터 전송

State Machine)을 수정하여 초기화 직후 PCB 보드의 지연시간을 예측하기 위해 일정한 패턴의 데이터를 메모리에 쓰고 읽어와 값을 확인하는 작업을 수행하게 하면서 자기 조정 기능을 수행하도록 할 수 있다. 이 때 데이터를 읽는 타이밍의 예상시간을 기준으로 일정 간격을 두고 반복해서 읽도록 하여 올바른 값이 읽히는 타이밍을 찾으면 지연시간을 예측할 수 있다. 여기서 얻은 값을 메모리 제어기의 설정(configuration) 레지스터에 저장하고 이후 매스터의 요청에 의한 메모리 접근시 이를 반영한다.

SDR SDRAM의 경우 데이터는 1 클록 사이클 동안 지속되고 제어기 클록의 상승 또는 하강 에지에서 한 번 읽힌다. 자기 조정 기능이 추가된 경우 SDRAM에 써놓은 패턴의 데이터를 정해진 수만큼 반복하여 읽어 오고 그림 2와 같이 예상되는 사이클에 제어기의 1 클록 사이클 동안 (1)의 상승 에지와 (2)의 하강 에지에서 두 번 읽고 다음은 예상 사이클보다 1 사이클 뒤에 역시 상승과 하강 에지에서 두 번을 읽는다. 이 동작을 정해진 수만큼 반복하여 올바로 읽힌 패턴을 찾는다. 예상 사이클의 상승 에지에서 값이 제대로 읽힌 경우는 PCB에 의한 지연시간을 무시할 수 있다는 의미이고 하강 에지에서 읽힌 경우는 클록 주기의 1/2 정도에 해당하는 지연시간이 발생하였으나 다음 클록의 상승 에지에서 다음 단이 데이터를 캡처할 수 있으므로 성능상의 차이는 없다. 그러나 예상 사이클의 다음 이후 사이클에서 데이터 패턴이 제대로 읽힌 경우에는 더해진 사이클 수만큼 CL(CAS Latency)이 늘어남을 의미한다.

DDRx (x=1,2,3) SDRAM의 경우에는 데이터가 제어기 전송 클록의 반 주기에 해당하는 시간동안 지속되고 한 주기에 두 개의 데이터를 읽는다. 내부적으로 90°와 270° 위상의 클록이 사용되는 것을 이용하여 그림 3의 (1) ~ (4)와 같이 네 번을 읽는다. (1) ~ (2) 중에서 올바로 읽힌 패턴이 있으면 버스 인터

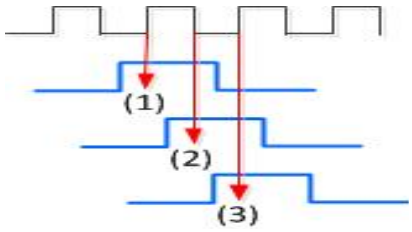



Fig. 2 Capture timing of read data for SDR SDRAM depending on the delay
그림 2 SDR SDRAM에서 지연시간에 따른 읽기 데이터의 캡처 시점

페이스에서 데이터가 전달될 때 추가 지연시간은 발생하지 않는다. 그러나 (3) 이후에 패턴이 읽히면 경우에 따라 추가적인 지연시간이 발생한다. 즉, 메모리의 동작주파수가 제어기 인터페이스의 동작주파수보다 크거나 같고 데이터 폭이 같으면 (3), (4)에서 데이터가 읽혀도 추가 지연 시간이 발생하지 않는다. 반면에 메모리의 동작주파수가 제어기 인터페이스보다 작거나 메모리의 데이터 폭이 제어기 인터페이스보다 작아 메모리에 읽거나 쓰는 데이터를 여러개 모아 인터페이스로 보내거나 받는 경우에는 (3), (4)에서 데이터가 읽히면 1 사이클의 지연시간이 발생한다. (5) 이후에 데이터가 읽히면 그에 따른 지연 시간이 항상 발생한다.

표 1에서 보듯이 DDRx의 경우에는 데이터 하나의 지속시간이 짧기 때문에 메모리 디바이스의 데이터 폭이 1 bit가 아닌 경우에는 각 bit 별로 지연시간이 달라질 수 있다. 이러한 경우에는 연속적으로 캡처한 데이터를 조합하여 패턴의 일치 여부를 확인하고 bit 별로 서로 다른 캡처 시점을 적용하여 데이터를 다시 읽어와 데이터 패턴의 일치 여부를 확인한다. 이 방법을 사용하면 캡처 로직이 복잡해지기는 하지만 불가피하게 PCB 연결선 패턴이 일정한 지연시간을 만들지 못하는 경우에도 메모리 제어기가 동작이 가능하다.

캡처 로직에서 일치되는 패턴을 찾으면 추가적인 지연시간과 캡처 타이밍을 알 수 있고 이 결과를 설정 레지스터에 저장하고 이후 정상 동작 모드에서 메모리 디바이스를 접근할 때 저장된 추가적인 지연시간을 고려하여 적절한 위상의 클록이 제어하는 레지스터에서 읽기 동작을 수행한다. 이 추가적인 지연시간은 CL이 증가한 것과 같은 효과를 나타내므로 초기화 로직을 제외하고는 기존의 제어기 구조를 크게 변경시키지 않고 제어기 설계가 가능하다. 초기화이후 정상 동작시에는 불필요한 위상의 클록이 제어하는 레지스터는 동작을 중지시킨다.

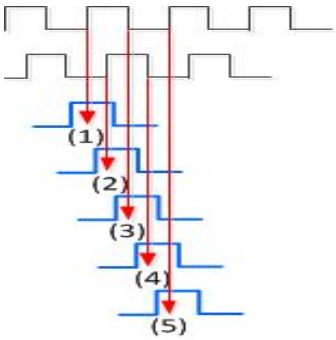



Fig. 3. Capture time of read data for DDRx SDRAMs depending on the delay
그림 3. DDRx SDRAM에서 지연시간에 따른 읽기 데이터의 캡처 시점

### 2. 적응적인 지연시간 학습기

앞에서 언급한 적응적 지연시간 학습 동작 방식에 따라 SDRAM 제어기의 메모리 디바이스와의 인터페이스 로직의 구조를 제안한다. DDRx SDRAM의 경우 기준이 되는 SDRAM 클록과 함께 90°, 180°, 270°의 위상차를 갖는 클록을 이용하여 한 사이클에 두 번씩 데이터를 읽는다. 따라서 일반적으로 두 개의 레지스터를 이용하여 상승 및 하강 에지에서 한 번씩 데이터를 읽어 저장한다. 적응적 지연시간 조정 기능이 추가되는 경우 그림 4에 나타난 것처럼 4 개의 레지스터를 초기화시에 자기 조정을 위한 패턴값을 메모리 디바이스에 쓰고 다시 읽어 저장하는 용도로 이용한다. 읽어 온 패턴값은 기준 클록에 대해 90° 위상차 간격으로 읽혀 저장된다. 0°부터 90° 간격으로 읽힌 데이터는 Reg. 1부터 Reg. 4에 차례로 저장된다. Reg. 1 또는 Reg. 2에 저장된 값이 패턴 정답과 일치하면 Reg. 3 또는 Reg. 4에는 패턴의 다음 값이 저장되므로 다음 값과의 일치 여부를 확인해야 한다. Reg.1과 Reg. 2의 값이 일치하지 않으면 패턴의 첫 번째 값을 Reg. 3 또는 Reg. 4의 값과 비교한다. 이 경우 1/2 사이클만큼의 지연시간이 추가되었음을 의미한다. 이러한 동작을 패턴의 길이만큼 반복하여 지연시간을 찾는다. 만일 일치하는 레지스터 값이 없다면 한 클록 이상의 지연이 필요하므로 en 신호를 한 사이클 늦게 보내 다시 값을 읽어 온다. 이러한 과정을 반복하여 지연시간을 찾고 일치하는 값을 가진 레지스터만을 활성화시켜 다시 한 번 패턴을 쓰고 읽어 동작을 확인한다. 이후 정상적인 읽기와 쓰기 동작에서는 4 개중 두 개의 레지스터만을 이용하여 데이터를 읽는다. SDR SDRAM인 경우에는 Reg. 1과 Reg. 3으로 학습이 가능하고 학습이 끝난 후에는 하나의 레지스터로 동작한다.

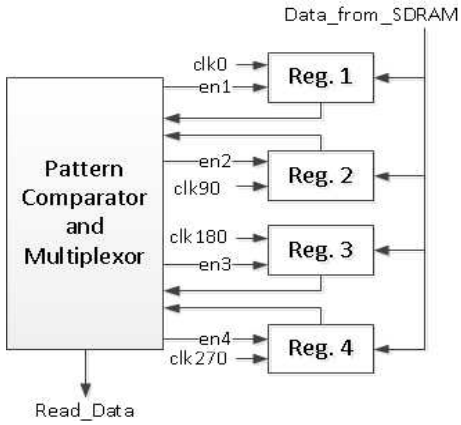



Fig. 4. Block diagram of adaptive delay learning machine
그림 4. 적응적 지연시간 학습기의 구조도

## Ⅳ. 설계 결과 및 분석

제안하는 학습기를 포함하는 DDR2 SDRAM 메모리 제어기를 Verilog-HDL을 이용하여 설계하였다. 기존의 메모리 제어기 구조에서 초기화 FSM과 입출력 인터페이스를 수정하는 것으로 제안하는 학습기를 메모리 제어기에 포함시킬 수 있다. 초기화 FSM은 기존의 초기화 과정이 끝난 뒤 테스트 패턴을 메모리 소자에 쓰고 다시 읽어 값을 비교하는 기능을 추가하고 입출력 인터페이스는 제어기에서 메모리 소자로부터 읽어 온 값을 제안한 구조의 학습기에서 처리하여 제어기와 메모리 소자 사이의 지연 시간을 추출한다.

그림 5에 제안한 방식에 따라 설계한 메모리 제어기의 구조가 나타나 있다. 앞에서 언급한 바와 같이 전체 블록다이어그램은 기존 메모리 제어기와 같고 초기화 FSM과 입출력 인터페이스의 구조만 다르다. 초기화 FSM은 메모리 소자에 대한 초기화 과정을 마친 뒤 지연시간에 대한 학습을 위한 과정이 추가된다. 이때 메모리에 대한 읽기와 쓰기 동작은 기존의 메모리 읽기/쓰기 제어기를 이용하므로 학습을 위한 추가적인 회로는 필요하지 않다. 메모리 인터페이스 로직은 그림 5(b)와 (c)에서와 같이 기존의 인터페이스 로직에 90°와 270° 위상의 클록에서 동작하는 레지스터를 추가하고 테스트 패턴을 생성하고 읽어 온 패턴을 비교하여 지연시간을 분석하여 설정 레지스터에 저장한다.

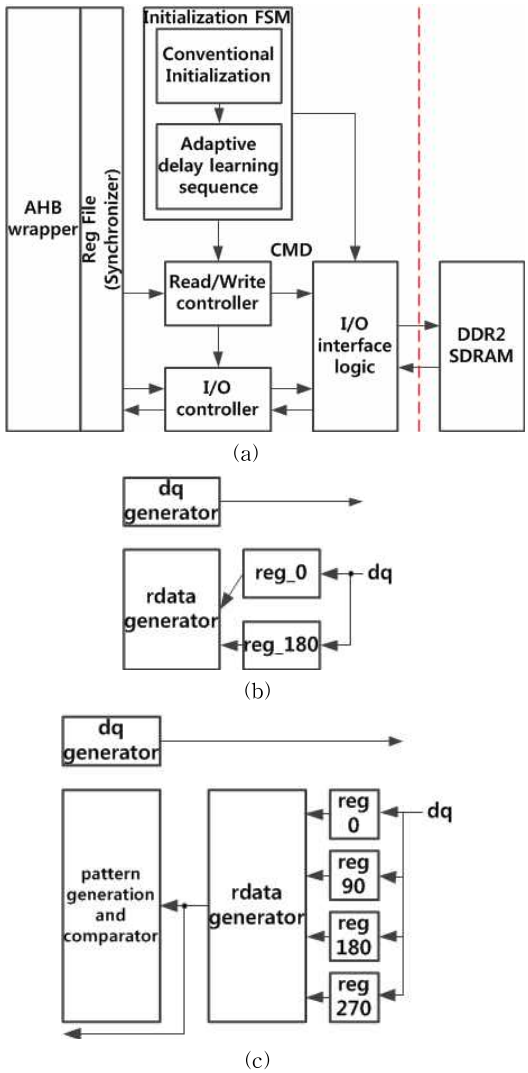



Fig. 5. DDR2 SDRAM memory controller with adaptive delay learning machine (a) Block diagram of the controller (b) Conventional memory interface logic (c) Proposed memory interface logic

그림 5. 적응적 지연시간 학습기를 갖는 DDR2 SDRAM 메모리 제어기 (a) 블록 다이어그램 (b) 기존 방식의 메모리 인터페이스 로직 구조도 (c) 제안한 방식에 따른 메모리 인터페이스 로직 구조도

설계한 메모리 제어기에 대해 다양한 지연시간을 적용하여 Micron사의 DDR2 SDRAM 모델을 이용하여 시뮬레이션을 한 결과가 그림 6에 나타나 있다. AHB 인터페이스와 메모리 제어기는 200MHz에서 동작하고 DDR2 SDRAM은 400MHz로 동작한다. 파형에서 dq는 지연시간 없이 SDRAM에서 읽어온 데이터를 의미하고 dq_delay는 전송선에 의해 지연시간이 발생하여 늦게 도착한 데이터를 나타낸다. out_HRDATA_ddr2는 메모리 제어기에서 읽기 요청을 한 매스터로 보낸 AHB 네트워크상의 데이터를 의미한다. dq_delay가 I/O 인터페이스 로직과 I/O 제어기, 그리고 AHB 인터페이스를 거쳐 AHB 네트워크에 도달하고 AHB의 동작주파수가 DDR2 동작주파수의 반이므로 데이터를 읽어온 이후 매스터에 전달되기까지는 잠복기가 필요하다.

그림 6(a)는 2ns의 전송선 지연시간을 가진 경우 기존의 제어기는 일부 데이터를 읽지 못하는 반면 제안한 제어기는 추가 잠복기 없이 데이터를 정상적으로 읽는 것을 볼 수 있다. 그림 6(b)는 5ns의 전송선 지연시간을 갖는 경우로 기존의 제어기는 데이터를 전혀 읽지 못하고 제안한 제어기는 1사이클의 추가 잠복기이후 데이터를 읽는데 AHB 버스의 HCLK은 메모리 클록의 반이므로 실제로 네트워크를 통해 데이터가 나가는 타이밍은 차이가 없는 것을 볼 수 있다.

표 2는 제안한 방식에 따라 설계한 메모리 제어기를 0.18um CMOS 라이브러리를 이용하여 합성한 결과를 기본의 제어기와 비교한 결과이다. DDR2 SDRAM의 경우 200MHz에서 제어기가 동작하면 되므로 동작주파수는 200MHz에 맞추었고 제안한 학습 로직을 추가하여 48%의 게이트 수가 증가함을 알 수 있는데 제안한 학습 로직을 설계하면서 검증을 위한 로직이 추가되고 면적에 대한 최적화가 이루어지지 않아 다소 면적 증가량이 크지만 이는 검증 로직 제거와 최적화를 통해 줄일 수 있다. 설계한 메모리 제어기는 AHB 인터페이스 기반으로 설계되어 그림 7(a)의 검증 구조를 Xilinx Virtex5(XC2V3000) FPGA와 Micron사의 DDR2 SDRAM(MT47H32M16)을 이

Table 2. Comparison results of the proposed memory controller and the conventional memory controller

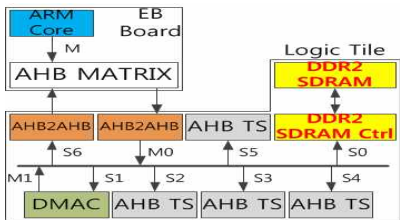
표 2. 제안한 구조에 따른 메모리 제어기와 기존 메모리 제어기의 비교

| | Proposed | Conventional |
|---|---|---|
| Operating frequency [MHz] | 200 | 200 |
| Gate count | 48,411 | 32,612 |
| PCB layout dependency | independent | dependent |

용하여 구현하고 검증하였다. 그림 7(b)에 ARM사의 RealView Debugger를 이용하여 DDR2 SDRAM에 데이터를 읽고 쓴 결과가 나타나 있다.

## Ⅴ. 결론

본 논문에서는 PCB상에서 연결선에 의한 지연시간 문제를 해결하기 위해 제어기 내부에 초기화 단계에서 메모리에 테스트 패턴을 쓰고 읽으며 전송선 지연시간을 측정하고 적응적으로 지연시간을 고려한 입출력 회로를 구성하는 학습 방법을 제안한다. 제안한 학습 방법에서는 테스트 패턴을 쓰고 최소 시간 단위로 데이터를 읽는 타이밍을 바꾸어 가며 차례로 읽기를 시도하여 테스트 패턴이 정확히 읽히는 타이밍을 기억하여 초기화가 끝난 뒤 정상 동작을 시작하였을 때 학습 결과를 반영하여 메모리 접근을 시도한다. 제안한 학습 방법을 이용하면 PCB에 새로운 시스템을 구성하여도 초기화시 지연시간을 새로 설정하므로 제어기와 메모리의 통신 지연 문제를 해결할 수 있다. 적응적 지연시간 학습기는 내부의 클록과 레지스터를 활용하여 최소한의 추가적인 로직으로 구현 가능하다. 제안한 방식에 따라 DDR2 SDRAM 메모리 제어기를 설계하여 시뮬레이션을 통해 다양한 전송선 지연시간에 대해 동작을 확인하였고 FPGA를 이용하여 구현 및 검증하였다. 제안한 방식은 PHY가 아닌 인터페이스 로직에 구현되므로 기존의 메모리 제어기에서 초기화 FSM과 입출력 로직만을 수정하여 적용 가능하고 고속의 SRAM, DRAM, 플래시 메모리 등에도 적용 가능하다.



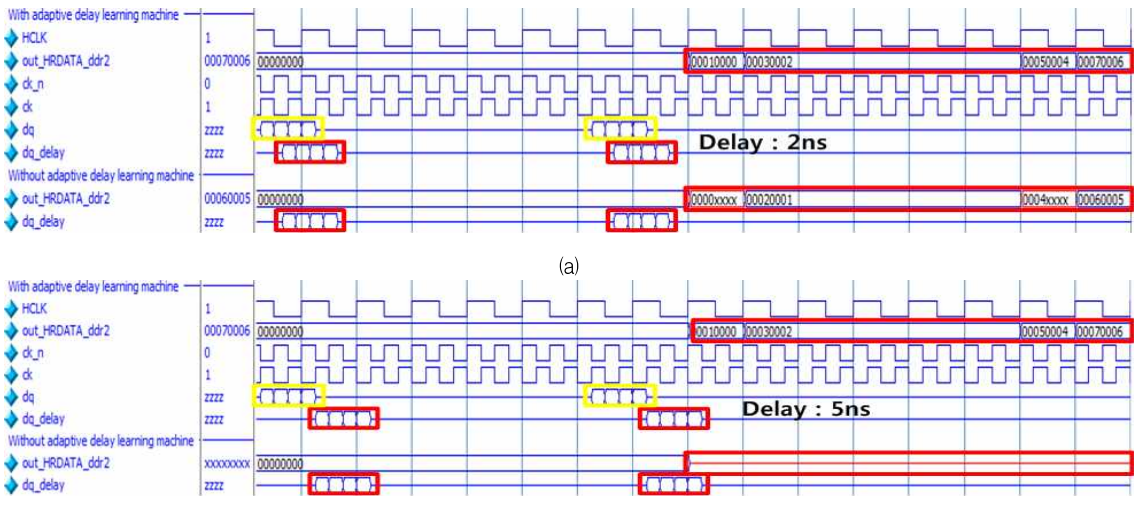

(a)

Start address 0xC0000000 Columns Auto column

| | +0 | +4 | +8 | +C |
|---|---|---|---|---|
| 0xC0000000 | 0x00000000 | 0x00001111 | 0x00002222 | 0x00003333 |
| 0xC0000020 | 0x00008888 | 0x00009999 | 0x0000AAAA | 0x0000BBBB |
| 0xC0000040 | 0x00010000 | 0x00011111 | 0x00012222 | 0x00013333 |
| 0xC0000060 | 0x00018888 | 0x00019999 | 0x0001AAAA | 0x0001BBBB |
| 0xC0000080 | 0x00020000 | 0x00021111 | 0x00022222 | 0x00023333 |
| 0xC00000A0 | 0x00028888 | 0x00029999 | 0x0002AAAA | 0x0002BBBB |
| 0xC00000C0 | 0x00030000 | 0x00031111 | 0x00032222 | 0x00033333 |
| 0xC00000E0 | 0x00038888 | 0x00039999 | 0x0003AAAA | 0x0003BBBB |

(b)

Fig. 7. Verification of the designed memory controller (a) Block diagram of verification system, (b) Read and write results using Realview debugger.
그림 7. 설계한 메모리 제어기의 검증구조 (a) 검증시스템의 블록 다이어그램 (b) Realview 디버거를 이용한 읽기/쓰기 동작 검증




(a)

(b)

Fig. 6. Simulation results of data reading for various transmission delays, td (a) td = 2ns (b) td = 5ns
그림 6. 다양한 전송선 지연시간(td)에 대한 메모리 제어기의 읽기 결과 비교 (a) td = 2ns (b) td = 5ns

## Reference

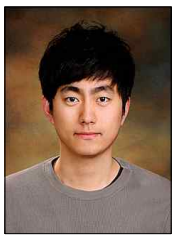
[1] Double Data Rate (DDR) SDRAM, JEDEC Standard, Feb. 2008.
[2] DDR2 SDRAM SPECIFICATION, JEDEC Standard, Nov. 2009
[3] DDR3 SDRAM SPECIFICATION, JEDEC Standard, Jul. 2010
[4] Engin ˙Ipek, Onur Mutlu, Jose F. Martınez, Rich Caruana1, "Self-Optimizing Memory Controllers: A Reinforcement Learning Approach", 35th International Symposium on Computer Architecture, 2008.(ISCA '08), pp. 39-50, June 21-25, 2008, Beijing, China
[5] Jose Carlos Sancho, Michael Lang, Darren J. Kerbyson, "Analyzing the Trade-off between Multiple Memory Controllers and Memory Channels on Multi-core Processor Performance", 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), pp. 1-7, Apr. 19-23, 2010, Atlanta, USA
[6] Vijay Gaikwad, Shashikant Lokhande, "An improved lane departure method for Advanced Driver Assistance System", 2011 International Conference on Electronics, Communications and Control (ICECC), pp. 372-375, Sep. 9-11, 2011, Ningbo, China
[7] E. Herrero, J. Gonzalez, R. Canal, D. Tullsen, "Thread Row Buffers: Improving Memory Performance Isolation and Throughput in Multiprogrammed Environments", IEEE Transactions on Computers, Early access, accepted for publishing, 2012
[8] M. Nazm Bojnordi, E. Ipek, "Programmable DDRx Controllers", IEEE Micro, Early access, accepted for publishing, 2013
[9] H.-W. Lee, H. Choi, B.-J. Shin, K.-H. Kim, K.-W. Kim, J. Kim, K.-H. Kim, J.-H. Jung, J.-H. Kim, E.-Y. Park, J.-S. Kim, J.-H. Kim, J.-H. Cho, N. Rye, J.-H. Chun, Y. Kim, C. Kim, Y.-J. Choi, B.-T. Chung, "A 1.0-ns/1.0-V Delay-Locked Loop With Racing Mode and Countered CAS Latency Controller for DRAM Interfaces," IEEE Journal of Solid-State Circuits, Vol. 47(6), pp. 1436 - 1447, 2012
[10] http://www.uniquify.com

## BIOGRAPHY

**Lee Chanho** (Member)

1987 : BS degree in Electronic Engineering, Seoul National University.
1989 : MS degree in Electronic Engineering, Seoul National University.
1994 : Ph.D degree in Electrical Engineering, University of California, Los Angeles
1994~1995 : Senior Research Engineer, Samsung Electronics. 1995~present : Professor, School of Electronic Engineering Soongsil University
<Research Area> SoC on-chip-network ,Memory controller, Image Pattern recognition, SoC design methodology, Implementation of video codec

**Koo Kyochul** (Student member)



2012 : BS degree in Electronic Engineering, Soongsil University
2012 ~present: Graduate student in Electronic
<Research Area> Memory controller