ML-AHB 버스 매트릭스를 위한 슬레이브 중심 중재 방식의 성능 분석

(Performance Analysis of Slave-Side Arbitration Schemes for the Multi-Layer AHB BusMatrix)

황 수 연 † 박 형 준 †† 장 경 선 †††
(Sooyun Hwang) (Hyeongjun Park) (Kyoungson Jhang)

요 약 온 칩 버스에서 중재 방식은 전체 시스템의 성능을 결정하는 중요한 요소 중 하나이다. 전통적인 공유 버스는 다수의 마스터와 단일 중재기 사이의 버스 사용 요청 및 권한 신호에 기반한 마스터 중심의 중재 방식을 사용한다. 마스터 중심의 중재 방식을 사용할 경우 한 순간에 오직 하나의 마스터와 슬레이브만이 데이타 전송을 수행할 수 있다. 따라서 전체 버스 시스템의 효율성 및 자원의 이용률이 감소되는 단점이 있다. 반면, 슬레이브 중심의 중재 방식은 중재기가 각 슬레이브 포트 별로 분산되며, 마스터는 중재 동작 없이 바로 트랜잭션을 시작하고, 다음 전송을 진행시키기 위해 슬레이브의 응답을 기다리는 방식을 취한다. 따라서 중재 동작의 단위가 트랜잭션 또는 단일 전송이 될 수 있다. 또한 다수의 마스터와 다수의 서로 다른 슬레이브 사이에 병렬적인 데이타 전송이 가능하기 때문에 버스 시스템의 효율성 및 자원의 이용률이 증가된다. 본 논문은 슬레이브 중심의 중재 방식을 사용하는 온 칩 버스인 ML-AHB 버스 매트릭스에 다양한 중재 방식을 적용시켜 전체 버스 시스템의 성능을 비교 분석해 보고, 어플리케이션의 특징에 따라 어떤 중재 방식을 사용하는 것이 더 유리한지에 대해 언급한다. 본 논문에서 구현한 중재 방식은 고정된 우선순위 방식, 라운드 로빈 방식 및 동적인 우선순위 방식으로 나뉘며, 마스터와 슬레이브의 특성 별로 각각 실험을 수행하였다. 성능 시뮬레이션 결과, 버스 시스템에서 임계 경로에 있는 마스터의 개수가 적을 경우 동적인 우선순위 방식이 가장 높은 성능을 보였으며, 임계 경로에 있는 마스터의 개수가 많거나, 또는 모든 마스터들의 작업 길이가 동일할 경우 라운드 로빈 방식이 가장 높은 성능을 보였다. 또한 SDRAM과 같이 접근을 위한 지연이 긴 메모리 또는 장치들을 슬레이브로 사용하는 어플리케이션에서는 단일 전송 단위의 중재 방식보다 트랜잭션 단위의 중재 방식이 더 높은 성능을 보였다. 실제 SDRAM의 지연 시간이 1, 2 및 3 클럭 사이클인 경우 각각 26%, 42% 및 51%의 성능 향상을 보였다.

키워드 : 시스템 온 칩, 온 칩 버스, 슬레이브 중심의 중재 방식, ML-AHB 버스 매트릭스

Abstract In On-Chip bus, the arbitration scheme is one of the critical factors that decide the overall system performance. The arbitration scheme used in traditional shared bus is the master-side arbitration based on the request and grant signals between multiple masters and single arbiter. In the case of the master-side arbitration, only one master and one slave can transfer the data at a time. Therefore the throughput of total bus system and the utilization of resources are decreased in the master-side arbitration. However in the slave-side arbitration, there is an arbiter at each slave port and the master just starts a transaction and waits for the slave response to proceed to the next transfer. Thus, the unit of arbitration can be a transaction or a transfer. Besides the throughput of total bus system and the utilization of resources are increased since the multiple masters can simultaneously perform transfers with independent slaves. In this paper, we implement and analyze the arbitration schemes for the Multi-Layer AHB BusMatrix based on the slave-side arbitration. We implement the slave-side arbitration schemes based on fixed priority, round robin and dynamic priority and accomplish the performance simulation to compare and analyze the performance of each arbitration

† 학생회원 : 충남대학교 컴퓨터공학과
charisma95@cnu.ac.kr
†† 비 회 원 : 한국전자통신연구원 이동통신연구단 초고속단말모뎀연구팀 팀장
parkhj@etri.re.kr
††† 종신회원 : 충남대학교 컴퓨터공학과 교수
sun@cnu.ac.kr
논문접수 : 2005년 8월 22일
심사완료 : 2007년 4월 10일

scheme according to the characteristics of the master and slave. With the performance simulation, we observed that when there are few masters on critical path in a bus system, the arbitration scheme based on dynamic priority shows the maximum performance and in other cases, the arbitration scheme based on round robin shows the highest performance. In addition, the arbitration scheme with transaction based multiplexing shows higher performance than the same arbitration scheme with single transfer based switching in an application with frequent accesses to the long latency devices or memories such as SDRAM. The improvements of the arbitration scheme with transaction based multiplexing are 26%, 42% and 51%, respectively when the latency times of SDRAM are 1, 2 and 3 clock cycles.

Key words : System on a Chip, On Chip Bus, Slave-Side Arbitration Scheme, Multi-Layer AHB BusMatrix

1. 서 론

반도체 공정 기술의 발달로 인해 프로세서, 메모리 및 기타 여러 가지 주변 장치들을 하나의 칩에 내장시키는 시스템 온 칩(System on a Chip: SoC) 설계가 가능해 졌다. 또한 점점 작아지는 TTM (Time-To-Market)을 지키고, 갈수록 짧아지는 제품들의 수명에 대처하기 위해 플랫폼 기반의 SoC 설계가 필요하다[1,2]. 플랫폼 기반의 SoC 설계는 IP(Intellectual Property) 재사용을 핵심으로 하고 있으며, 플랫폼 내에 다수의 IP들을 집적하여 다양한 기능을 수행할 수 있는 어플리케이션 개발에 중점을 두고 있다. 이와 같이 다수의 IP가 하나의 SoC 플랫폼 상에서 동작하기 위해서는 IP간의 데이타 교환, 즉 온 칩 버스 상에서의 대역폭 할당 문제가 중요한 요소로 대두된다. 특히 프로세서, DSP, 멀티미디어 IP와 같이 보다 높은 버스 대역폭을 요구하는 IP가 사용될 경우 이 문제는 더욱 심각해진다. 이와 같은 버스 대역폭 문제를 해결하기 위해 다양한 버스 구조 및 중재 방식들이 연구되고 있다[3,4].

최근 많이 사용되고 있는 온 칩 버스로는 ARM 사의 AMBA AHB 버스[5], Silicore의 Wishbone 버스[6], IBM의 CoreConnect[7] 버스가 있다. 이와 같은 온 칩 버스들은 대부분 전통적인 마스터 중심의 중재 방식을 사용한다. 마스터 중심의 중재 방식은 보통 공유 버스에서 많이 사용되는 방식으로, 마스터가 공유 버스를 통해 데이타를 전송하기 전에 미리 중재기로부터 버스 사용 권한을 받아야 버스를 사용할 수 있다[8]. 즉, 마스터와 중재기사이의 버스 사용 요청(Request) 및 버스 사용 권한(Grant) 신호에 기반한 중재 방식을 사용한다. 이와 같은 마스터 중심의 중재 방식을 사용할 경우 한 순간에 오직 하나의 마스터와 슬레이브만이 데이타 전송을 수행할 수 있기 때문에 전체 버스 시스템의 효율성 및 자원의 이용률이 감소되는 단점이 있다[8]. 반면, 슬레이브 중심의 중재 방식은 마스터 중심의 중재 방식과 같이 중재기가 중앙에 위치하여 전체 데이타 흐름을 관장하는 구조가 아닌, 각 슬레이브 포트 별로 중재기가 하나씩 위치하는 분산된 구조이다. 또한 마스터와 중재기 사이의 버스 사용 요청 및 버스 사용 권한 신호에 기반한 중재 방식을 사용하지 않고, 슬레이브 측면에서 중재기가 마스터를 선택하는 마스터 선택(Select) 신호에 기반한 중재 방식이다. 즉, 중재기는 전송 응답 신호를 이용하여 마스터를 선택한다. 따라서 다수의 마스터가 서로 다른 다수의 슬레이브로 동시에 데이타를 전송할 수 있으며, 전체 시스템의 효율성 및 자원의 이용률이 증가되는 장점이 있다. 이는 데이타 처리량이 많은 다중 마스터 및 슬레이브 시스템에 적용시키기 좋은 구조임을 의미한다. 슬레이브 중심의 중재 방식을 사용하는 온 칩 버스로는 ALTERA 사의 Avalon Switch Fabric[8,9]과 ARM 사의 ML-AHB(Multi-Layer AHB)[10] 버스 매트릭스가 있다. 특히, ML-AHB 버스 매트릭스는 저전력 임베디드 시스템에 적합한 온 칩 버스[11]로써 현재 널리 사용되고 있다. 본 논문은 슬레이브 중심 중재 방식의 성능 분석에 관한 것으로, ARM 사의 ML-AHB 버스 매트릭스에 다양한 중재 방식들을 적용시켜 성능을 비교 분석해 보고, 최근 점점 더 복잡해지는 어플리케이션의 특성 별로 어떤 중재 방식을 사용하는 것이 더 나은 성능을 보일지에 대해 언급한다.

본 논문은 다음과 같이 구성된다. 2장에서는 전통적인 공유 버스에서 사용되는 마스터 중심의 중재 방식과 최근 온 칩 버스의 성능 향상을 위해 제안되고 있는 슬레이브 중심의 중재 방식에 대해 비교 분석해 본다. 3장에서는 본 논문의 실험 대상인 ML-AHB 버스 매트릭스에 대해 간략히 소개하고, 4장에서 ML-AHB 버스 매트릭스에 적용시킨 다양한 중재 방식에 대해 설명한다. 5장에서는 구현 및 성능 시뮬레이션 결과를 보여주며, 6장에서 결론 및 향후 과제에 대해 언급한다.

2. 마스터 및 슬레이브 중심의 중재 방식

2.1 마스터 중심의 중재 방식

그림 1은 마스터 중심의 중재 방식을 사용하는 전통적인 공유 버스의 구조를 보여준다.

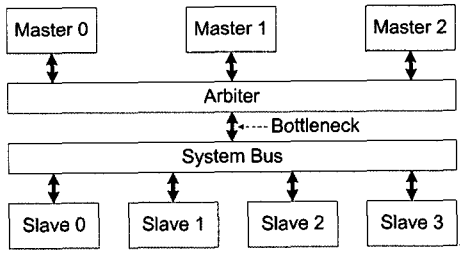

그림 1 마스터 중심의 중재 방식을 사용하는 전통적인 공유 버스 구조

전통적인 공유 버스는 그림 1과 같이 하나 또는 그 이상의 마스터들과 슬레이브들이 공유 버스를 통해 연결되며 단일 중재기가 중앙에서 버스를 제어한다. 각 마스터는 중재기로 버스 사용 요청 신호를 보내고 중재기는 고유의 중재 방식(라운드 로빈 또는 고정된 우선순위 방식)을 사용하여 하나의 마스터를 결정한 후, 해당 마스터에게 버스 사용 권한 신호를 보낸다. 만약 다수의 마스터가 동시에 버스 사용 요청 신호를 중재기로 보낸다면, 중재기에 의해 선택된 하나의 마스터를 제외하고 다른 마스터들은 대기해야 한다. 따라서 다중 마스터가 동시에 버스를 구동할 수 없게 되며 그림 1과 같이 중재기와 공유 버스 사이에 병목현상(Bottleneck)이 발생된다. 마스터 중심의 중재 방식을 사용하는 공유 버스 시스템에서는 한 순간에 하나의 마스터만이 데이타 전송이 가능하기 때문에 전체 시스템의 효율성(Throughput)이 감소되며, 한 순간에 오직 하나의 슬레이브로만 데이타 전송이 가능하기 때문에 자원(Resource)의 이용률(Utilization)이 저하된다[8].

2.2 슬레이브 중심의 중재 방식

슬레이브 중심의 중재 방식에서는 버스 라인을 공유하지 않기 때문에 마스터 중심의 중재 방식에서 발생되는 병목현상이 제거된다. 그림 2는 슬레이브 중심의 중재 방식을 사용하는 일반적인 온 칩 버스의 구조를 보여준다.

슬레이브 중심의 중재 방식에서는 그림 2와 같이 마스터와 슬레이브 사이의 모든 신호들이 1:1로 직접 연결되며, 다수의 마스터가 한 슬레이브로 동시에 데이타 전송을 요청하지 않는다면, 마스터는 슬레이브로 원하는 데이타 전송을 별도의 중재 동작으로 인한 지연 없이 바로 수행할 수 있다. 또한 다수의 마스터가 서로 다른 다수의 슬레이브로 동시에 데이타 전송을 수행할 수 있기 때문에 병렬성(Parallelism)이 증가된다. 슬레이브 중심의 중재 방식에서는 마스터 중심의 중재 방식에서와 같이 중재기가 중앙에 집중되지 않고, 각 슬레이브 포트별로 하나씩 분산되어 있다. 만약 다수의 마스터가 하나의 공유 슬레이브로 데이타 전송을 동시에 요청한다면, 슬레이브 측면에서 중재기가 고유의 중재 방식을 이용하여 마스터를 선택한다. 이때 중재 동작은 마스터와 슬레이브 사이의 버스 사용 요청 및 권한 신호가 아닌 전송 응답 신호에 기반한다. 즉, 선택된 마스터쪽으로는 슬레이브의 응답 신호가 그대로 출력되며, 선택되지 않은 마스터쪽으로는 지연 응답이 출력된다. 이 동작은 중재기에 의해 이루어진다. 따라서 모든 데이타 전송을 단일 전송 단위로 처리할 수 있게 된다. 이는 마스터들이 고정된 길이의 버스트 전송으로 데이타 전송을 수행하더라도, 슬레이브 측면의 중재기에 의해 단일 전송으로 분리될 수 있음을 의미한다. 온 칩 버스에서 이와 같은

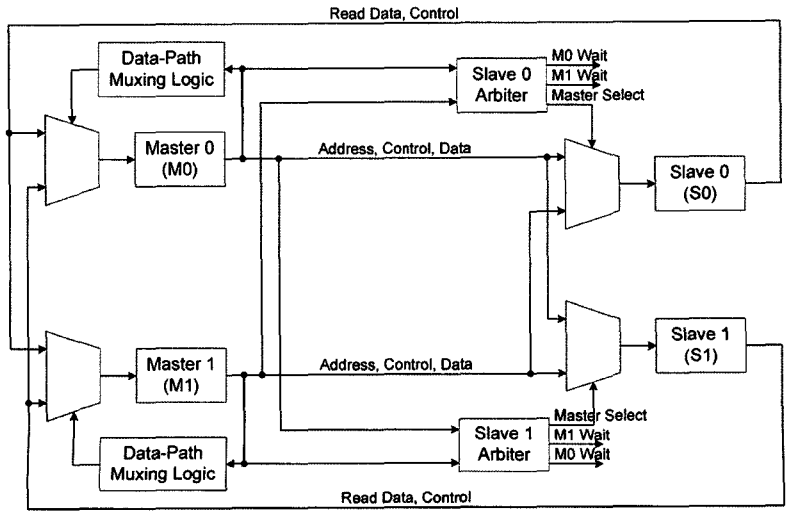

그림 2 슬레이브 중심의 중재 방식을 사용하는 일반적인 버스 구조

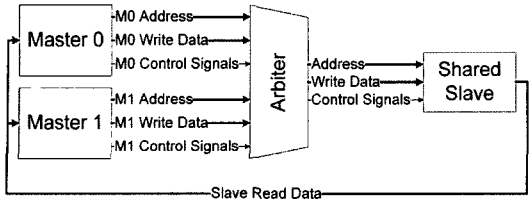

그림 3 다중 마스터 연결

특징은 융통성(Flexibility) 측면에서 유리한 조건이 될 수 있다. 그림 3은 슬레이브 중심의 중재 방식에서 마스터-슬레이브 포트 및 중재기의 연결 상태를 주소, 데이타 및 제어 신호를 중심으로 보여준다.

그림 3과 같이 중재기는 각 마스터의 주소, 데이타 및 제어 정보들 중 하나를 선택하여(Multiplexing) 공유 슬레이브 쪽으로 전송한다. 슬레이브 중심의 중재 방식을 사용하는 버스 시스템에서는 동시에 다수의 마스터와 다수의 슬레이브 사이에 데이타 전송이 가능하기 때문에, 병렬성 및 전체 시스템의 효율성이 증가하며, 다수의 슬레이브가 동시에 접근될 수 있기 때문에 자원의 이용률이 증가된다. 본 논문에서는 슬레이브 중심의 중재 방식을 사용하는 온 칩 버스인 ML-AHB 버스 매트릭스를 위한 다양한 중재 방식을 구현해 보고, 각 중재 방식의 성능을 비교 분석해 보았다.

3. ML-AHB 버스 매트릭스

ML-AHB 버스 매트릭스는 ARM 사에서 제안한 고성능 온 칩 버스 구조로써, AMBA AHB 프로토콜에 기반한 버스 매트릭스의 새로운 연결 방식이다. ML-AHB 버스 매트릭스는 시스템 내의 다중 마스터와 다중 슬레이브간의 병렬적인 접근 경로를 제공한다. 이는 한 번에 하나의 마스터만 데이타 전송이 가능했던 기존 온 칩 버스의 한계를 극복할 수 있게 해준다. ML-AHB 버스 매트릭스는 보다 복잡한 연결 매트릭스를 사용함으로써 전체 버스 대역폭을 증가시켜주고, 최근 많은 프로세서 요소들을 사용하는 휴대형 기기 및 통신 기기 등에 적합한 고성능 온 칩 버스이다. 그림 4는 ML-AHB 버스 매트릭스의 전체 구조를 보여준다.

그림 4와 같이 ML-AHB 버스 매트릭스는 입력 스테이지, 디코더 및 중재기를 포함하고 있는 출력 스테이지로 구성된다. 특히, ML-AHB 버스 매트릭스는 슬레이브 중심의 중재 방식을 사용한다. 즉, 마스터는 중재 동작 없이 바로 트랜잭션을 시작하고, 다음 전송을 진행시키기 위해 슬레이브의 응답을 기다리는 방식을 취한다. 따라서 중재기는 버스트 트랜잭션뿐만 아니라 단일 전송 단위로도 마스터를 선택할 수 있다. 하지만, ARM 사의 ML-AHB 버스 매트릭스는 단일 전송 단위의 고

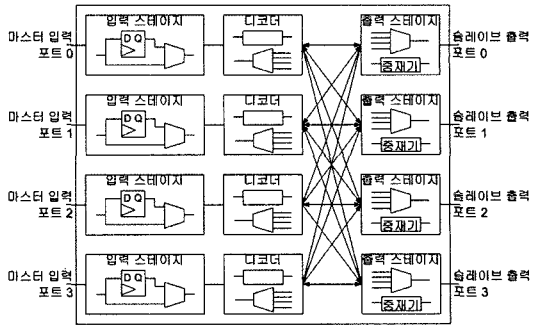

그림 4 ML-AHB 버스 매트릭스의 전체 구조[10]

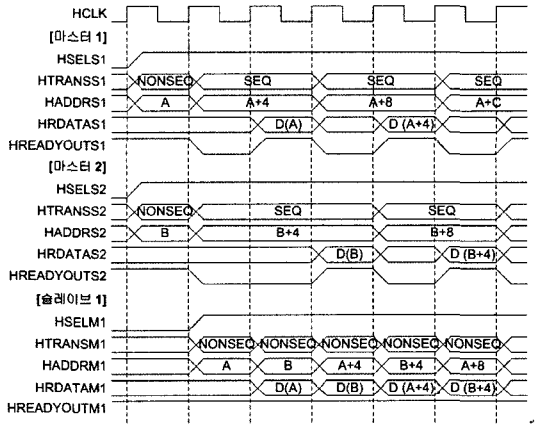

그림 5 병렬 접근에 대한 타이밍 다이어그램

정된 우선순위 방식 및 라운드 로빈 중재 방식만을 제공한다[10]. 그림 5는 공유 슬레이브 1쪽의 중재기가 마스터 1과 마스터 2를 단일 전송 단위의 라운드 로빈 방식으로 선택하는 모습을 보여준다. 그림 5에서 마스터 1과 마스터 2는 각각 자신과 연결된 입력 스테이지의 출력 신호인 슬레이브 응답 신호에 따라 다음 전송을 준비하며, 실제적인 슬레이브 응답 신호는 중재기에서 출력하는 제어 신호 및 공유 슬레이브 1에서 출력하는 응답 신호에 의해 생성된다.

4. ML-AHB 버스 매트릭스를 위한 중재 방식

본 논문에서 구현한 ML-AHB 버스 매트릭스를 위한 중재 방식은 ARM 사의 ML-AHB 버스 매트릭스에서 제공하는 중재 방식을 포함하여, 그림 6과 같이 크게 고정된 우선순위, 라운드 로빈 및 동적인 우선순위 방식으로 나뉜다.

라운드 로빈 및 동적인 우선순위 방식은 처리되는 전송 길이에 따라 각각 단일 전송, 트랜잭션 및 마스터의 전송 의지에 따른 특정 전송 길이 단위로 분류되며, 고정된 우선순위 방식은 기아현상(Starvation) 때문에 단

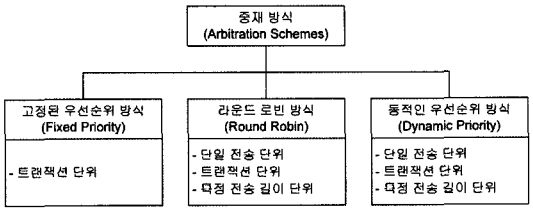

그림 6 중재 방식 분류

일 전송 단위의 중재 방식은 제외하고, 트랜잭션 단위 중재 방식만을 고려했다. 또한 동적인 우선순위 방식을 구현하기 위해 다음을 가정하였다.

- 마스터는 중재기 쪽으로 전송 요청과 함께 자신의 우선순위 레벨(Priority Level) 및 원하는 전송 길이를 알릴 수 있다.

일반적으로 시스템 개발자는 구현할 어플리케이션의 특징을 미리 알고 있다. 각 마스터의 작업량 및 작업 순서 등이 그 예이다[12]. 이와 같은 정보들을 중재기 쪽으로 동적으로 알리면 중재기는 보다 효율적으로 마스터를 선택할 수 있으며, 이는 곧 전체 시스템의 성능 향상으로 반영될 수 있다. 따라서 본 논문에서는 중재 방식의 융통성을 위해 위와 같은 사항들을 가정하였고, 그 수단으로 별도의 신호 및 선을 추가하지 않고 기존 32 비트 주소 버스의 일부를 사용하였다. 그림 7은 주소 버스의 디코딩 정보를 보여준다.

그림 7에서 최상위 4 비트는 타겟 슬레이브의 번호를, 다음 4 비트는 마스터의 우선순위 레벨을, 다음 4 비트는 원하는 전송길이를 나타내며 나머지 비트들은 슬레이브 내부의 옵셋 주소를 의미한다. 다음은 각 중재 방식에 대한 설명이다.

4.1 고정된 우선순위 방식

고정된 우선순위 방식은 대부분의 버스 시스템에서 사용하는 방식으로[13], 고정된 우선순위에 의해 마스터를 선택한다. 즉 가장 높은 우선순위를 갖는 마스터가 항상 먼저 버스를 이용할 수 있다. 따라서 상대적으로 우선순위가 낮은 마스터는 기아상태가 될 수 있다. 이 방식은 시스템 상에 마스터의 개수가 적을 경우 적합하지만, 많을 경우 심각한 기아 문제를 발생시킬 수 있다. 본 논문에서는 이와 같은 문제 때문에 트랜잭션 단위의 고정된 우선순위 방식만을 고려했다.

4.2 라운드 로빈 방식

고정된 우선순위 방식의 기아 문제를 해결하기 위해

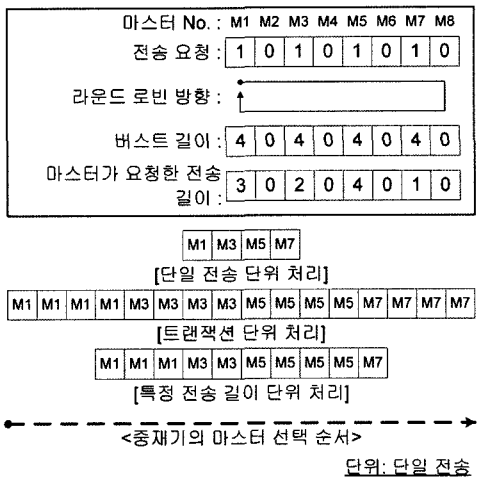

그림 8 라운드 로빈 방식의 처리 예

라운드 로빈 방식이 제안되었다[14]. 라운드 로빈 방식은 모든 마스터의 우선순위를 동일하게 유지하면서, 공평하게 버스를 사용할 수 있도록 해 주는 방식이다. 본 논문에서는 단일 전송, 트랜잭션 및 특정 전송 길이 단위의 라운드 로빈 중재 방식을 구현하였다. 그림 8은 각 중재 방식에 대한 예를 보여준다. 단일 전송(트랜잭션) 기반의 중재기는 단일 전송(버스트 트랜잭션) 단위로 데이타를 처리하며, 특정 전송 길이 기반의 중재기는 마스터가 요청한 전송 길이 단위로 데이타를 처리한다.

4.3 동적인 우선순위 방식

동적인 우선순위 방식에서 각 마스터의 우선순위는 마스터 자신의 의지에 따라 동적으로 바뀔 수 있다. 만약 어느 한 순간에 마스터들의 우선순위가 모두 동일하다면, 중재기는 기아현상을 피하기 위해 라운드 로빈 방식으로 마스터를 선택한다. 본 논문에서는 단일 전송, 트랜잭션 및 특정 전송 길이 단위의 동적인 우선순위 방식을 구현하였다. 그림 9는 각 중재 방식에 대한 예를 보여준다. 그림 9에서 마스터의 우선순위는 레벨 1이 가장 높고 레벨 4가 가장 낮다.

5. 구현 결과 및 성능 분석

5.1 구현 결과

본 논문에서는 4장에서 언급한 7 가지 중재 방식에 기반한 ML-AHB 버스 매트릭스를 각각 구현하였고, 구현된 버스 매트릭스는 중재 방식에 따라 다음과 같이

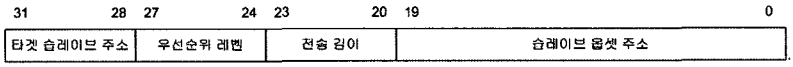

그림 7 주소 버스의 디코딩 정보

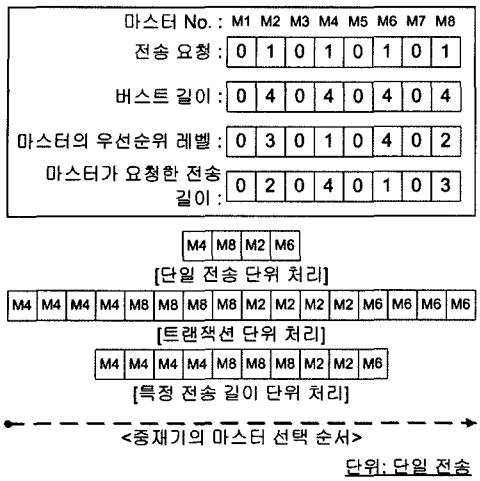

그림 9 동적인 우선순위 방식의 처리 예

분류하였다.

- 트랜잭션 단위의 고정된 우선순위 -(FR)
- 단일 전송 단위의 라운드 로빈 -(RT)
- 트랜잭션 단위의 라운드 로빈 -(RR)
- 특정 전송 길이 단위의 라운드 로빈 -(RL)
- 단일 전송 단위의 동적인 우선순위 -(DT)
- 트랜잭션 단위의 동적인 우선순위 -(DR)
- 특정 전송 길이 단위의 동적인 우선순위 -(DL)

본 논문에서 구현한 각 중재 방식이 적용된 ML-AHB 버스 매트릭스는 합성 가능한 RTL VHDL 코드로 구현되었다. 합성에 사용된 툴은 XILINX ISE 7.1i 이며 타겟 디바이스는 Virtex2P xc2vp100-6ff1704이다. 표 1과 표 2는 각 중재 방식이 적용된 ML-AHB 버스 매트릭스의 합성 결과를 슬라이스 개수 및 클럭 주기로 보여준다.

표 1 합성 결과(슬라이스 개수)

MxS	FR	RT	RR	RL	DT	DR	DL
2×2	241	233	242	263	239	245	274
4×4	712	744	724	810	795	834	981
6×6	2063	2218	2183	2446	2331	2310	2701
8×8	2964	3262	3108	3532	3360	3531	3628

(MxS: 마스터의 개수 x 슬레이브의 개수)

표 2 합성 결과(클럭 주기)

M×S	FR	RT	RR	RL	DT	DR	DL
2×2	4.4	4.5	4.4	5.8	4.4	5.2	6.4
4×4	5.9	6.3	5.9	8.8	7.9	7.4	9.6
6×6	6.7	7.8	7.8	10.2	9.0	9.5	10.9
8×8	6.8	7.9	8.0	10.9	10.5	9.6	13.0

표 1과 표 2의 합성 결과를 보면 중재기에서 고려해야 할 정보 즉, 마스터의 우선순위 레벨 및 원하는 전송 길이에 따라 면적 및 클럭 주기가 약간씩 증가하는 것을 확인할 수 있다. 전체적으로 고정된 우선순위 방식의 하드웨어 면적이 가장 작고, 마스터의 우선순위 레벨을 고려하는 동적인 우선순위 방식이 가장 크다. 또한, 라운드 로빈(동적인 우선순위) 방식에서도 마스터가 원하는 전송 길이를 고려하는 특정 전송 길이 단위의 라운드 로빈(동적인 우선순위) 방식의 하드웨어 면적이 가장 크게 나타났다.

5.2 성능 분석

ML-AHB 버스 매트릭스를 위한 슬레이브 중심의 중재 방식에 따른 버스 시스템의 성능을 분석하기 위해 Mentor 사의 Modelsim II 시뮬레이터를 사용하였다.

5.2.1 시뮬레이션 환경

그림 10은 본 논문에서 사용한 시뮬레이션 환경을 보여준다. 그림 10의 모든 컴포넌트들의 클럭 주파수는 100 MHz로 동일하며, 구현된 ML-AHB 버스 매트릭스는 32 비트의 주소 버스, 32 비트의 쓰기 데이타 버스, 32 비트의 읽기 데이타 버스, 15 비트의 제어 버스 및 3 비트의 응답 버스를 갖는다.

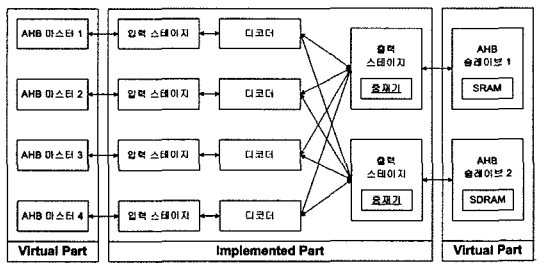

그림 10 성능 시뮬레이션을 위한 ML-AHB 버스 시스템

시뮬레이션 환경은 그림 10과 같이 크게 가상 부분(Virtual Part)과 구현 부분(Implemented Part)으로 구성된다. 구현 부분은 본 논문에서 구현한 각 중재 방식이 적용된 ML-AHB 버스 매트릭스에 해당하며, 각 버스 매트릭스는 4개의 마스터와 2개의 슬레이브로 구성된다. 가상 부분은 AHB 마스터와 AHB 슬레이브로 구성된다. AHB 마스터는 랜덤(Random) 함수에 의해 Uniform하게 트랜잭션을 발생시키며, 마스터들의 각 트랜잭션의 길이는 8-beat incrementing 버스트 타입으로 모두 동일하다. AHB 슬레이브는 AHB 마스터가 발생시킨 트랜잭션에 대해 응답한다. AHB 마스터와 AHB 슬레이브는 모두 AMBA AHB 프로토콜을 따르며, 행위 수준(Behavioral Description)의 VHDL로 모델링 되었다. 또한 시뮬레이션 중 프로토콜에 위배되는 동작이

발생될 때, 이를 검사하기 위해 VHDL과 FLI C를 이용하여 프로토콜 검사기(Protocol Checker)와 성능 모니터(Performance Monitor) 모듈을 모델링하였다.

각 ML-AHB 버스 시스템의 성능을 측정하기 전에, 적당한 워크 로드(Workload)가 결정되어야 한다. 왜냐하면 시뮬레이션 결과는 워크 로드에 의해 결정되기 때문이다. 하지만 실제 어플리케이션 수준의 워크 로드는 모든 컴포넌트들이 준비되고, 이에 따라 입력 데이타가 정확하게 모델링 되어야 하기 때문에 실제 어플리케이션 수준에서 본 논문에서 구현한 버스 매트릭스의 성능을 잘 보여줄 수 있는 워크 로드를 찾기에는 약간의 제약이 있다. 따라서 본 논문에서는 다음과 같은 파라미터들을 갖는 워크 로드 생성(Synthetic Workload Generation)에 의해 시뮬레이션을 수행하였다.

- P1: 버스 시스템에서 발생되는 전체 트랜잭션 중 각 마스터가 수행해야 하는 트랜잭션의 발생 비율
- P2: 각 마스터가 수행하는 트랜잭션 중, 마스터 내부적으로 처리해야 하는 트랜잭션의 발생 비율(각 마스터의 트랜잭션은 마스터 내부적으로 처리해야 하는 트랜잭션과 버스를 이용하여 처리해야 하는 트랜잭션으 로 나누었다. 그 이유는, 실제 어플리케이션에서 CPU나 DSP 같은 마스터들은 버스를 이용하여 각 슬레이브에 데이타를 전송할 뿐만 아니라, 마스터 내부적으로도 별도의 연산을 수행하기 때문이다. 이에 따라 보다 더 실제 어플리케이션 수준과 동일한 조건에서 시뮬레이션을 수행하기 위해 위와 같이 분류하였다.)
- P3: 각 마스터에 의해 접근되는 슬레이브의 지연 시간

위와 같은 워크로드 생성을 통해, 다양한 상황에서 시뮬레이션을 수행하였고, 이를 통해 본 논문에서 구현한 중재 방식이 적용된 ML-AHB 버스 매트릭스의 성능을 잘 보여줄 수 있는 두 가지 유용한 실험 요소를 다음과 같이 조사하였다.

(1) 마스터의 특성에 따른 실험: 마스터의 작업 길이가 성능에 미치는 영향

(2) 슬레이브의 특성에 따른 실험: 슬레이브 접근 시간이 성능에 미치는 영향

실험 (1)을 위한 타겟 슬레이브는 접근을 위해 별도의 지연 시간을 필요로 하지 않는 SRAM 타입의 AHB 슬레이브 1이며, 표 3과 표 4는 실험 (1)을 위한 시뮬레이션 파라미터 정보를 보여준다.

표 3 트랜잭션 분포: P1/P2

cases	마스터 1	마스터 2	마스터 3	마스터 4
m_case1	70%/25%	10%/0%	10%/0%	10%/0%
m_case2	40%/25%	40%/25%	10%/0%	10%/0%
m_case3	30%/25%	30%/25%	30%/25%	10%/0%
m_case4	25%/25%	25%/25%	25%/25%	25%/25%

표 4 트랜잭션 분포: P1/P2

cases	마스터 1	마스터 2	마스터 3	마스터 4
m_case5	70%/75%	10%/0%	10%/0%	10%/0%
m_case6	40%/75%	40%/75%	10%/0%	10%/0%
m_case7	30%/75%	30%/75%	30%/75%	10%/0%
m_case8	25%/75%	25%/75%	25%/75%	25%/75%

- 표 3에서, 작업 길이가 긴 임계 경로에 있는 마스터의 트랜잭션은 대부분 버스를 이용해서 처리해야 하는 트랜잭션이다.
- 표 4에서, 작업 길이가 긴 임계 경로에 있는 마스터의 트랜잭션은 대부분 마스터 내부적으로 처리해야 하는 트랜잭션이다.

실험 (1)에서 전체 트랜잭션의 개수는 4800 이다.

실험 (2)를 위한 타겟 슬레이브는 접근을 위해 별도의 지연 시간을 필요로 하는 SDRAM 타입의 AHB 슬레이브 2이며, 표 5는 실험 (2)를 위한 시뮬레이션 파라미터 정보를 보여준다.

표 5 트랜잭션 분포: P1/P2

cases	마스터 1	마스터 2	마스터 3	마스터 4
s_case1	25%/25%	25%/25%	25%/25%	25%/25%
s_case2	25%/75%	25%/75%	25%/75%	25%/75%

- s_case1에서, 마스터들의 트랜잭션은 대부분 버스를 이용하여 처리해야 하는 트랜잭션이다.
- s_case2에서, 마스터들의 트랜잭션은 대부분 마스터 내부적으로 처리해야 하는 트랜잭션이다.

실험 (2)에서 전체 트랜잭션의 개수는 2400이다.

5.2.2 시뮬레이션 결과

그림 11과 그림 12는 실험 (1)에 대한 시뮬레이션 결과를 보여준다. 성능 분석을 위해 사용된 Throughput은 다음과 같이 정의하였다.

$$Throughput = N_{transactions} * N_{transfers} * N_{bit} / T$$

위 수식에서, $N_{transactions}$은 전체 트랜잭션의 개수를, $N_{transfers}$는 트랜잭션 당 전송 길이를, N_{bit}는 전송되는 데이타 폭을, T는 전체 트랜잭션의 종료 시간을 의미한다.

그림 11과 그림 12의 결과를 통해 마스터의 작업 특성에 관계없이 임계 경로에 있는 마스터의 개수와 중재기에 의해 처리되는 데이타 단위가 전체 시스템의 성능을 결정하는 중요한 요소임을 확인할 수 있다.

임계 경로에 있는 마스터의 개수에 따른 버스 시스템의 성능 변화는 다음과 같다. m_case1및 m_case5와 같이 임계 경로에 있는 마스터가 오직 하나인 경우, DL 방식의 Throughput이 가장 높았다. 그 이유는, 임계 경로에 있는 마스터 1은 자신이 원하는 전송 길이와 함께

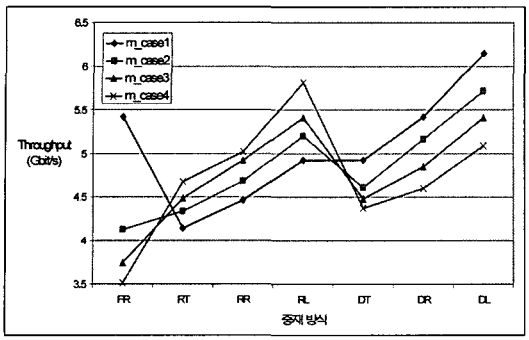

그림 11 실험 (1)에 대한 시뮬레이션 결과

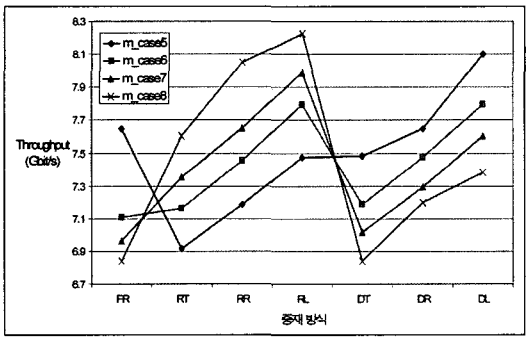

그림 12 실험 (1)에 대한 시뮬레이션 결과

가장 높은 우선순위를 중재기 쪽으로 알렸고, 중재기는 이러한 마스터 1의 전송 요구에 기반하여 데이타 전송을 처리하였다. 따라서 마스터 1은 보다 빨리 자신의 작업을 종료할 수 있게 된다. 비록 마스터 2, 마스터 3 및 마스터 4의 작업이 상대적으로 지연될지라도, 전체 버스 시스템의 트랜잭션 종료 시간은 DL 방식이 가장 짧아진다. 즉, 작업량이 많은 임계 경로에 있는 마스터의 트랜잭션을 먼저 처리해 줌으로써, 전체 트랜잭션의 종료 시간을 단축시키는 효과를 얻게 된다.

특히, m_case5의 경우, 마스터 1은 버스를 이용해서 처리해야 하는 작업을 중재기의 도움으로 빨리 끝냈기 때문에, 내부적으로 처리해야 하는 작업 또한 빨리 끝내게 되었다. DL 방식은 임계 경로에 있는 마스터가 2개인 m_case2와 m_case6에서도 가장 높은 성능을 보였다. 그 이유 또한 위와 동일하다. 즉, 버스 시스템에서 임계 경로에 있는 마스터의 개수가 적을수록 동적인 우선순위 방식이 가장 높은 성능을 보였다. 반면, m_case3, m_case4, m_case7 및 m_case8의 경우, RL 방식이 가장 높은 성능을 보였다. 이는 버스 시스템에서 임계 경로에 있는 마스터의 개수가 많거나, 또는 모든 마스터들의 작업 길이가 동일할 경우 라운드 로빈 방식이 가장 높은 성능을 보임을 의미한다. 또한 대부분의 경우, 기

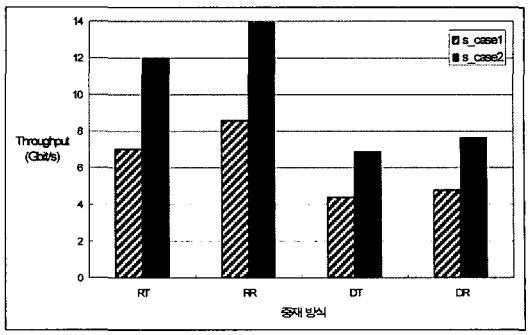

그림 13 실험 (2)에 대한 시뮬레이션 결과(SDRAM 지연 시간: 1 클럭 사이클)

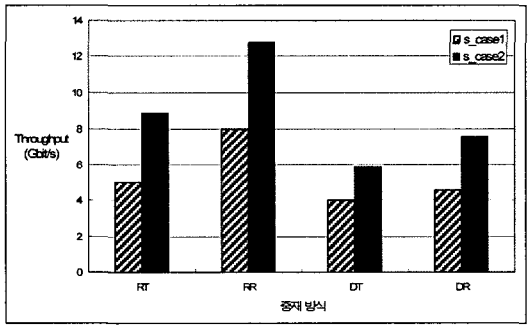

그림 14 실험 (2)에 대한 시뮬레이션 결과(SDRAM 지연 시간: 2 클럭 사이클)

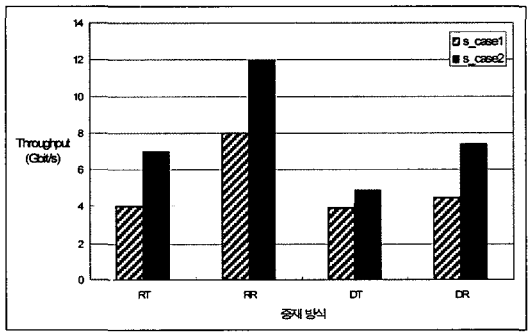

그림 15 실험 (2)에 대한 시뮬레이션 결과(SDRAM 지연 시간: 3 클럭 사이클)

아현상 때문에 고정된 우선순위 방식이 가장 낮은 성능을 보였다.

중재기에 의해 처리되는 데이타 단위에 따른 버스 시스템의 성능은 다음과 같은 순서를 갖는다.

특정 전송 길이 > 트랜잭션 > 단일 전송

즉, 마스터의 의지에 따라 처리해 주는 방식이 가장 높은 성능을 보였고, 마스터들이 전송 길이에 대한 의지를 중재기 쪽으로 알려주지 않았을 경우 일반적으로 트

랜잭션 단위로 처리해 주는 방식이 다음으로 높은 성능을 보였다. 반면 단일 전송 단위로 데이타 전송을 처리해 주는 방식은 가장 낮은 성능을 보였다.

그림 13, 그림 14 및 그림 15는 실험 (2)에 대한 시뮬레이션 결과를 보여준다. 실험 (2)의 결과를 통해, 슬레이브를 접근하기 위해 필요한 지연 시간 또한 전체 버스 시스템의 성능을 결정하는 중요한 요소임을 확인하였다. 즉, SDRAM과 같이 접근을 위해 긴 지연 시간을 필요로 하는 장치 또는 메모리를 포함하고 있는 어플리케이션에선, 트랜잭션 단위의 중재 방식이 단일 전송 단위의 중재 방식보다 더 높은 성능을 보였다. 실제 예로써 SDRAM을 접근하기 위해 필요한 지연 시간에 따라 각각 다음과 같은 성능 향상을 보였다.

- SDRAM 지연 시간이 1 클럭 사이클인 경우: 26%
- SDRAM 지연 시간이 2 클럭 사이클인 경우: 42%
- SDRAM 지연 시간이 3 클럭 사이클인 경우: 51%

6. 결론 및 향후 과제

본 논문에서는 ML-AHB 버스 매트릭스를 위한 다양한 슬레이브 중심 중재 방식을 구현하였고, 각 중재 방식의 성능을 비교 분석하였다. 본 논문에서 구현한 중재 방식은 크게 고정된 우선순위 방식, 라운드 로빈 방식 및 동적인 우선순위 방식으로 나뉘며, 중재기에 의해 처리되는 데이타 단위에 따라 각각 단일 전송, 트랜잭션 및 특정 전송 길이 단위의 중재 방식으로 분류된다. 성능 시뮬레이션 결과, 버스 시스템에서 임계 경로에 있는 마스터의 개수가 적을 경우 동적인 우선순위 방식이 가장 높은 성능을 보였으며, 임계 경로에 있는 마스터의 개수가 많거나, 또는 모든 마스터들의 작업 길이가 동일할 경우 라운드 로빈 방식이 가장 높은 성능을 보였다. 또한 SDRAM과 같이 접근을 위한 지연이 긴 메모리 또는 장치들을 슬레이브로 사용하는 어플리케이 션에서는 단일 전송 단위의 중재 방식보다 트랜잭션 단위의 중재 방식이 더 높은 성능을 보였다. 실제 SDRAM의 지연 시간이 1, 2 및 3 클럭 사이클인 경우 각각 26%, 42% 및 51%의 성능 향상을 보였다.

추후, 본 논문에서 구현한 중재 방식을 성능 분석 결과를 기반으로 실제 어플리케이션에 적용시킬 예정이다.

참 고 문 헌

[1] W.O. Cesario, D. Lyonnard, G. Nicolescu, Y. Paviot, Sungjoo Yoo, A.A. Jerraya, L. Gauthier, M. Diaz-Nava, "Multiprocessor SoC platforms: a component-based design approach," Design & Test of Computers, IEEE, Vol. 19, No. 6, pp. 52-63, Dec. 2002.

[2] Li Li, inglun Gao, uoren Cheng, uoli Zhang, huzhuan He, "A new platform-based orthogonal SoC design methodology," ASIC, 2003. Proceedings. on 5th International Conference, Vol.1, No.3, pp. 428-432, Oct.2003.

[3] Kyeong Keol Ryu, Eung Shin, V.J. Mooney, "A comparison of five different multiprocessor SoC bus architectures," in Proc. of Euromicro Symposium on Warsaw Poland, Digital Systems, Design, pp. 202-209, Sept. 2001.

[4] K. Lahiri, A. Raghunathan, G, Lakshminaray- ana, "LOTTERYBUS : A new high-performance communica tion architecture for system-on-chip designs," in Proc. Design Automation Conf. pp 15-20, 2001.

[5] "AMBA Specification," http://www.arm.com/products/solutions/AMBA_Spec.html

[6] "Wishbone," http://www.opencores.org/

[7] "The CoreConnect Bus Architecture," http://www-3.ibm.com /chips/products/coreconnect/

[8] "Avalon Switch Fabric," Quartus II 5.0 Handbook, Volume 4, Chapter 3, http://www.altera.com/literature/h b/qts/qts_qii54003.pdf

[9] "Avalon Interface Specification," http://www.altera.com /literature/manual/mnl_avalon_spec.pdf

[10] "AMBA AHB BusMatrix Specification," Document Number ARM DUI 0092C.

[11] Flynn, D, "AMBA: enabling reusable on-chip designs," Micro, IEEE, Volume 17, Issue 4, July-Aug. 1997 Page(s):20-27.

[12] Jin Ho Han et al., "Application Specific Processor Design for H.264 Decoder with a Configurable Embedded Processor," ETRI Journal, vol.27, no.5, Oct. 2005, pp.491-496.

[13] K. A. Kettler, J.P Lehoczky, and J.K. Ttrosnider, "Modeling Bus Scheduling Policies for Real-time systems," 16th IEEE Real-Time Systems Symposium, pp.242-253, 1995.

[14] Rojas-Cessa, R. "High-performance round-robin arbitration schemes for input-crosspoint buffered switches," High Performance Switching and Routing, 2004. pp.167-171.

황 수 연

2002년 한남대학교 컴퓨터공학과 졸업(학사). 2004년 충남대학교 대학원 컴퓨터공학과 졸업(석사). 2004년~현재 충남대학교 대학원 컴퓨터공학과 박사과정 2004년 9월~2006년 2월 한국전자통신연구원 IT융합/부품연구소 멀티미디어 SoC 설계팀 근무. 관심분야는 온 칩 버스 설계, 이동통신, 시스템 온 칩 설계

박 형 준

1987년 한양대학교 전자통신공학과 졸업(학사). 2001년 충남대학교 대학원 전자공학과 졸업(석사). 2006년 충남 대학교 대학원 정보통신공학과 박사 수료 1987년 2월~현재 한국전자통신 연구원 책임연구원. 1987년~1990년 전전자교환기(TDX-10) 개발. 1991년~1995년 CDMA 이동통신시스템 기지국/제어국 내부 네트워킹 장치 SoC 개발. 1996년~1998년 대용량 ATM 스위치 SoC 개발. 1999년~2001년 IMT-2000 이동통신시스템 제어국 개발. 2002년~2005년 4세대 이동통신 단말/기지국 모뎀 개발. 2006년~현재 3G Evolution 단말모뎀 개발. 관심분야는 이동통신, 시스템 온 칩 설계

장 경 선

1986년 서울대학교 전자계산기공학과 졸업(학사). 1988년 서울대학교 대학원 컴퓨터공학과 졸업(석사). 1995년 서울 대학교 대학원 컴퓨터공학과 졸업(박사) 1996년 3월~2001년 8월 한남대학교 컴퓨터공학과 교수. 2001년 9월~현재 충남대학교 컴퓨터공학과 교수. 관심분야는 컴퓨터 구조, 설계 자동화, 시스템 온 칩 설계