ML-AHB 버스 매트릭스 구현 방법의 개선

(An Improvement of Implementation Method for Multi-Layer AHB BusMatrix)

황 수 연 † 장 경 선 ††

(Soo-Yun Hwang) (Kyoung-Sun Jhang)

요 약 시스템 온 칩 설계에서 온 칩 버스는 전체 시스템의 성능을 결정하는 중요한 요소이다. 특히 프로세서, DSP 및 멀티미디어 IP와 같이 보다 높은 버스 대역폭을 요구하는 IP가 사용될 경우 온 칩 버스의 대역폭 문제는 더욱 심각해진다. 이에 따라 최근 ARM 사에서는 고성능 온 칩 버스 구조인 ML-AHB 버스 매트릭스를 제안하였다. ML-AHB 버스 매트릭스는 시스템 내의 다중 마스터와 다중 슬레이브간의 병렬적인 접근 경로를 제공하여 전체 버스 대역폭을 증가시켜주고, 최근 많은 프로세서 요소들을 사용하는 휴대형 기기 및 통신 기기 등에 적합한 고성능 온 칩 버스 구조이다. 하지만 내부 컴포넌트인 입력 스테이지와 무어 타입으로 구현된 중재 방식으로 인해 마스터가 새로운 전송을 수행할 때 또는 슬레이브 레이어를 변경할 때 마다 항상 1 클럭 사이클 지연 현상이 발생된다. 본 논문에서는 이러한 문제점을 해결하기 위해 기존 ML-AHB 버스 매트릭스 구조를 개선하였다. 기존 버스 매트릭스 구조에서 입력 스테이지를 제거하고, 개선된 구조에 적합하도록 중재 방식을 변경하여 1 클럭 사이클 지연 문제를 해결하였다. 개선된 결과 4-beat incrementing 버스트 타입으로 다수의 트랜잭션을 수행할 경우, 기존 ML-AHB 버스 매트릭스에 비해 전체 버스 트랜잭션 종료 시간 및 평균 지연 시간이 각각 약 20%, 24% 정도 짧아졌다. 또한 FPGA의 슬라이스 수는 기존의 ML-AHB 버스 매트릭스보다 약 22% 정도 감소하였고, 클럭 주기도 약 29% 정도 짧아졌다.

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

Abstract In the System on a Chip design, the on chip bus is one of the critical factors that decides the overall system performance. Especially, in the case of reusing the IPs such as processors, DSPs and multimedia IPs that requires higher bandwidth, the bandwidth problems of on chip bus are getting more serious. Recently ARM proposes the Multi-Layer AHB BusMatrix that is a highly efficient on chip bus to solve the bandwidth problems. The Multi-Layer AHB BusMatrix allows parallel access paths between multiple masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and a more flexible system architecture. However, there is one clock cycle delay for each master in existing Multi-Layer AHB BusMatrix whenever the master starts new transactions or changes the slave layers because of the Input Stage and arbitration logic realized with Moore type. In this paper, we improved the existing Multi-Layer AHB BusMatrix architecture to solve the one clock cycle delay problems and to reduce the area overhead of the Input Stage. With the elimination of the Input Stage and some restrictions on the arbitration scheme, we can take away the one clock cycle delay and reduce the area overhead. Experimental results show that the end time of total bus transaction and the average latency time of improved Multi-Layer AHB BusMatrix are improved by 20% and 24% respectively, in case of executing a number of transactions by 4-beat incrementing burst type. Besides the total area and the clock period are reduced by 22% and 29% respectively, compared with existing Multi-Layer AHB BusMatrix.

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

---

· 본 논문은 IT SoC 핵심설계인력양성 사업 및 한국전자통신연구원(ETRI)의 지원으로 수행된 결과이며, 본 논문에 사용된 CAD 툴은 IDEC으로부터 지원받았습니다.

† 학생회원 : 충남대학교 컴퓨터공학과
charisma95@cnu.ac.kr

†† 종신회원 : 충남대학교 컴퓨터공학과 교수
sun@cnu.ac.kr

논문접수 : 2005년 1월 24일
심사완료 : 2005년 7월 24일

1. 서 론

최근 시스템 온 칩 설계 기술의 발달로 하나의 칩 내에 집적되는 시스템이 다양한 기능을 수행함에 따라 높은 버스 대역폭이 요구되고 있다. 다수의 IP(Intellectual Property)가 하나의 시스템 온 칩 플랫폼 상에서 동작하기 위해서는 IP간의 데이타 교환, 즉 온 칩 버스 상에서의 대역폭 할당 문제가 중요한 요소로 대두된다. 특히 데이타 처리량이 많은 프로세서, DSP 및 멀티미디어 어플리케이션의 경우 이 문제는 더욱 심각해진다. 이와 같은 버스 대역폭 문제를 해결하기 위해 다양한 버스 구조들이 연구되고 있다[1,2].

기존 온 칩 버스는 한번에 하나의 마스터만이 데이타 전송이 가능하다. 따라서 다른 후보 마스터들의 대기 시간이 증대되며, 결국 전체 시스템의 성능이 저하되는 결과를 초래한다. 이러한 제약점을 극복하기 위해 최근 ARM 사의 AMBA 버스 (Multi-Layer AHB 버스 매트릭스)[3,4], IBM 사의 CoreConnect (PLB Crossbar Switch)[5], Silicore의 Wishbone 버스 (CONMAX)[6], Sonics Inc.의 Silicon Backplace 버스[7] 등과 같은 고성능 온 칩 버스들이 제안되었다. 특히 ARM 사의 Multi-Layer AHB (ML-AHB) 버스 매트릭스는 저전력 임베디드 시스템에 적합한 버스 구조로써 현재 널리 사용되고 있는 온 칩 버스이다. ML-AHB 버스 매트릭스는 시스템 내의 다중 마스터와 다중 슬레이브간의 병렬적인 접근 경로를 제공하여 전체 버스 대역폭을 증가시켜주고, 최근 많은 프로세서 요소들을 사용하는 휴대형 기기 및 통신 기기 등에 적합한 고성능 온 칩 버스 구조이다. 또한 ML-AHB 버스 매트릭스 내부의 입력 스테이지와 무어 타입으로 구현된 중재 방식으로 인해 동시에 다수의 마스터가 단일 슬레이브 쪽으로 일정량의 버스트 전송을 수행할 경우, 여러 마스터들이 단일 전송 사이클 단위로 데이타 전송을 공유할 수 있는 특징이 있다. 하지만 마스터가 새로운 전송을 수행할 때 또는 슬레이브 레이어를 변경할 때 마다 항상 1 클럭 사이클의 지연 현상이 발생된다. 이는 데이타 전송양이 많은 특정 어플리케이션의 경우 전체 버스 트랜잭션 종료 시간을 길게 하며 각 마스터의 평균 지연 시간을 증가시키는 결과를 초래한다. 또한 온 칩 버스를 이용하여 처리해야 할 데이타양이 많을수록 이 문제는 더욱 심각해진다. 본 논문에서는 이러한 문제점을 해결하기 위해 기존 ML-AHB 버스 매트릭스 구조를 개선하였다. 기존 버스 매트릭스 구조에서 입력 스테이지를 제거하고, 개선된 구조에 적합하도록 디코더, 출력 스테이지 및 중재 방식을 변경하여 1 클럭 사이클 지연 문제를 해결하였다. 본 논문에서 개선한 ML-AHB 버스 매트릭스에서 동시에 다수의 마스터가 단일 슬레이브 쪽으로 일정량의 버스트 전송을 수행할 경우, 기존 ML-AHB 버스 매트릭스에서와 같이 각 마스터의 버스트 전송이 단일 전송 단위로 쪼개져 수행되지 않고, 버스트 전송 단위로 차례로 수행된다. 이때, 마스터 입장에서 본다면, 기존 구조에서 발생되는 1 클럭 사이클의 지연 현상이 제거되며, 공유 슬레이브 입장에서 본다면, 별도의 버스 사이클이 낭비되지 않고 다중 마스터들의 전송이 차례로 수행된다. 따라서 기존 ML-AHB 버스 매트릭스의 주요 기능을 그대로 유지하면서, 성능 및 하드웨어 오버헤드 측면에서 기존 ML-AHB 버스 매트릭스보다 많이 향상된 결과를 실험을 통해 확인하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 기존 ML-AHB 버스 매트릭스 구조에 대해 간략히 설명하고, 3장에서 개선된 ML-AHB 버스 매트릭스 구조에 대해 소개한다. 4장에서는 기존 ML-AHB 버스 매트릭스와 개선된 ML-AHB 버스 매트릭스를 성능 및 하드웨어 오버헤드 측면에서 비교 분석해 보고, 5장에서 결론 및 향후 과제에 대해 언급한다.

2. 기존 ML-AHB 버스 매트릭스의 구조

ML-AHB 버스 매트릭스[4]는 ARM 사에서 개발한 고성능 온 칩 버스 구조로써, AMBA AHB 프로토콜에 기반한 버스 매트릭스 구조의 새로운 연결 방식이다. ML-AHB 버스 매트릭스는 시스템 내의 다중 마스터와 다중 슬레이브간의 병렬적인 접근 경로를 제공한다. 이는 한번에 하나의 마스터만 데이타 전송이 가능했던 기존 온 칩 버스의 제약점을 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 버스 구조이다. 그림 1은 ML-AHB 버스 매트릭스의 전체 구조를 보여준다.

그림 1과 같이 ML-AHB 버스 매트릭스는 입력 스테이지 (Input Stage), 디코더 (Decoder), 출력 스테이지 (Output Stage) 및 중재기 (Output Arbiter)로 구성된다. 그림 1의 각 블록들에 대한 설명은 다음과 같다.

2.1 입력 스테이지

입력 스테이지의 내부 구조는 그림 2와 같다.

그림 2의 좌측에 있는 모든 입출력 신호들은 마스터와 연결되며, 우측의 모든 입출력 신호들은 디코더 또는 출력 스테이지와 연결된다. 입력 스테이지의 주요 기능은 주소 및 제어 정보들을 잠시 저장해 두는 일이다. 즉, 타겟 공유 슬레이브 쪽으로 전송이 바로 이루어질

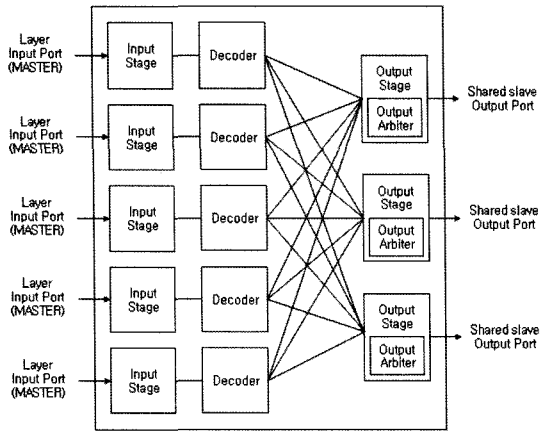

그림 1 ML-AHB 버스 매트릭스의 전체 구조[4]

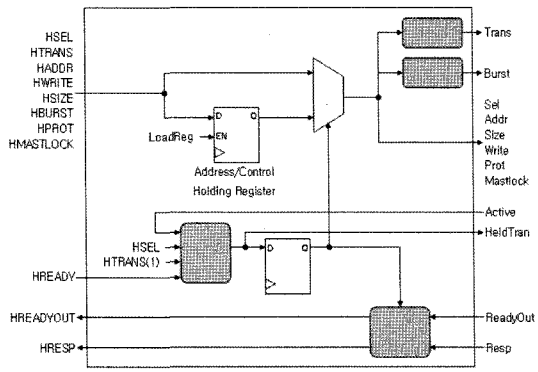

그림 2 입력 스테이지 구조[4]

수 없을 때, 마스터쪽으로부터 전송된 주소 및 제어 정보들을 잠시 저장해 두는 역할을 한다. 이 부분을 담당하는 것이 그림 2의 Address/Control Holding 레지스터이며, 인에이블 신호인 LoadReg 신호는 Active 신호에 의해 제어된다. 각 출력 스테이지는 입력 스테이지당 하나의 Active 신호를 생성하며, Active 신호는 해당 입력 스테이지로부터의 주소 및 제어정보가 현재 타겟 공유 슬레이브쪽으로 전송되고 있음을 알려준다. 입력 스테이지에서 생성되는 HeldTran 신호는 Holding 레지스터의 상태를 알려준다. Active 신호가 '0'일 경우, 요청된 주소 및 제어정보들은 Holding 레지스터에 저장되며 동시에 HeldTran 신호가 '1'로 출력된다. 이는 곧 다음 클럭 사이클부터 준비된 전송이 존재함을 의미한다.

입력 스테이지의 두 번째 주요 기능은 슬레이브의 응답 신호인 HREADYOUT 신호 및 HRESP 신호의 생성이다. HREADYOUT 신호와 HRESP 신호는 다음과 같이 생성된다.

- 전송이 적당한 출력 스테이지 쪽으로 연결되면 출력 스테이지의 HREADYOUT 신호 및 HRESP 신호가 그대로 출력된다.
- 전송 정보들이 Holding 레지스터에 저장되었을 경우, HREADYOUT 신호는 '0'을, HRESP 신호는 OKAY를 출력한다. 이는 전송이 적당한 출력 스테이지 쪽으로 연결되지 않아, 잠시 대기하는 상태를 의미한다.

2.2 디코더

그림 3은 디코더의 내부 구조를 보여준다.

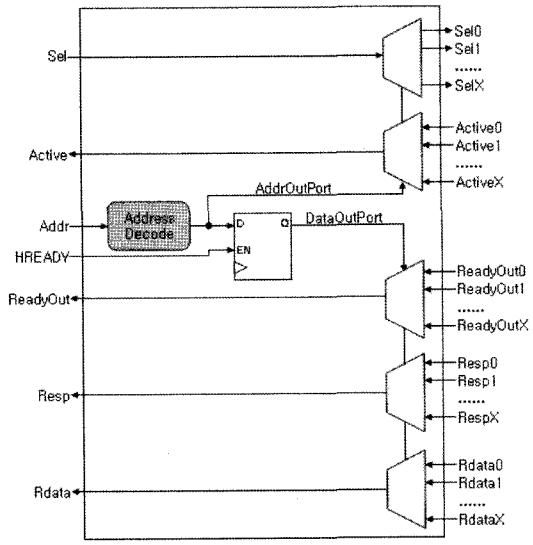

그림 3 디코더 구조[4]

각 입력 스테이지는 디코더와 연결되며, 디코더는 입력 스테이지 쪽으로부터 주소를 입력 받아 해당되는 슬레이브를 결정한다. 시스템들마다 주소 맵(Address Map) 정보가 달라질 수 있기 때문에, 그림 3의 주소 디코딩 모듈은 필요에 따라 변경이 가능하다.

디코더 내부의 AddrOutPort 신호는 주소 맵 정보에 의해 선택된 슬레이브 번호를 의미한다. 주소 구간(Address Phase)은 HREADY 신호가 '1'이 될 때 종료되고, 다음 클럭 사이클부터 데이타 구간 (Data Phase)이 시작된다. 이 과정이 그림 3에서 AddrOutPort 신호가 DataOutPort 신호로 전이되는 부분이다. DataOutPort 신호는 그림 3의 하단에 있는 3 개의 멀티플렉서에 대한 선택신호로 사용된다. 3 개의 멀티플렉서는 데이타 구간에서 동작하는 컴포넌트로써, 각 출력 스테이지로부터 ReadyOut, Resp, Rdata 신호들을 입력 받아, 자신과 연결된 입력 스테이지 쪽으로 선택된 ReadyOut, Resp, Rdata 신호를 출력한다.

2.3 출력 스테이지

그림 4는 출력 스테이지의 내부 구조를 보여준다. 그

림 4의 좌측에 있는 모든 입출력 신호들은 디코더 또는 입력 스테이지와 연결되며, 우측의 모든 입출력 신호들은 슬레이브와 연결된다. 출력 스테이지는 다음과 같은 기능을 수행한다.

- 출력 스테이지는 각각 하나의 중재기를 내장하고 있다. 중재기는 어떤 마스터를 선택할 것인지 결정하는 모듈이다.
- 출력 스테이지의 내부에 있는 멀티플렉서 및 디코더를 제어한다.

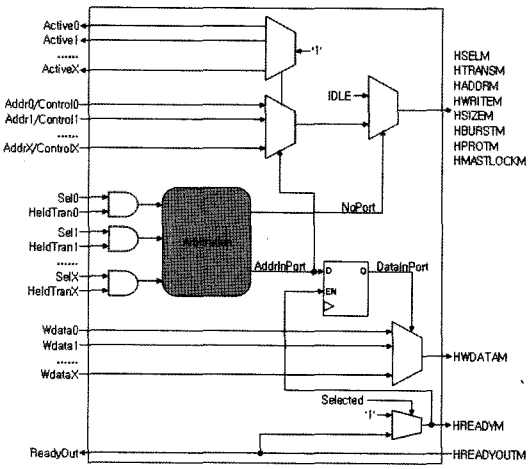

그림 4 출력 스테이지 구조[4]

출력 스테이지 내부에 있는 멀티플렉서에는 AddrInPort 신호를 선택신호로 사용하는 Address/Control 멀티플렉서와 DataInPort 신호를 선택신호로 사용하는 HWDATA 멀티플렉서가 있다. 또한 그림 4에서 Address/Control 멀티플렉서 앞단에 2×1 멀티플렉서가 하나 더 있음을 확인할 수 있다. 이 멀티플렉서는 중재기의 출력 신호인 NoPort 신호를 선택신호로 사용한다. NoPort 신호는 출력 스테이지와 연결된 공유 슬레이브 쪽으로 어떠한 전송 요청도 발생되지 않았음을 의미한다. 따라서 NoPort 신호가 '1' 일 때, 자신과 연결된 공유 슬레이브쪽으로 주소 및 제어 정보가 모두 Inactive 상태로 출력된다.

출력 스테이지의 또 다른 기능은 디코더 쪽으로 출력되는 Active 신호 및 공유 슬레이브쪽으로 출력되는 HREADY 신호의 생성이다. Active 신호는 디코더를 거쳐 최종적으로 입력 스테이지로 전송되는 신호이며, 현재 타겟 공유 슬레이브로 주소 및 제어정보들이 전송되고 있음을 의미한다. 공유 슬레이브 쪽으로 출력되는 HREADY 신호는 데이타 구간에서 해당 공유 슬레이브가 선택되었을 때, HREADYOUT 신호가 그대로 출력되며 그 외의 경우에는 항상 '1'로 유지된다.

2.4 중재기

그림 5는 중재기의 내부 구조를 보여준다.

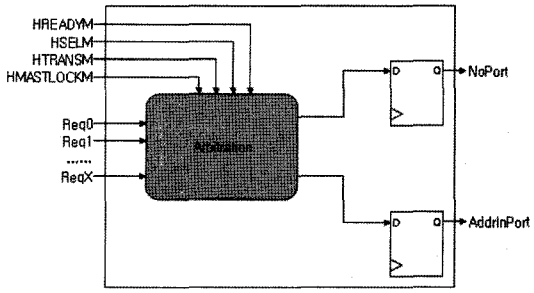

그림 5 중재기 구조[4]

중재기의 입력 신호인 Request 신호는 입력 스테이지의 출력 신호인 HeldTran 신호와 Sel 신호가 모두 '1' 일 경우에만 의미가 있다. 중재기로 모든 Request 신호들이 입력되며, 중재기는 입력된 Request 신호들을 이용하여 어떤 마스터가 다음 전송을 수행할지 결정한다. 또한 중재기의 출력 신호인 NoPort와 AddrInPort 신호는 플립플롭을 거쳐 출력된다. 즉, 무어 타입으로 구현된 중재 방식을 사용한다. 따라서 이 신호들은 다음 클럭 사이클부터 적용된다.

3. 개선된 ML-AHB 버스 매트릭스의 구조

3.1 기존 ML-AHB 버스 매트릭스의 1 클럭 사이클 지연 현상

기존 ML-AHB 버스 매트릭스는 2 장에서 설명한 입력 스테이지와 무어 타입으로 구현된 중재방식으로 인해 새로운 전송을 시작할 때 마다 그림 6과 같이 항상 1 클럭 사이클 지연 현상이 발생된다.

그림 6의 경우 외에도 마스터가 슬레이브 레이어를 변경할 때 마다 항상 1 클럭 사이클 지연 현상이 발생된다. 이는 데이타 전송양이 많은 멀티미디어 어플리케이션의 경우 전체 트랜잭션 종료 시간을 증가시키고 각 마스터의 지연 시간도 증가하게 되는 결과를 초래한다.

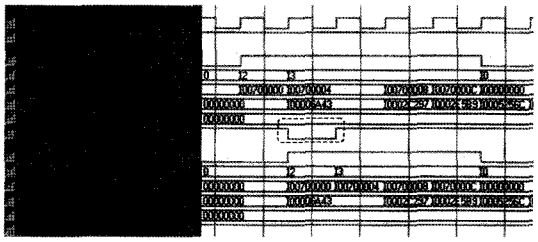

그림 6 타이밍 다이어그램 예제-1

본 논문에서는 이 문제를 해결하기 위해 기존 ML-AHB 버스 매트릭스의 구조를 개선하였다.

3.2 개선된 ML-AHB 버스 매트릭스 구조

기존 ML-AHB 버스 매트릭스에서 발생되는 1 클럭 사이클 지연 현상을 없애기 위해 본 논문에선 다음과 같이 AMBA AHB 프로토콜을 수정하였다.

- AMBA AHB 프로토콜: 마스터의 IDLE 전송에 대한 슬레이브의 응답은 항상 지연 없는 OKAY 응답이어야 하며 슬레이브는 마스터의 IDLE 전송을 무시한다.
- 수정된 프로토콜: 마스터가 새로운 전송을 시작할 때, 바로 이전 IDLE 전송에 대한 슬레이브 응답은 지연 없는 OKAY 또는 몇 사이클 지연 응답이 될 수 있다. 기본적으로 IDLE 전송에 대한 슬레이브 응답은 항상 지연 없는 OKAY 응답으로 유지되지만, 마스터가 새로운 전송을 시작할 때, 바로 이전 IDLE 전송에 대해서만 지연 응답을 발생시킬 수 있다. AMBA AHB 프로토콜에서 마스터의 IDLE 전송은 슬레이브에 대해 아무런 의미를 갖지 않기 때문에 위와 같은 수정이 가능하다.

기존 ML-AHB 버스 매트릭스에서 마스터는 첫 번째 전송을 무조건 보내고 두 번째 전송을 보내면서 첫 번째 전송에 대한 응답 신호를 보고 세 번째 전송을 준비한다. 이때 첫 번째 전송에 대한 주소 및 제어정보들은 실제적으로 슬레이브로 바로 전송되는 것이 아니라 입력 스테이지의 Holding 레지스터에 저장된다. 따라서 첫 번째 전송에 대해 지연 응답이 발생되어 마스터는 세 번째 전송을 바로 시작하지 못하게 된다. 하지만 본 논문에서 개선한 ML-AHB 버스 매트릭스 구조는 그림 7과 같이 입력 스테이지가 제거 되었다.

따라서 주소 및 제어 정보들을 잠시 저장해 둘 Holding 레지스터가 없다. 또한 마스터는 밀리 타입으로 출력되는 중재기의 마스터 선택 신호인 Active 신호를 보고 첫 번째 전송을 보낼지 결정한다. 마스터가 바로 선택된다면, 첫 번째 전송이 시작되지만, 그렇지 않다면 첫 번째 전송을 시작시키지 말아야 한다. 이를 제어하는 신호가 슬레이브의 응답 신호이며, 이때, 마스터의 첫 번째 전송 바로 이전의 IDLE 전송에 대해 지연 응답을 디코더에서 발생시킬 수 있다.

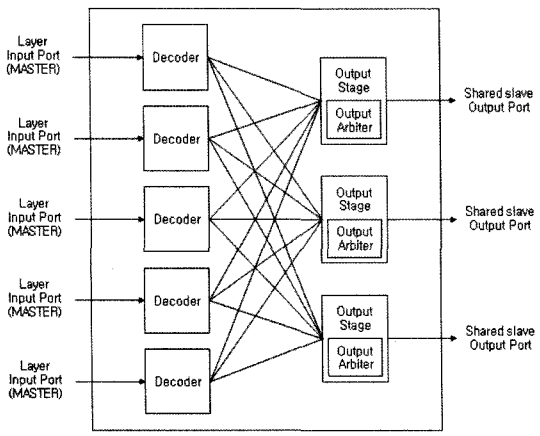

그림 7 개선된 ML-AHB 버스 매트릭스 구조

다음은 개선된 디코더, 출력 스테이지 및 중재기에 대한 설명이다.

3.2.1 디코더

그림 8은 개선된 디코더의 구조이다.

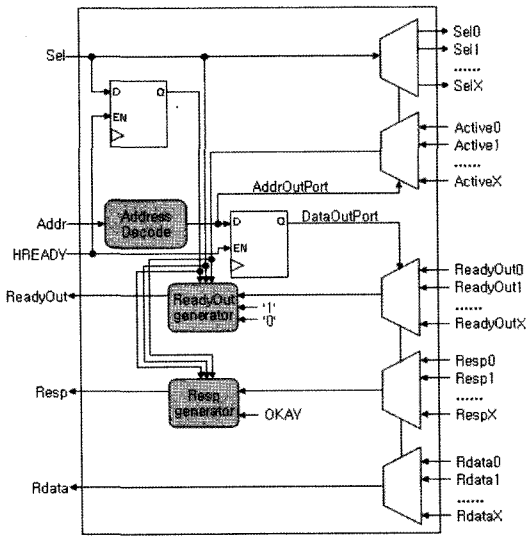

그림 8 개선된 디코더 구조

그림 8과 같이 기존의 디코더 구조에 슬레이브 응답 신호인 ReadyOut 및 Resp 신호 생성기가 추가되었다. 즉, 기존 입력 스테이지의 기능 일부를 디코더에서 수행하게 된다. 새로 추가된 모듈은 중재기의 마스터 선택 신호인 Active 신호에 따라 다음과 같은 동작을 수행한다.

- Active 신호가 '1'이면 출력 스테이지의 HREADYOUT 신호 및 HRESP 신호가 그대로 출력된다.
- Active 신호가 '0'이면 지연 응답을 발생시킨다. 즉, ReadyOut 신호는 '0'으로, Resp 신호는 OKAY로 출력되며, 이때의 지연 응답이 IDLE 전송에 대한 지연 응답이다.

3.2.2 출력 스테이지

그림 9는 개선된 출력 스테이지의 구조이다.

그림 9와 같이 기존 출력 스테이지 구조에서 중재기의 입력 신호만 변경되었다. 즉, 기존의 Sel 신호와

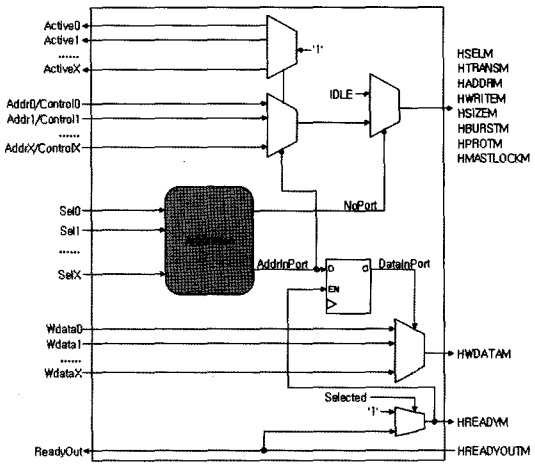

그림 9 개선된 출력 스테이지 구조

Heldtran 신호를 모두 고려한 방법에서, 개선된 구조에선 Sel 신호만 고려한다. 이는 Heldtran 신호를 생성하는 입력 스테이지가 제거되었기 때문이다.

3.2.3 중재기

그림 10은 개선된 중재기의 구조이다.

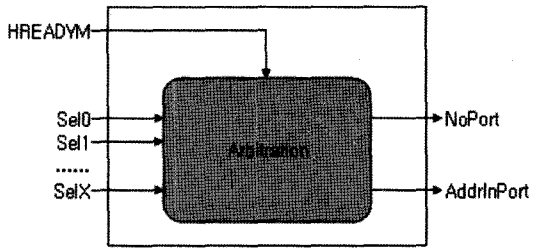

그림 10 개선된 중재기 구조

개선된 중재기는 기존 무어 타입의 중재 방식에서 밀리 타입의 중재 방식으로 변경되었다. 기존의 중재기 구조에선 NoPort와 AddrInPort 신호들이 플립플롭을 거쳐 생성되었지만, 개선된 중재기 에서는 그림 10과 같이 플립플롭이 제거되었다. 따라서 NoPort, AddrInPort 및 마스터 선택 신호인 Active 신호(그림 9)는 밀리 타입으로 출력된다. 또한 입력 신호도 HREADYM 신호와 Sel 신호만 고려된다.

개선된 ML-AHB 버스 매트릭스에 기존의 라운드 로빈 중재 방식을 사용할 경우 다음과 같은 문제점이 발생될 수 있다.

- 전체 트랜잭션을 모두 종료한 마스터가 바로 다음 클럭 사이클에 또 다른 트랜잭션을 시작할 경우, 일반적인 라운드 로빈 방식을 사용하면 그림 11과 같은 문제가 발생될 수 있다.

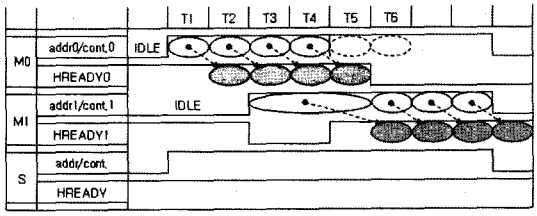

그림 11 타이밍 다이어그램 예제-2

그림 11은 마스터 M0과 M1이 각각 4개의 버스트 전송을 슬레이브 S로 수행하는 모습을 보여준다. T1에서 마스터 M0이 첫 번째 전송을 수행하였고 이에 대한 응답은 T2에서 출력되었다. 이와 같은 방식으로 마스터 M0이 슬레이브 S로 전송을 수행하는 도중 마스터 M1도 T3에서 슬레이브 S로 전송을 요청하였다. 이때, 마스터 M1은 IDLE 전송에 대한 지연 응답으로 인해 잠시 대기하게 된다. 기존의 라운드 로빈 중재 방식을 사용할 경우 T4에서 마스터 M0의 마지막 전송이 끝나면 T5에서는 마스터 M1을 선택하게 된다. 만약, 이때 마스터 M0이 또 다른 전송을 요청한다면, 이전 트랜잭션의 마지막 전송에 대한 응답 신호를 보고 새로운 전송을 바로 시작해 버린다. 하지만, T5에서 선택된 마스터는 M0이 아닌 마스터 M1이다. 즉, 자신이 선택되지도 않았는데 새로운 전송을 시작해 버린 경우가 된다.

이러한 문제점을 해결하기 위해 본 논문에선 기존 라운드 로빈 중재 방식을 개선하였다. 개선된 중재 방식은 마스킹(Masking) 기법[8]을 사용하여 구현한 비 선점 라운드 로빈(Non-Preemptive Round Robin) 기반 중재 방식으로 다음과 같다.

기본적으로 라운드 로빈 기반 중재 방식을 사용하면서, 다음 마스터를 선택할 시점은 현재 선택된 마스터의 Sel 신호를 보고 결정한다. 현재 선택된 마스터가 하나의 트랜잭션을 종료하고, 두 번째 트랜잭션을 바로 수행할 경우, Sel 신호는 계속해서 '1'로 유지될 것이며, 이 신호를 보고 중재기는 현재 선택된 마스터의 Sel 신호가 '0'으로 떨어질 때 까지 계속 버스를 사용할 수 있도록 해준다. 즉, 선택된 마스터의 버스 사용 권한을 대기하고 있는 다른 마스터들이 선점할 수 없게 된다. 이와 같은 비 선점 라운드 로빈 기반 중재 방식은 시스템내의 모든 마스터 IP에 대한 버스 사용 권한 및 응답 시간(Response Time)을 동등하게 보장해주지 않아도 되는 어플리케이션에 적합한 방법이다. 하지만, 시스템내의 모든 마스터 IP에 대한 버스 사용 권한 및 응답 시간을 동등하게 보장해줘야 하는 어플리케이션에 비 선점 라운드 로빈 기반 중재 방식을 사용할 경우, 약간의 기아(Starvation) 상태가 발생될 수 있다. 즉, 선택된 마스터가 계속해서 새로운 트랜잭션을 연속적으로 수행할

경우, 버스 사용을 대기하고 있는 다른 마스터들은 버스를 선점할 수 없게 된다. 따라서 그만큼의 대기 시간이 증가하게 된다. 이러한 기아 문제는 주소 구간과 데이타 구간이 파이프라인 구조인 AMBA AHB 프로토콜 때문에 사실상 해결이 불가능하다. 이 문제를 극복하기 위해서는 약간의 제약이 필요하다. 즉, 마스터가 연속적으로 다수의 트랜잭션을 수행할 때, 각 트랜잭션이 종료될 때마다 최소 1 클럭 사이클의 IDLE 전송을 수행하도록 하는 것이다. 그러면 IDLE 전송을 수행하는 시점에서 버스 사용을 대기하고 있는 다른 마스터들이 라운드 로빈 방식으로 선택될 수 있으며, 따라서 기아 상태가 발생되지 않게 된다. 만약 마스터가 연속적으로 트랜잭션을 수행하지 않는다면 위 제약사항을 반드시 지킬 필요는 없다. 그림 12는 이에 대한 타이밍 다이어그램이다.

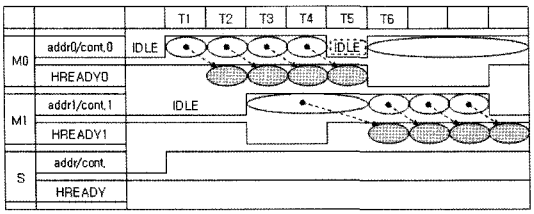

그림 12 타이밍 다이어그램 예제-3

그림 12는 그림 11과 동일한 상황이다. 단, 마스터 M0이 T4에서 첫 번째 트랜잭션을 종료하고 T5부터 새로운 트랜잭션을 바로 수행하지 않고 1 클럭 사이클 IDLE 전송을 수행 한 후, T6부터 새로운 전송을 시작하였다. T5에서는 아무 문제없이 마스터 M1이 선택되어 슬레이브 S로 전송을 수행하고, 마스터 M0은 마스터 M1의 트랜잭션이 종료될 때 까지 대기하게 된다. 결과적으로 마스터 M0과 마스터 M1은 동등하게 버스를 사용할 수 있게 되며, 기아 문제는 해결된다. 또한 마스터 M0이 T5에서 불필요한 IDLE 전송을 수행했다하더라도, 마스터 M1에 의해 슬레이브 S는 계속해서 동작하고 있다. 따라서 전체 버스 사이클은 낭비되지 않는다.

본 논문에서 개선한 디코더, 출력 스테이지 및 중재기로 구성된 ML-AHB 버스 매트릭스에 대한 타이밍 다이어그램은 그림 13과 같다.

그림 13에서 마스터 0과 마스터 1은 동시에 슬레이브 0으로 데이타 전송을 요청하였다. 이때, 마스터 0은 슬레이브 0의 중재기로부터 바로 선택되어 1 클럭 사이클 지연 현상 없이 데이타를 전송하였고, 마스터 1은 마스터 0의 모든 전송이 종료될 때까지 대기하는 모습을 보여준다. 즉, 첫 번째 전송을 수행하기 바로 이전 IDLE 전송에 대한 지연 응답으로 인해 마스터 1은 첫 번째 전송을 수행하지 않고 기다리게 된다.

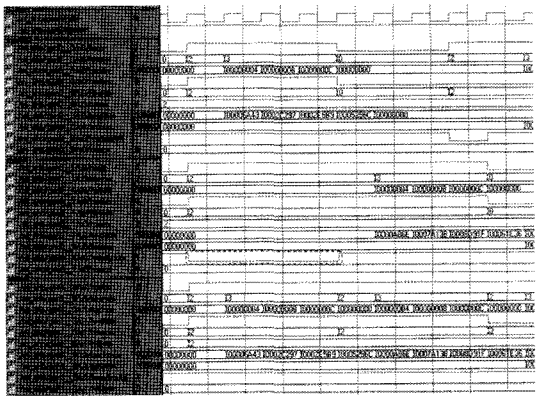

그림 13 타이밍 다이어그램 예제-4

4. 실 험

본 논문에서 개선한 ML-AHB 버스 매트릭스는 합성 가능한 RTL VHDL 코드로 구현되었다. 기존 ML-AHB 버스 매트릭스와 본 논문에서 개선된 ML-AHB 버스 매트릭스의 성능을 비교하기 위해 AMBA AHB 프로토콜을 따르는 AHB-Lite 마스터/슬레이브용 BFM (Bus Functional Model), 프로토콜 검사기(Protocol Checker) 및 성능 모니터(Performance Monitor) 모듈을 VHDL과 FLI C를 이용하여 모델링하였고, 시뮬레이션을 수행하기 위해 ModelSim II 시뮬레이터를 사용하였다. 실험에 사용된 ML-AHB 버스 시스템은 worst case인 8개의 마스터와 1개의 슬레이브로 구성된다.

마스터 BFM은 랜덤 함수에 의해 uniform하게 트랜잭션을 발생시키며 수행할 전체 트랜잭션 개수를 파라미터로 갖는다. 따라서 수행할 전체 트랜잭션의 개수를 각 마스터별로 동일하게 증가시켜가며 성능 시뮬레이션을 수행하였다. 또한 ML-AHB 버스 매트릭스에서 발생되는 고유의 지연 응답만을 측정하기 위해 슬레이브 BFM의 지연 응답은 발생시키지 않았다. 즉, 슬레이브 BFM은 항상 지연 없는 OKAY 응답만을 출력하도록 하였다. 그림 14와 15는 4-beat incrementing 버스트 타입으로 위와 같이 성능 시뮬레이션을 수행했을 때 각각 버스 전체 트랜잭션의 종료 시간과 평균 지연 시간을 측정한 결과이다.

트랜잭션 종료 시간은 개선된 구조가 기존의 구조보다 평균 20% 정도 짧아졌다. 그림 14의 결과 전체 트랜잭션 종료 시간의 감소율은 식 (1)로 나타낼 수 있다.

$$\frac{\text{전체 트랜잭션}}{\text{종료 시간의 감소율}} = \frac{1}{\text{버스트 길이}+1} \times 100 \qquad (1)$$

즉, 기존 버스 매트릭스에서 발생되는 1 클락 사이클 지연 시간이 감소하기 때문에 식 (1)과 같이 나타낼 수 있다. 따라서 버스트 길이가 8, 16일 경우 각각 약

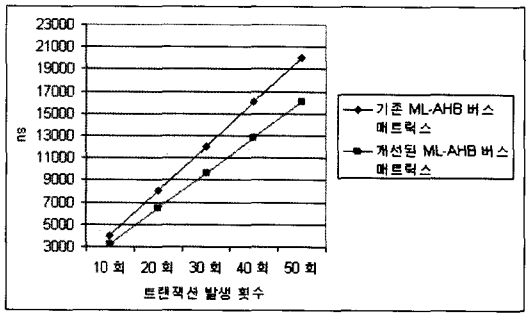

그림 14 버스 트랜잭션 종료 시간 비교

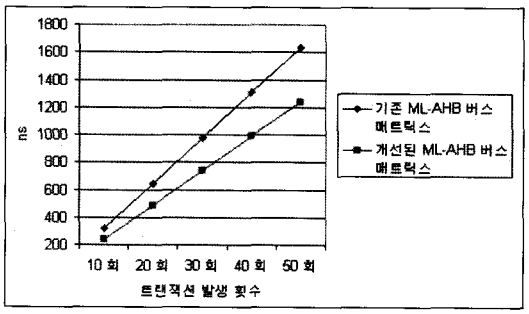

그림 15 평균 지연 시간 비교

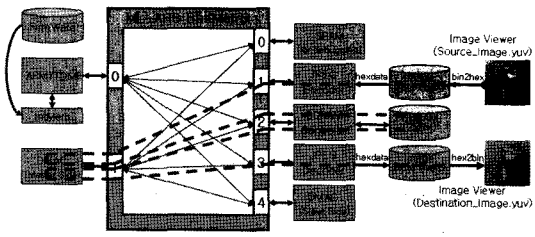

그림 16 JPEG 코덱을 위한 기본 플랫폼

11.1% (1/9), 5.9% (1/17)의 감소율을 예측할 수 있다.

평균 지연 시간은 개선된 구조가 기존의 구조보다 평균 24% 정도 짧아진 결과를 보여준다. 이와 같은 성능 시뮬레이션을 기반으로 본 논문에서 개선한 ML-AHB 버스 매트릭스를 실제 시스템에 적용시키기 위해 그림 16과 같이 JPEG 코덱을 위한 기본 플랫폼을 구성하여 실험을 수행하였다.

그림 16은 마스터가 2개 (ARM7TDMI, DMAC), 슬레이브가 5개 (SRAM, JPEG Encoder, SDRAMC, JPEG Decoder, DMAC)로 구성된 ML-AHB 버스 매트릭스 기반의 기본 플랫폼이다. ARM7TDMI는 펌웨어 프로그램에 따라 SDRAM과 DMA 컨트롤러의 컨트롤 레지스터를 설정해 주는 동작을 수행하며, 실제 이미지 데이타 전송은 DMA에 의해 이루어진다. 그림 16에서 점선으로 표시된 부분이 이미지 데이타의 이동 경로를

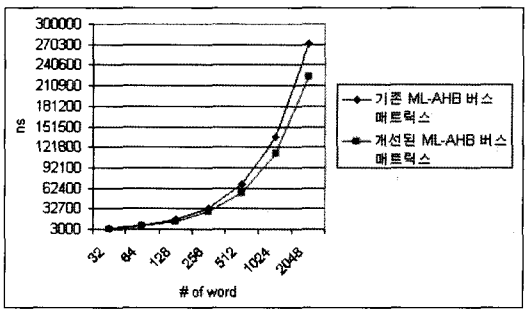

그림 17 버스 트랜잭션 종료 시간 비교

보여준다. (JPEG Encoder -> DMAC -> SDRAM -> DMAC -> JPEG Decoder) 이와 같은 이동 경로로 DMA는 일정량의 이미지 데이타 (단위: 워드)를 연속해서 전송하게 된다. 그림 17은 그림 16의 기본 플랫폼에서 8-beat incrementing 버스트 타입으로 성능 시뮬레이션을 수행한 결과이다.

그림 17의 결과 트랜잭션 종료 시간은 개선된 구조가 기존의 구조보다 평균 17% 정도 짧아졌다. 식 (1)에 의하면 트랜잭션 종료 시간의 감소율이 8-beat 버스트 일 때, 11.1%정도 되어야 하는데, 실험 결과는 그렇지 않다. 그 이유는 SDRAM의 Refresh 타임 때문이다. 즉, SDRAM은 Auto Refresh 타입으로 주기적으로 Refresh가 이루어지며, 이때 SDRAM에 데이타 전송이 중첩되면 그만큼의 지연이 발생된다. 시뮬레이션 결과 개선된 구조가 기존의 구조보다 버스트 트랜잭션을 빨리 끝내기 때문에, 데이타 전송이 SDRAM의 Refresh 타임과 중첩되는 경우가 기존의 구조보다 적었음을 확인하였다. 또한 버스트 길이가 길수록 데이타 전송이 SDRAM의 Refresh 타임과 중첩될 확률이 높기 때문에 8-beat 버스트 타입으로 데이타 전송을 수행했을 경우, 그림 17과 같이 11.1%의 트랜잭션 종료 시간의 감소율보다 약간 더 큰 감소율을 보이는 것이다.

기존 ML-AHB 버스 매트릭스와 개선된 ML-AHB 버스 매트릭스의 하드웨어 오버헤드를 측정하기 위해 마스터와 슬레이브의 개수를 증가시켜가며 로직 합성을 수행하였다. 합성에 사용된 툴은 XILINX ISE 6.3이며 타겟 디바이스는 virtex2 xc2v3000-4 (XILINX FPGA)이다. 표 1은 합성 결과를 보여준다.

개선된 ML-AHB 버스 매트릭스에는 다수의 플립플롭들을 포함하고 있는 입력 스테이지가 없기 때문에 합성 결과 플립플롭의 개수가 현저하게 감소하였다. 플립플롭 개수는 개선된 구조가 기존의 구조보다 평균 83% 정도 작아졌고, LUT도 평균 22% 정도 작아졌다. 클럭 주기는 입력 스테이지 내부의 주소 및 전송 정보들이 통과하는 2×1 멀티플렉서가 없기 때문에 개선된 구조가

표 1 합성 결과

기존 ML-AHB 버스 매트릭스				
버스구성 (마스터 × 슬레이브)	LUT (#)	플립플롭 (#)	클럭주기 (ns)	슬라이스 (#)
2 × 2	411	114	6.772	231
4 × 4	1396	249	8.711	720
6 × 6	3506	412	10.319	2037
8 × 8	5445	556	11.087	2805
개선된 ML-AHB 버스 매트릭스				
버스구성 (마스터 × 슬레이브)	LUT (#)	플립플롭 (#)	클럭주기 (ns)	슬라이스 (#)
2 × 2	263	12	3.850	150
4 × 4	1089	36	5.875	554
6 × 6	2845	78	7.696	1680
8 × 8	4785	112	9.101	2453

기존의 구조보다 평균 29% 정도 짧아진 결과를 보여주고 있으며, 전체적인 슬라이스 개수도 평균 22% 정도 감소한 결과를 보여준다.

5. 결론 및 향후 과제

본 논문에서는 기존 ML-AHB 버스 매트릭스의 구조를 개선하여 1 클럭 사이클 지연 문제를 해결하였다. 기존 ML-AHB 버스 매트릭스는 내부 컴포넌트인 입력 스테이지와 무어 타입으로 구현된 중재 방식으로 인해 마스터가 새로운 전송을 시작할 때 또는 슬레이브 레이어를 변경할 때 마다 항상 1 클럭 사이클 지연 현상이 발생하였다. 본 논문에서는 이 문제를 해결하기 위해 기존 ML-AHB 버스 매트릭스 구조에서 입력 스테이지를 제거하였고, 무어 타입으로 구현된 중재 방식을 밀리 타입으로 구현된 중재 방식으로 변경하였으며, 기존 입력 스테이지의 기능 일부를 디코더가 수행하도록 디코더의 구조를 개선하였다. 또한, 일반적인 라운드 로빈 중재 방식을 사용할 때 발생되는 문제점을 해결하기 위해 개선된 ML-AHB 버스 매트릭스 구조에 적합한 비 선점 라운드 로빈 중재 방식을 제안하였다. 비 선점 라운드 로빈 중재 방식은 시스템상의 모든 마스터 IP에 대한 버스 사용 권한과 응답 시간을 동등하게 보장해주지 않아도 되는 어플리케이션에 적합한 방식이다. 하지만, 모든 마스터 IP의 버스 사용 권한과 응답 시간을 동등하게 보장해줘야 하는 어플리케이션에 비 선점 라운드 로빈 중재 방식을 사용할 경우, 약간의 기아 상태가 발생된다. 이 문제는 주소 구간과 데이타 구간이 파이프라인 구조인 AMBA AHB 프로토콜 때문에 해결이 불가능하다. 이와 같은 기아 문제를 극복하기위해 마스터 IP가 다수의 트랜잭션을 연속적으로 수행할 때, 각 트랜잭션이 종료될 때 마다 최소 1 클럭 사이클의 IDLE 전송을 수행시키는 방법을 제안하였다.

본 논문에서 개선한 ML-AHB 버스 매트릭스는 4-beat incrementing 버스트 타입으로 다수의 트랜잭션을 수행할 경우, 기존 ML-AHB 버스 매트릭스보다 전체 트랜잭션 종료 시간 및 평균 지연 시간이 각각 약 20%, 24% 정도 짧아졌다. 또한 플립플롭 및 LUT 개수도 각각 약 83%, 22% 정도 작아졌고, 클럭 주기도 개선된 구조가 기존의 구조보다 약 29% 정도 짧아졌다. 전체적인 슬라이스 수는 기존의 ML-AHB 버스 매트릭스보다 약 22% 정도 감소하였다.

추후, 본 논문에서 개선한 ML-AHB 버스 매트릭스를 데이타 처리량이 많고, 마스터와 슬레이브 개수도 많은 멀티미디어 어플리케이션(Video Phone, MPEG-4, H.264 codec 등)에 실제 적용시킬 예정이다. 또한, AMBA 버스 외의 다른 종류의 고성능 온 칩 버스 구조(예: IBM의 PLB Crossbar Switch, Wishbone의 CONMAX, Sonics Inc.의 Silicon Backplace 등)와의 성능 비교에 대한 연구도 필요하다.

참 고 문 헌

[1] 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.

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

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

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

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

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

[7] "SiliconBackplaneTM III MicroNetwork IP, "http://www.sonicsinc.com/sonics/products/siliconbackplaneIII/

[8] Sung-Ho Moon; Dan Keun Sung "High-performance variable-length packet scheduling algorithm for IP traffic," Global Telecommunications Conference, GLOBECOM '01. IEEE, Vol. 4, Nov. 2001 Page(s):2666 - 2670.

황 수 연

2002년 한남대학교 컴퓨터공학과 졸업(학사). 2004년 충남대학교 대학원 컴퓨터공학과 졸업(석사). 2004년~현재 충남대학교 대학원 컴퓨터공학과 박사과정 관심분야는 설계자동화, 온 칩 버스 설계, 시스템 온 칩 설계

장 경 선

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