[1]버스 매트릭스 구현 및 ML(Multi-Layer) AHB를 위한 테스트 환경

황수연° 장경선
충남대학교
{charisma°, ksjhang}@ce.cnu.ac.kr

An Implementation of Bus Matrix and Testing Environments for ML AHB

Soo-Yun Hwang° Kyoung-Son Jhang
Dept. of Computer Engineering, Chungnam National University

요 약

SoC 분야에서 온 칩 버스는 전체 시스템의 성능을 결정하는 중요한 요소이다. 이에 따라 최근 ARM 사에서는 고성능 온 칩 버스 구조인 ML(Multi-Layer) AHB 버스를 제안하였다. ML AHB 버스는 저전력 임베디드 시스템에 적합한 버스 구조로써 현재 널리 사용되고 있다. 하지만, 고가이기 때문에 ADK(AMBA™ Design kit) 구매에 대한 부담이 적지 않다. 본 논문은 ML AHB의 버스 구조인 버스 매트릭스 구현 및 ADK에서 제공되지 않는 테스트 환경 즉, Protocol Checker 및 Performance Monitor Module 구현에 관한 것이다.

1. 서 론

반도체 공정기술이 발달하고 시스템의 집적도가 높아짐에 따라 SoC(System on a Chip)가 가능해졌다. SoC의 각 IP(Intellectual Property)는 온 칩 버스를 통해 데이터를 전송한다. 따라서 온 칩 버스에 따라 전체 시스템의 성능이 달라질 수 있다. 이는 온 칩 버스의 구조가 전체 시스템의 성능을 결정하는 중요한 요소임을 의미한다.

기존의 온 칩 버스에서 사용하는 공유버스는 한번에 하나의 마스터만이 데이터 전송이 가능하다. 따라서 다른 후보 마스터들의 대기시간이 증대되며, 결국 전체 버스 시스템의 성능이 저하되는 결과를 초래한다. 특히, 데이터 처리량이 많은 멀티미디어 IP나 많은 프로세서 요소들을 사용하는 휴대형 기기 및 통신 기기 등에 기존의 온 칩 버스를 사용할 경우 높은 성능을 얻기 힘들다. 이러한 제약점을 극복하기 위해, ARM 사의 AMBA 버스(ML AHB)[1], IBM 사의 CoreConnect(PLB Crossbar Switch)[2], Wishbone 버스(CONMAX)[3], Sonics Inc. 의 Silicon Backplane[4]버스들이 제안되었다. 특히, ARM 사의 AMBA 버스는 저전력 임베디드 시스템에 적합한 버스 구조로써 현재 널리 사용되고 있는 고성능 온 칩 버스 구조이다.

ARM 사에서 개발한 고성능 온 칩 버스 구조는 ML(Multi-Layer) AHB로써, AHB 프로토콜에 기반한 버스 매트릭스 구조의 새로운 연결 방식이다. ML AHB는 시스템 내의 다중 마스터와 다중 슬레이브간의 병렬적인 접근 경로를 제공한다. 이는 한번에 하나의 마스터만 데이터 전송이 가능했던 기존 온 칩 버스의 제약점을 극복할 수 있게 해준다. ML AHB는 보다 복잡한 연결 매트릭스를 사용함으로써 전체 버스 대역폭을 증가시켜주고, 최근 많은 프로세서 요소들을 사용하는 휴대형 기기 및 통신 기기 등에 적합한 온 칩 버스 구조이다. 하지만, ML AHB 버스를 사용하기 위해서는 고가의 ADK(AMBA™ Design Kit)을 구매해야 한다.

본 논문은 ML AHB의 버스 구조인 버스 매트릭스 구현 및 ADK에서 제공되지 않는 테스트 환경 즉, Protocol Checker 및 Performance Monitor Module 구현에 관한 것이다. 본 논문의 2장에서는 구현된 버스 매트릭스의 구조를 간략하게 설명하고, 3장에서는 ML AHB를 위한 테스트 환경이 소개된다. 4장에서는 시뮬레이션 결과 파형 및 합성 결과를 보여주며, 5장에서 결론 및 향후 과제에 대해 언급한다.

2. 버스 매트릭스 구조[1]

본 논문에서 구현한 버스 매트릭스는 VHDL로 구현되었으며 그림 1과 같이 Input stage, Decode stage, Output stage, Output arbiter로 구성된다. Input stage는 마스터가 공유 슬레이브쪽으로 바로 접근할 수 없을 때, 주소와 제어 정보들을 임시로 보관해 두는 일을 담당한다. 즉, 두 개 이상의 마스터가 하나의 공유 슬레이브쪽으로 동시에 접근할 때 발생되는 충돌을 막기 위해 데이터 흐름을 제어해 주는 역할을 한다. Decode stage는 마스터쪽으로부터 주소를 입력 받아 해당되는 슬레이브를 결정한다. Output stage는 각 공유 슬레이브에 연

1) 본 논문은 정보통신부의 출연금으로 수행한 IT SoC 핵심설계 인력양성 사업의 수행결과이며, 본 논문에 사용된 CAD 툴은 IDEC 으로부터 지원받았습니다.

결되며 최종적으로 마스터와 슬레이브 사이의 데이터 전송 경로를 결정한다. 이때, 슬레이브 중심으로 어떠한 마스터를 선택할 것인지 결정하기 위해 Output arbiter를 사용한다. 즉, Output stage는 Output arbiter를 내장하고 있는 형태이다. 또한 Output stage는 마스터와 슬레이브 사이에서 전송되는 주소, 제어 정보, 데이터들을 모두 포함하고 있다.

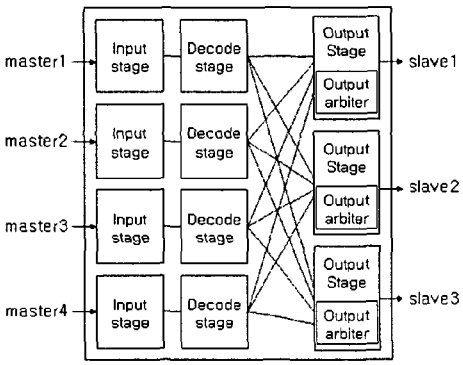

그림 1. 버스 매트릭스 구조 (마스터 수: 4, 슬레이브 수: 3)

3. ML AHB를 위한 테스트 환경

버스 매트릭스를 사용하여 하나의 ML AHB 버스 시스템을 구성할 때, 고려해야할 사항 중 하나는 테스트 전략을 어떻게 세울 것인가에 대한 문제이다. 이는 곧 전체 시스템 설계 시간과도 밀접한 관계가 있다. 보통 버스 시스템의 경우 확인해야 할 프로토콜과 모니터링 해야 할 신호들이 굉장히 많기 때문에, Mentor 사의 ModelSim[5]과 같은 GUI 기반의 전용 시뮬레이터만으로 테스트 할 경우 많은 시간이 소비될 뿐만 아니라, 테스트함에 있어서 발생되는 불편함이 적지 않다. 또한 ADK에서는 적절한 테스트 환경이 지원되지 않는다.

본 논문에서 구현한 테스트 환경은 그림 2와 같이 Master/Slave Protocol Checker와 Performance Monitor Module로 구성된다. Protocol Checker는 버스 매트릭스에 연결된 마스터/슬레이브 모듈이 AMBA AHB 프로토콜에 호환되는지 체크하는 모듈로써 Master Protocol Checker와 Slave Protocol Checker로 구성된다. 그림 2와 같이 Master Protocol Checker는 AHB 마스터 모듈과 Input stage 사이의 신호들을, Slave Protocol Checker는 Output stage와 AHB 슬레이브 모듈 사이의 신호들을 각각 모니터링한다. 이 모듈들은 미리 정의된 Protocol Compliance 체크 요소들에 따라 구현되었으며, 트랜잭션 수행 중, 정의된 요소들을 위반했을 경우 텍스트 파일 형태로 결과가 출력된다. 이때 출력되는 정보에는 에러 발생 시간 및 에러 내용 등이 있다. Master/Slave Protocol Checker는 VHDL로 구현되었으며, 결과 출력을 위해 ModelSim FLI(Foreign Language Interface)[5] C 모듈을 이용하였다.

Performance Monitor Module은 어떤 마스터가 어떤 슬레이브쪽으로 어떤 종류의 전송을 수행했는지에 대하여 모니터링 하는 모듈이다. 특히 버스 매트릭스는 특성상 Output stage에서 마스터와 슬레이브 사이에 전송된 모든 정보(주소, 제어정보, 데이터)들을 포함하고 있기 때문에, 이 부분의 외부 입출력 신호들만 모니터링하면 된다(그림 2). 이러한 Performance Monitor Module은 버스 사용율 측면에서 중요한 정보가 되며, 추후에 성능 측정 및 비교 분석 등에 유용하게 활용될 수 있는 정보들이다. Performance Monitor Module도 Protocol Checker와 같이 VHDL로 구현되었고, 결과 출력을 위해 ModelSim FLI C 모듈을 이용하였다.

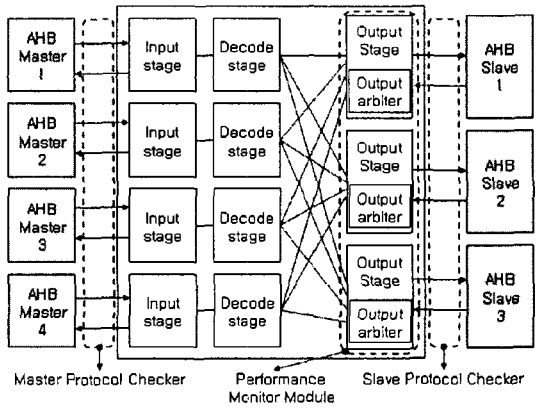

그림 2. 본 논문에서 구현한 ML AHB를 위한 테스트 환경

4. 시뮬레이션 결과 및 분석

본 논문에서 구현한 버스 매트릭스가 정상적으로 동작하는지 검증하기 위해 그림 3과 같이 VHDL로 구현된 AHB 마스터/슬레이브 트랜잭터를 이용하여 ML AHB 버스 시스템을 구현하였다. 또한 테스트 시간을 줄이기 위해 3장에서 설명한 Master/Slave Protocol Checker 및 Performance Monitor Module을 이용했다.

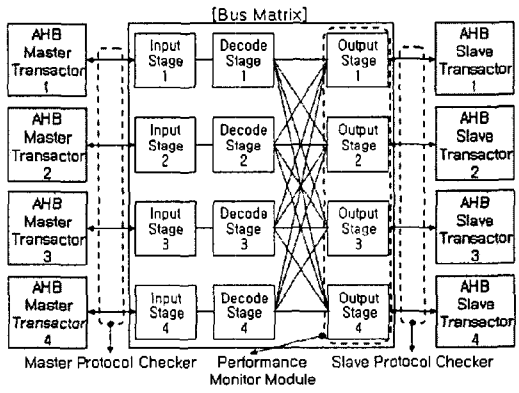

그림 3. 본 논문에서 구현한 ML AHB 버스 시스템 (마스터 수: 4, 슬레이브 수: 4)

그림 3의 마스터/슬레이브 트랜잭터는 설정 파라메타에 따라 무작위로 트랜잭션 및 응답을 생성한다. 그림 4

는 ModelSim[5] 시뮬레이터를 이용한 시뮬레이션 결과 파형이다.

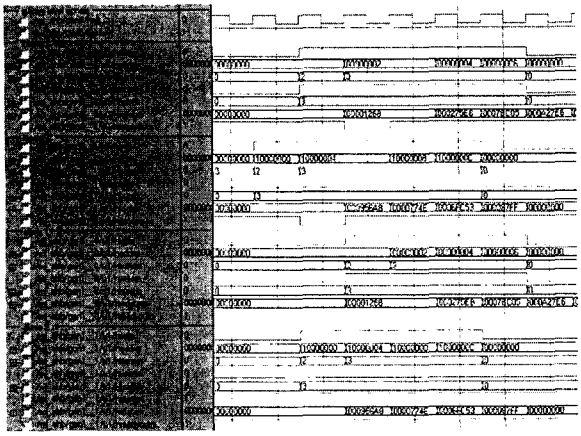

그림 4. 시뮬레이션 결과 파형

그림 4는 마스터 1과 마스터 2가 각각 슬레이브 1과 슬레이브 2를 4-beat incrementing 버스트 타입으로 동시에 접근한 시뮬레이션 결과 파형을 보여준다. 또한 마스터 1은 쓰기 전송을, 마스터 2는 읽기 전송을 수행하였으며, 슬레이브 1과 2는 모두 zero-wait OKAY로 응답하였다. 그림 4의 마스터와 슬레이브 사이에서 전송된 주소, 제어 정보, 데이터들을 보면 본 논문에서 구현한 버스 매트릭스가 정상적으로 동작함을 확인할 수 있다. 또한 시뮬레이션 수행 후, Performance Monitor Module에 의해 생성된 모든 전송 정보를 담고 있는 텍스트 파일은 그림 5와 같다. 그림 5의 Time 정보는 그림 4의 시뮬레이션 시간과 동일하다.

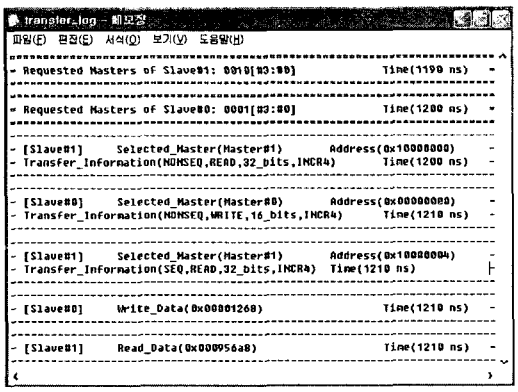

그림 5. Performance Monitor Module에 의해 생성된 텍스트 파일

그림 5는 슬레이브에 어떤 마스터들이 버스 사용을 요청했는지(후보 마스터), 어떤 마스터가 최종적으로 버스 사용권을 얻었는지에 대한 정보를 포함하고 있다. 또한 어떤 마스터가 어떤 슬레이브에 어떤 종류의 전송을 수행했는지에 대한 정보들도 담고 있다.

본 논문에서 구현한 버스 매트릭스는 VHDL로 구현되었으며, 표 1은 마스터 수가 4이고 슬레이브 수가 4인 버스 매트릭스의 합성 결과이다. 합성에 사용된 툴은 XILINX ISE 6.2[6]와 Synplify Pro 7.6[7]이며, 타겟 디바이스는 virtex2 xc2v3000-4(XILINX FPGA)이다.

표 1. 합성 결과 표

XILINX ISE로 합성한 경우	
4 input LUTs #	1416 out of 28672 (4%)
Clock Period	8.624 ns (115.955 MHz)
Synplify Pro로 합성한 경우	
4 input LUTs #	1566 out of 28672 (5%)
Clock Period	7.511 ns (132.4 MHz)

표 1과 같이 면적은 XILINX ISE로 합성한 경우 적게 나왔고, Clock Period는 Synplify Pro로 합성한 경우가 더 짧게 나왔다.

5. 결론 및 향후 과제

최근 ARM 사에서는 고성능 온 칩 버스 구조인 ML AHB 버스를 제안하였다. ML AHB 버스는 저전력 임베디드 시스템에 적합한 버스 구조로써 현재 널리 사용되고 있다. 하지만, 고가이기 때문에 ADK 구매에 대한 부담이 적지 않다.

본 논문은 ML AHB의 버스 구조인 버스 매트릭스 구현 및 ADK에서 제공되지 않는 테스트 환경 즉, Protocol Checker 및 Performance Monitor Module 구현에 관한 것이다. 본 논문에서 구현한 버스 매트릭스가 정상적으로 동작하는지 검증하기 위해 트랜잭터를 이용한 ML AHB 버스 시스템을 구현하였고, 정상적으로 동작하는 시뮬레이션 결과 파형 및 합성 결과를 보였다. 또한 Performance Monitor Module에 의해 생성된 모든 전송 정보들을 담고 있는 텍스트 파일을 통해 어떤 마스터가 어떤 슬레이브에 어떤 종류의 전송을 수행했는지 확인하였다. 본 논문에서 구현한 테스트 환경은 실제 버스 매트릭스 구현 시 적용시켜 설계 시간 단축의 효과를 보았다. 추후, Performance Monitor Module의 결과 중, 버스 사용 시간 및 대기 시간 등을 추출하여 버스 전체의 성능평가 출력에 대한 연구가 요구된다.

참고문헌

[1] "AMBA Specification", http://www.arm.com/products/solutions/AMBA_Spec.html
[2] "The CoreConnect Bus Architecture", http://www-3.ibm.com/chips/products/coreconnect/
[3] "Wishbone", http://www.opencores.org
[4] "SiliconBackplane™ III MicroNetwork IP" http://www.sonicsinc.com/sonics/products/siliconbackplaneIII/
[5] "ModelSim", http://www.mentor.com/
[6] "XILINX", http://www.xilinx.com/
[7] "Synplify Pro", http://www.synplicity.com/