SoC의 성능 향상을 위한 크로스바 스위치 온칩 버스 설계

허정범* · 류광기**

Design of Crossbar Switch On-chip Bus for Performance Improvement of SoC

Jung-bum Heo* · Kwang-ki Ryoo**

이 논문은 IDEC의 CAD Tool 지원 및 중소기업청의 산학협력실 지원사업의 연구결과임.

요 약

기존에 사용되는 대부분의 SoC는 공유버스 구조를 가지고 있어, 병목현상이 발생하는 문제점을 가지고 있다. 이러한 문제점은 SoC의 내부의 IP 수가 많을수록, 전체적인 SoC의 성능을 저하시키게 되어, CPU 자체의 속도보다는 전체적인 통신 분배에 의해 SoC의 성능이 좌우 된다. 본 논문에서는 공유버스의 단점인 병목현상을 줄이고 SoC의 성능을 향상시키기 위해 크로스바 스위치버스 구조를 제안한다. 크로스바 스위치 버스는 마스터 모듈 8개, 슬레이브 모듈 16개까지 연결이 가능하며, 다중 버스 채널구조로 되어 있어 병렬통신이 가능하다. 또한 각 16개의 슬레이브 인터페이스마다 우선순위 정보가 저장된 아비터가 내장되어 하나의 마스터가 슬레이브를 독점하는 것을 방지하는 것과 동시에 효율적인 통신을 지원한다. OpenRISC 프로세서, VGA/LCD 제어기, AC97 제어기, 디버그 인터페이스, 메모리 인터페이스로 구성되는 SoC 플랫폼의 WISHBONE 온칩 공유버스 구조와 크로스바 스위치 버스 구조의 성능을 비교한 결과, 기존의 공유버스보다 26.58%의 성능이 향상되었다.

ABSTRACT

Most of the existing SoCs have shared bus architecture which always has a bottleneck state. The more IPs are in an SoC, the less performance it is of the SoC, Therefore, its performance is effected by the entire communication rather than CPU speed. In this paper, we propose cross-bar switch bus architecture for the reduction of the bottleneck state and the improvement of the performance. The cross-bar switch bus supports up to 8 masters and 16 slaves and parallel communication with architecture of multiple channel bus. Each slave has an arbiter which stores priority information about masters. So, it prevents only one master occupying one slave and supports efficient communication. We compared WISHBONE on-chip shared bus architecture with crossbar switch bus architecture of the SoC platform, which consists of an OpenRISC processor, a VGA/LCD controller, an AC97 controller, a debug interface, a memory interface, and the performance improved by 26.58% than the previous shared bus.

키워드

오픈리스크, 온칩버스, 시스템온칩, 크로스바 스위치, 공유 버스

Key word

OpenRISC, On-chip bus, SoC, Crossbar Switch, Shared Bus

* 한밭대학교 정보통신공학과 석사과정

** 한밭대학교 정보통신공학과 부교수 (교신저자)

접수일자 : 2009. 09. 20

심사완료일자 : 2009. 11. 06

Ⅰ. 서 론

최근 EDA 툴의 기술적인 향상과 반도체 공정의 발달로 IC 설계자들은 RISC 프로세서, DSP 프로세서, 메모리 등 많은 IP가 하나로 집적되는 SoC구조가 가능해 지게 되었다. 또한 해를 거듭할수록 칩의 집적도가 기하학적으로 높아지고, 하나의 SoC에 요구되어지는 기능들 또한 많아지게 되어, SoC 안에는 많은 수의 IP들이 내장하게 되었다. 이러한 기술의 발달 및 하나의 칩 안에 요구되어지는 IP들의 수가 많아짐에 따라, SoC 내부의 IP간에 효율적인 통신과 균형있는 연산의 분배가 SoC의 전반적인 성능을 좌우 하게 되었다.

기존 대부분의 SoC에서는 하나의 버스를 공유하여 상호간에 통신을 하는 공유버스 구조로 이루어져 있는데, 이러한 구조는 다수의 IP들이 하나의 공유된 버스를 사용하려 하기 때문에 병목현상을 야기하는 원인이 되어 SoC의 전반적인 성능을 떨어뜨리게 된다. Time-to-Market이 짧아짐에 따라 플랫폼을 이용한 설계가 증가 하게 되었는데, 플랫폼을 이용한 SoC설계에도 이러한 공유버스 구조를 가지고 있어 SoC의 성능을 제대로 살리지 못하고 있다. 이러한 문제를 해결하기 위하여 병렬적인 통신구조가 요구된다[1-2].

본 논문에서는 공유버스 구조의 문제점인 병목현상을 해결하고, 효율적인 통신을 통해 SoC플랫폼의 성능을 향상시키기 위하여 크로스바 스위치 온칩 버스 구조를 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 기존의 공유버스 구조 및 문제점을 설명하고, 3장에서는 제안된 크로스바 스위치 온칩 버스의 특징 및 구조를 설명하며, 4장에서는 SoC플랫폼을 이용한 통합 설계 및 검증 결과를 설명한다. 마지막으로 5장에서는 본 연구의 결론을 도출한다.

Ⅱ. 기존의 공유버스 구조

기존의 공유 버스 구조는 최대 8개의 마스터와 9개의 슬레이브를 연결할 수 있으며, 크게 마스터 인터페이스, 슬레이브 인터페이스로 구성된다. 그림 1은 공유버스의 전체 블록도이다. 공유버스는 크게 마스터 인터페이스와 슬레이브 인터페이스로 구성된다. 마스터 인터페이스는 8개의 마스터 중 하나의 마스터를 선택하여 공유버스를 통해 슬레이브 인터페이스와 통신하도록 제어한다. 마스터 인터페이스는 마스터 모듈들로부터 버스를 사용하겠다는 요청신호를 받으면, 아비터에 의해 요청신호를 보낸 마스터들 중 가장 높은 우선순위를 갖는 마스터가 버스의 소유권을 점유한다. 우선순위는 마스터 0, 1, 2, 3, 4, 5, 6, 7 순서로 고정된다. 슬레이브 인터페이스는 공유버스를 통해 전달받은 주소를 디코딩하여 8개의 슬레이브 중 하나의 슬레이브를 선택하여 마스터와 통신이 이루어지도록 제어한다.

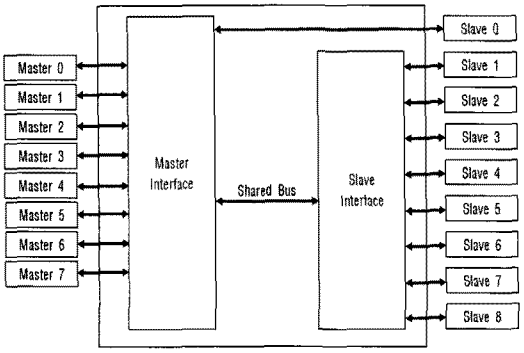

그림 1. 공유버스 전체 블록도
Fig. 1 Block diagram of Shared Bus

이 공유버스는 단순한 구조와 프로토콜로 구성되지만, 동등한 우선순위레벨을 갖는 마스터들이 존재하지 않는다. 따라서 우선순위가 높은 하나의 마스터가 버스를 독점하는 문제점이 있다. 또 다른 문제점은 하나의 공유버스를 통하여 다수의 마스터와 슬레이브가 통신한다는 점이다. 한번에 하나의 마스터와 슬레이브만이 버스를 점유할 수 있기 때문에, 다른 마스터들은 이미 버스를 점유한 마스터와 슬레이브 간의 통신이 끝날 때까지 기다려야하는 병목현상이 일어난다.

Ⅲ. 다중 채널 버스구조

다중 채널 버스는 기존의 공유버스 구조의 문제점인 병목현상을 해결하고, 균형 있고 효율적인 통신을 지원한다.

그림 2는 다중 채널 버스의 내부구조로 최대 8개의 마스터와 16개의 슬레이브를 연결할 수 있어, 기존의 버스 구조보다 뛰어난 확장성을 가지며, 마스터 인터페이스, 슬레이브 인터페이스, 레지스터 파일로 구성된다.

마스터 인터페이스는 마스터 모듈로부터 통신 요청 신호를 받으면, 주소를 디코딩하여 16개의 통신채널 중 하나를 선택하여, 통신하고자 하는 슬레이브 인터페이스에 요청신호를 보낸다. 슬레이브 인터페이스는 슬레이브 모듈과 통신하기를 원하는 각 마스터 모듈들의 우선순위를 검사하여 우선순위가 가장 높은 마스터 모듈과 슬레이브 모듈을 연결해준다. 레지스터 파일은 16개의 16비트 레지스터를 가지며, 각 레지스터는 슬레이브 인터페이스와 매핑되어 마스터 모듈 8개의 우선순위 정보를 제공한다.

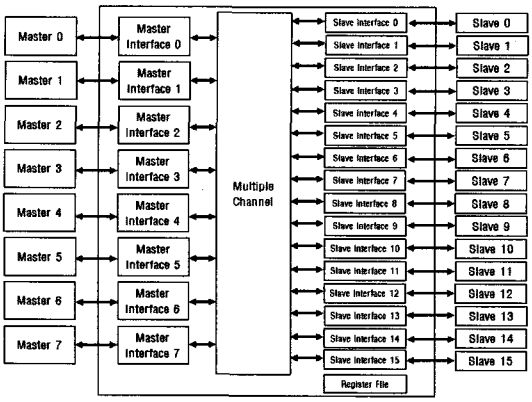

그림 2. 다중채널 버스 전체 블록도
Fig 2. Block Diagram of multiple channels bus

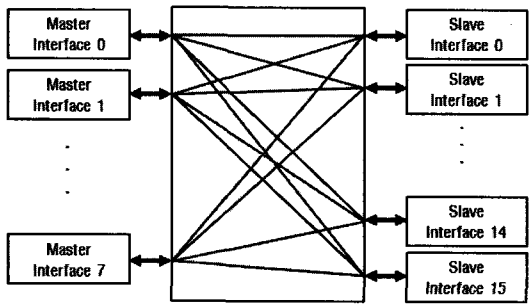

그림 3. 다중채널 내부 구조
Fig. 3 Multiple channels architecture

그림 3은 다중 채널 버스의 내부 구조이다. 다중 채널 버스는 WISHBONE 스펙의 크로스바 스위치 연결 구조이다. 각 마스터는 16개의 슬레이브에 접근이 가능한 16개의 통신채널을 확보하여, 다른 마스터가 사용하고 있지 않은 슬레이브와 통신이 가능하도록 하여 병렬통신을 지원한다. 따라서 공유버스구조의 각 마스터들이 하나의 통신채널을 공유하여 사용함으로써 발생되는 병목현상의 문제점을 해결한다.

3.1 마스터 인터페이스

그림 4는 마스터 인터페이스의 내부 구조이다. 마스터 인터페이스는 마스터로부터 통신 요청 신호를 받으면, 주소를 판별해 16개의 연결된 통신채널중 하나를 선택하여 마스터가 원하는 슬레이브 인터페이스에 통신 요청 신호를 보내는 역할을 하며, 연결이 이루어진 이후에는 연결된 마스터와 슬레이브간에 통신이 이루어지도록 지원한다.

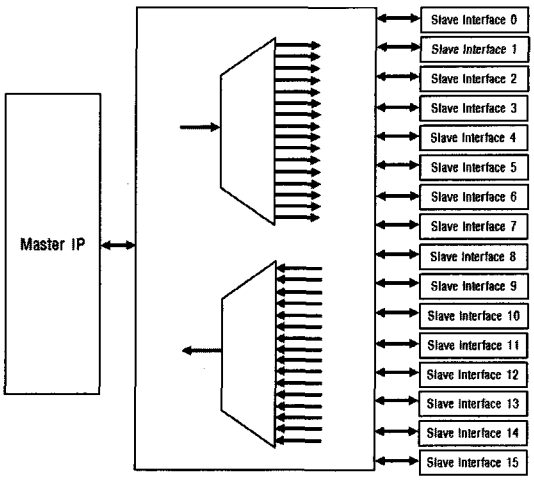

그림 4. 마스터 인터페이스 블록도
Fig. 4 Block diagram of Master Interface

3.2 슬레이브 인터페이스

그림 5는 슬레이브 인터페이스를 나타낸다. 슬레이브 인터페이스는 마스터선택제어기에 의해 통신 요청이 들어온 마스터들 중 하나를 선택하여 슬레이브와 통신이 이루어지도록 지원한다.

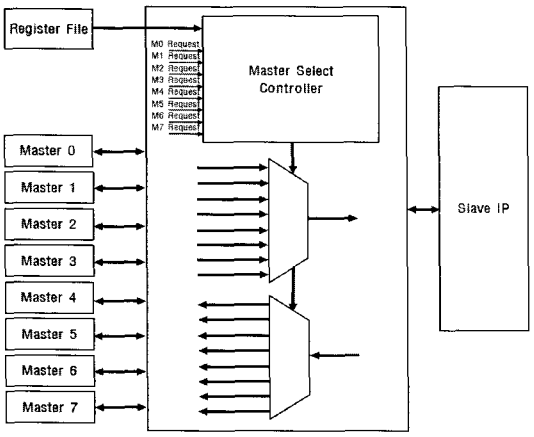

그림 5. 슬레이브 인터페이스 블록도
Fig. 5 Block diagram of Slave Interface

그림 6은 마스터 선택제어기의 구조를 나타낸 것으로써 0부터 3까지 4단계의 우선순위 아비터와 우선순위선택모듈로 구성 된다. 우선순위는 3이 가장 높으며, 0이 가장 낮다. 우선 마스터 모듈로부터 연결요청 신호를 받으면, 우선순위선택모듈은 레지스터 파일로부터 제공 받은 마스터 모듈들의 우선순위 정보를 비교하여 연결요청을 한 마스터 모듈들의 우선순위를 비교해 가장 높은 우선순위를 값을 멀티플렉서(Multiplexer)로 출력한다.

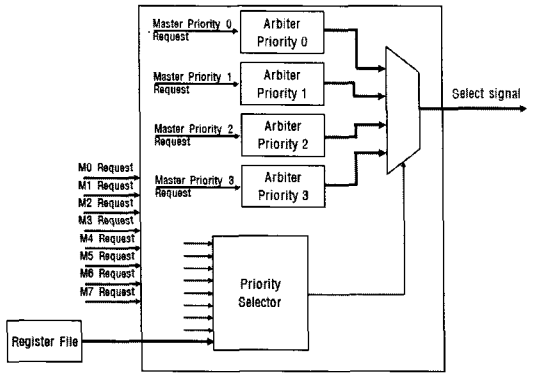

그림 6. 마스터선택제어기의 블록도
Fig. 6 Block diagram of Master Select Controller

각각 4개의 우선순위 아비터 또한 레지스터 파일로부터 마스터모듈들의 우선순위 정보를 제공받아, 각 아비터의 우선순위와 동일한 우선순위를 갖는 마스터 모듈들의 연결요청신호를 받는다. 각 아비터 내부는 라운드-로빈 방식에 의해 마지막으로 통신 연결을 하였던 마스터 모듈의 정보를 가지고 있다가 다음 연결 요청이 들어오면 그다음 순번에 해당하는 마스터 모듈에게 접근권한(Grant)를 부여한다. 즉 우선순위 0 아비터는 우선순위가 0인 마스터 모듈들만 라운드 로빈을 하게 되고, 우선순위 3 아비터는 우선순위가 3인 마스터 모듈들만 라운드 로빈을 하게 된다. 마스터 모듈들로부터 연결요청이 들어오면, 우선순위선택 모듈에 의해 연결요청을 한 마스터 모듈의 우선순위를 비교해 가장 높은 우선순위에 해당하는 아비터를 선택하고, 선택된 아비터의 라운드 로빈 순서에 해당하는 마스터 모듈을 선택한다.

표 1은 마스터 0번의 우선순위 값을 3, 마스터 1, 2, 3, 4번의 우선순위 값을 1로 설정하고, 하나의 슬레이브 모듈에 5개의 마스터 모듈이 계속해서 요청신호를 보냈을 때, 슬레이브 모듈과 통신하는 순서를 나타낸다. '↓'는 통신이 이루어진 것을 표시한 기호이다. 공유 버스의 경우 아비터의 라운드 로빈의 기능이 없기 때문에 두 개의 마스터가 계속해서 버스를 점유하는 반면, 다중 채널 버스는 우선순위가 높은 마스터 0번을 제외한 나머지 마스터들이 균일하게 슬레이브 모듈과 통신을 하게 되어, 특정 마스터모듈들이 슬레이브 모듈을 독점하는 것을 방지함으로써 효율적인 통신이 이루어지도록 지원한다.

표 1. 5개의 마스터가 슬레이브 인터페이스에 계속해서 접근할 경우의 처리순서
Table. 1 5 Masters continue to send Request signals to Slave Interface

Shard bus					Multichannel bus				
m0	m1	m2	m3	m4	m0	m1	m2	m3	m4
↓					↓				
	↓					↓			
↓					↓				
	↓						↓		
↓					↓				
	↓							↓	
↓					↓				
	↓								↓

3.3 레지스터 파일

그림 7은 레지스터 파일을 나타낸다. 레지스터 파일은 15번째 슬레이브 인터페이스와 연결되어 있으며, 주소 값에 의해 레지스터 파일이 선택된다. 마스터 모듈에 의해 레지스터 파일이 선택되면, 슬레이브 모듈과의 통신은 차단된다. 레지스터 파일의 내부에는 16비트 레지스터가 16개 존재하여, 16개 슬레이브 각각에 대한 8개의 마스터 모듈의 우선순위가 저장된다. 8개의 마스터 모듈의 우선순위는 2비트씩 지정이 되어 3 ~ 0 값으로 3이 가장 높은 우선순위, 0이 가장 낮은 우선순위를 나타낸다. 레지스터 파일 내부의 16개 레지스터 값은 사용자에 의해 각각 다르게 저장할 수 있다. 이는 각각의 슬레이브에 따라 먼저 처리해야 할 마스터 모듈의 우선순위를 높게 지정함으로써, 효율적인 통신 분배가 이루어지도록 지원한다.

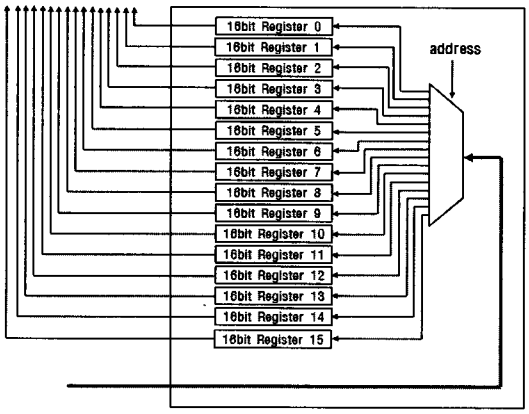

그림 7. 레지스터 파일 블록도
Fig. 7 Block diagram of Register File

Ⅳ. 실험결과

크로스바 스위치 버스의 검증을 위해 32비트 OpenRISC 프로세서, 디버그 인터페이스, VGA/LCD 제어기, AC97 제어기, UART, 메모리 인터페이스로 구성된 WISHBONE 프로토콜기반의 멀티미디어 SoC 플랫폼을 사용한다[3-7]. 검증은 두 가지의 테스트 프로그램을 이용하여 진행하였다. 첫 번째 테스트 프로그램은 프로세서가 메모리로부터 명령 및 데이터를 읽어와 VGA/LCD 컨트롤러에 명령 및 데이터를 전송하도록 하였으며, 두 번째 테스트 프로그램은 VGA/LCD 컨트롤러와, UART가 동시에 통신하는 프로그램이다. 그림 8은 첫 번째 테스트 프로그램의 이미지-디스플레이를 위해 멀티미디어 SoC 플랫폼에서 병렬통신이 이루어지는 모습을 나타낸다. 프로세서가 메모리로부터 명령을 읽어가는 것과 동시에 VGA/LCD 컨트롤러에 제어신호를 보내어, VGA/LCD 컨트롤러가 이미지 데이터를 읽어가는 동작을 나타낸다.

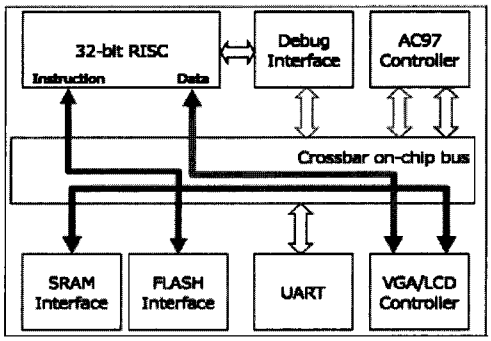

그림 8. 테스트 프로그램1에 의해 동작하는 멀티미디어 SoC 플랫폼의 블록도
Fig. 8 Block Diagram of Multimedia SoC Platfrom operation by test program1

그림 9는 두 번째 테스트 프로그램의 이미지-디스플레이 및 UART를 제어하기 위해 멀티미디어 SoC 플랫폼에서 병렬통신이 이루어지는 모습을 나타낸다.

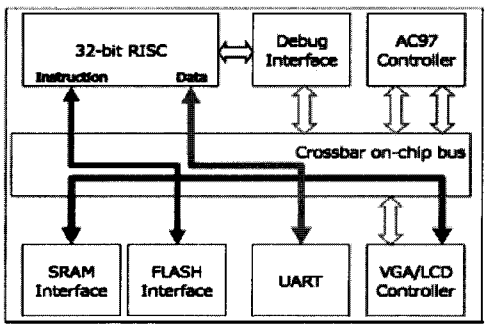

그림 9. 테스트 프로그램2에 의해 동작하는 멀티미디어 SoC 플랫폼의 블록도
Fig. 9 Block Diagram of Multimedia SoC Platfrom operation by test program2

프로세서가 플래시 메모리로부터 명령을 읽어가는 것과 동시에 UART에 출력할 메시지 데이터를 전송하고, VGA 컨트롤러는 SRAM으로 부터 이미지데이터를 읽어가는 동작을 나타낸다.

표 2는 두 가지의 테스트 프로그램을 이용하여 크로스바 온칩 버스를 내장한 SoC 플랫폼의 성능을 확인한 결과로써, Xillinx ISE와 Modelsim을 연동하여 기능 시뮬레이션을 수행하였으며, 표에 나타난 측정결과는 80MHz 클럭 주파수상에서 VGA/LCD 모듈이 설정 되었을 때 걸리는 시간과 VGA/LCD 및 UART 컨트롤 모듈이 설정 되었을 때 걸리는 시간을 공유버스와 크로스바 온칩 버스로 나누어 비교한 것이다.

표 2. 테스트 프로그램을 적용한 시뮬레이션결과
Table. 2 The computation result by test program

	공유버스 (ns)	크로스바 버스(ns)	차이 (ns)	효율 (%)
테스트 프로그램1	19,556	14,357	5,199	26.58
테스트 프로그램2	23,568	16,832	6,736	28.58

테스트 프로그램1을 통해 VGA/LCD 컨트롤러를 제어할 경우, 공유버스는 OpenRISC의 Instruction 부분과 Data부분의 두개의 마스터가 FLASH와 SRAM 메모리로부터 데이터를 가져올 때, Instruction부분의 마스터가 FLASH로부터 명령데이터를 가져온 이후, Data부분의 마스터가 SRAM으로부터 데이터를 가져오게 된다. 따라서 Instruction 마스터 부분과 Data부분의 마스터는 순차적으로 통신이 끝난 다음에 데이터를 가져올 수 있다. 하지만 크로스바 온칩 버스의 경우 OPenRISC의 Instruction과 Data 부분의 두 개의 마스터가 멀티채널을 통해 SRAM과 FLASH 메모리와 병렬로 통신을 하기 때문에, 공유버스에서 발생하는 병목현상을 해결하여 속도가 개선된다. 두 번째 프로그램의 경우 VGL/LCD 컨트롤러를 제어한 이후에 UART컨트롤러까지 제어하는 프로그램으로써, VGA/LCD 컨트롤러의 마스터 부분이 SRAM으로부터 이미지 데이터를 읽어오는 동안 UART 컨트롤러를 계속해서 OPenRISC는 멀티채널을 통해 제어하게 되어, 첫 번째 프로그램보다 더 좋은 성능을 나타낸다. 실험 결과 첫 번째 프로그램의 경우 기존의 공유버스에 비해 성능이 26.58%가 향상되었으며, 두 번째 프로그램의 경우 28.58%가 향상되었다.

표 3은 크로스바 온칩 버스를 내장한 SoC 플랫폼을 Xilinx사의 Virtex4 디바이스를 이용하여 FPGA에 구현한 결과를 나타낸다. Flip Flop은 7,565개(10%), LUT(Look Up Table)은 17,996개 (25%), I/O Buffer는 117(8%)로 총 1,331,705개의 게이트가 사용되었으며, 최대 48.66 MHz의 클록 주파수로 동작하였음을 확인하였다.

표 3. SoC의 플랫폼의 FPGA 구현 결과
Table. 3 Implementation Result of SoC Platform

디바이스	xc4vlx80-10ff1148
Flip Flops	7,565 (10%)
LUTs	17,996 (25%)
I/OBs	117(8%)
총 사용된 게이트 수	1,331,705
최대 동작 클록주파수	48.66 MHz
소비전력	1,009mW

V. 결 론

본 논문은 대부분의 SoC에 사용되는 공유버스의 단점인 병목현상을 줄이고 성능을 향상시키기 위하여 크로스바 스위치 온칩 버스 구조를 제안한다. 크로스바 스위치 온칩 버스 구조는 WISHBONE 프로토콜을 사용하며, 최대 8개의 마스터와 16개의 슬레이브 인터페이스를 지원한다. 마스터 모듈의 주소 값을 디코딩하여 연결요청 신호를 슬레이브 인터페이스로 전달하는 마스터 인터페이스와, 각 마스터 모듈로부터 전달된 연결요청 신호를 받아 슬레이브 모듈로 전달하는 슬레이브 인터페이스, 슬레이브 모듈에 각 마스터 모듈의 우선순위정보 값을 전달하는 레지스터 파일로 구성된다. 슬레이브 인터페이스는 우선순위 선택 제어 모듈과 라운드로빈 기능을 갖는 아비터가 각 우선순위에 맞게 4개 존재하여, 우선순위가 높은 마스터 모듈은 먼저 처리하고, 우선순위가 동일한 마스터 모듈들은 순서대로 처리해 전체적으로 효율적이고 균등한 통신이 이루어지도록 지원한다. 또한 각 마스터 인터페이스와 슬레이브

인터페이스 사이에는 멀티채널이 존재하여 병렬통신을 지원함으로써, 공유버스에서 발생하는 병목현상을 해결해 전체적인 통신성능을 향상 시켰다. 실험 검증을 위해 OPenRISC 프로세서 기반의 멀티미디어 SoC 플랫폼을 사용하였고, 테스트를 위해 VGA/LCD 컨트롤러와 UART 컨트롤러를 제어하는 테스트 프로그램을 적용한 결과, VGL/LCD 컨트롤러를 제어할 경우 크로스바 온칩 버스가 공유버스보다 26.58%의 성능향상을 보였으며, VGA/LCD 컨트롤러와 UART 컨트롤러 두개의 모듈에 대한 통신을 제어하는 경우에는 28.58%의 성능향상을 보였다. 따라서 제시한 크로스바 온칩 버스 구조가 공유버스의 병목현상을 제거함으로써 전체적인 통신 속도를 개선한다는 결론을 이끌었다.

참고문헌

[1] Zhihui Xiong, Sikun Li, Jihua Chen and Dawei Wang, "A Platform-based SoC Hardware/ Software Co-Design Environment", *The 8th International Conference on Computer Supported Cooperative Work in Design*, vol.2, pp 443-448, May 2004

[2] Sanghun Lee, Chanho Lee, "A High Performance SoC On-chip-bus with Multiple Channels and Routing Processes", *2006 IFIP International Conference on Very Large Scale Integration*, pp 86-91, Oct. 2006

[3] Jacob Gorban, *UART IP Core Specification, Rev. 0.6* August 11, 2002

[4] Damjan Lampret, *OpenRISC1200 IP Core Specification Rev. 0.7*, September 6, 2001

[5] Rudolf Usselmann, *AC97 Controller IP Core Specification Revision 1.2*, September 19, 2002

[6] Richard Herveille, *VGA/LCD core specification, Rev 2.0*, March 20, 2003

[7] Igor Mohor, *SoC Debug Interface, Rev. 3.0* April 14, 2004

저자소개

허정범(Jung-bum Heo)

2009년 한밭대학교 정보통신공학과 공학사
2009년~현재 한밭대학교 정보통신전문대학원 정보통신공학과 석사과정
※ 관심분야 : 임베디드 프로세서, SoC 플랫폼 설계, 하드웨어/소프트웨어 통합설계, 멀티미디어 코덱 설계

류광기(Kwang-ki Ryoo)

1986년 한양대학교 공과대학 전자공학과 공학사
1988년 한양대학교 대학원 전자공학과 공학석사
2000년 한양대학교 대학원 전자공학과 공학박사
1991년~1994년 육군사관학교 교수부 전자공학과 전임강사
2000년~2002년 한국전자통신연구원 집적회로설계 연구부 시스템IC 설계팀 선임연구원
2003년~현재 한밭대학교 정보통신공학과 부교수
※ 관심분야 : SoC 플랫폼 설계 및 검증, 하드웨어/소프트웨어 통합설계 및 검증, 멀티미디어 코덱 설계