마이크로컨트롤러를 위한 variable width system bus 제어기 구현

김 도 집, 이 문 기
연세대학교 전기전자공학과
전화 : 02-2123-4731 / e-mail : greendrm@spark.yonsei.ac.kr

Design and Implementation of variable width system bus controller for microcontroller

Do Jip Kim, Moon Key Lee
Dept. of Electrical & Electronic Eng., Yonsei University
E-mail : greendrm@spark.yonsei.ac.kr

Abstract

In this paper, Variable width system bus provides a variable data bus width from 8bit to 128bit.

This paper has designed variable width system bus controller for microcontroller to connect to peripherals.

I. 서론

현재 마이크로컨트롤러는 다양한 어플리케이션을 위해 동작하는 것들이 있으며 점점 더 많은 곳에서 활용되고 있다. 시스템 제어 및 다양한 서비스에 대한 처리능력을 갖추고 있다. 이에 대한 요구는 확장되어가고 있다. 마이크로컨트롤러는 외부 장치와 매우 밀접한 관계를 가지고 있다. 특히, 다양한 주변장치와의 효과적인 연결은 전체 성능과 확장성에 있어서 중요한 이슈로 작용하고 있다[1].

본 논문은 마이크로컨트롤러의 주변장치의 효과적인 연결과 확장성을 위하여 가변 대역을 갖는 데이터 버스 시스템을 위한 버스 제어기(VWSBC) 구현을 소개한다.

II. 마이크로컨트롤러를 위한 시스템 버스

마이크로컨트롤러가 고성능화 되면서 주변장치 역시 그에 따른 처리 능력이 요구된다. 일반적으로 버스 마스터, On-chip 메모리, 외부 메모리 인터페이스, DMA 등은 마이크로컨트롤러와 같은 고성능을 처리할 수 있다. 이와 달리 타이머, I^2C, 병렬 인터페이스, PIO 등은 고성능을 필요로 하지 않는다.

이를 위해 마이크로컨트롤러를 위한 시스템 버스를 설계함에 있어서 두 개의 버스 시스템으로 분리하였다[2,3,4]. 버스 마스터와 고성능의 주변장치와의 연결을 위한 AMBA의 AHB 버스 프로토콜을 사용하는 고성능 시스템 버스(High-performance System Bus; HSB)와 그 외의 일반적인 주변장치와의 연결을 위한 AMBA의 APB 버스 프로토콜을 사용하는 주변장치 시스템 버스(Peripheral System Bus;PSB)이다. 이 서로 다른 두 버스를 PSB 브릿지가 중계하게 된다[4].

[그림1]은 HSB와 PSB를 사용한 시스템의 한 구성 예이다. 하나의 MCU와 메모리 컨트롤러, 외부 메모리 버스 인터페이스가 HSB와 연결되어 있으며 PSB 버스엔 타이머, 인터럽트 제어기, UART가 연결되어 있다. 이 두 버스는 PSB 브릿지를 통해 연결된다.

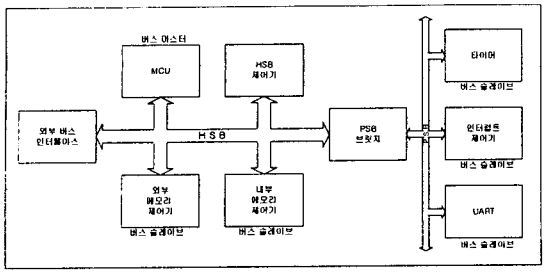


그림 1 HSB와 PSB 버스를 갖는 시스템 구성

III. Variable Width System Bus

3.1 VWSB의 정의

데이터 버스에 있어서 마이크로컨트롤러와 주변장치 간의 데이터 버스 대역의 불일치가 문제된다. 주변장치의 경우 마이크로컨트롤러의 데이터 버스 대역보다 작은 대역을 사용하는 경우가 많다. 또는 그 반대의 경우도 있다. 이러한 경우 각각의 주변장치의 데이터 버스 대역에 따른 별도의 버스를 구축해야한다.

본 논문에서는 이러한 문제점을 해결하고자 서로 다른 데이터 버스 대역에 대해 버스 시스템에서 사용자 정의 가능한 가변 대역 시스템 버스(Variable Width System Bus; VWSB)를 설계하였다. VWSB를 HSB 버스와 PSB 버스를 중계하는 PSB 브릿지에 포함시켜 버스 대역을 변환할 수 있도록 하였다.

VWSB는 HSB와 PSB 버스가 8, 16, 32, 64, 128비트의 서로 다른 버스 대역을 갖을 때 높은 대역의 데이터를 여러개의 데이터로 나눠 전송함으로써 상호 연결할 수 있도록 하였다. [그림2]은 HSB의 데이터 버스 대역이 64비트이고 PSB의 데이터 버스 대역이 32비트인 narrow 버스 슬레이브를 갖는 VWSBC의 한 구성 예이다. HSB의 64비트 데이터는 VWSBC를 통해 32비트 데이터로 변환되어 두 번에 걸쳐 전송이 이뤄지게 된다. 반대로 HSB가 64비트의 데이터를 읽는 경우엔 VWSBC를 통해 32비트 데이터를 두 번에 걸쳐 읽어와 64비트 데이터로 변환한다. [그림3]는 PSB가 64비트 데이터 버스 대역을, HSB가 32비트 데이터 버스 대역을 갖는 wide 버스 슬레이브를 갖는 VWSBC의 구성이다. 동작 방식은 [그림2]의 narrow 버스 슬레이브의 VWSBC와 같다.

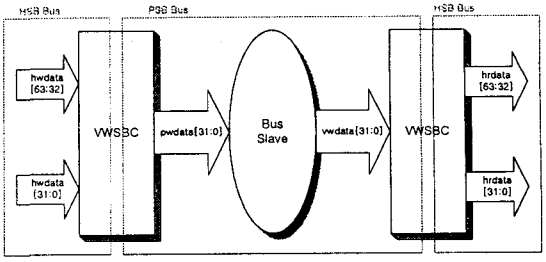


그림 2 Narrow salve with VWSBC

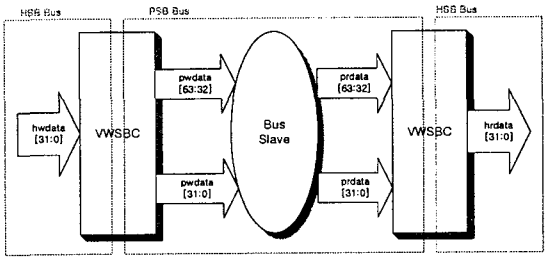


그림 3 Wide slave with VWSBC

HSB의 버스 대역과 PSB의 버스 대역의 설정은 VWSB 제어기(VWSB Controller; VWSBC) 내의 제어 상태 레지스터(Control Status Register; CSR)를 통해 설정한다. CSR를 이용함으로써 사용자 정의가 가능하다. 프로그램 수준에서 데이터 버스 대역을 설정하여 이용 가능한 주변장치에 따라 서로 다른 버스 대역을 이용할 수 있다. 이를 통해 다양한 데이터 버스 대역을 갖는 시스템을 구성할 수 있다. 각 대역에 대한 설정은 프로그램에서 VWSBC의 CSR 값을 설정하면 된다.

[그림4]은 VWSBC의 제어 상태 레지스터를 나타낸 것이다.

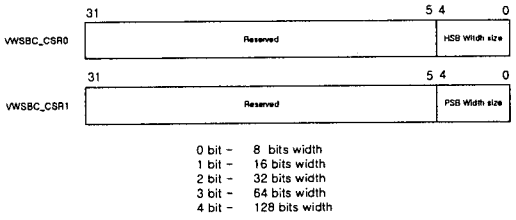


그림 4 VWSBC의 제어 상태 레지스터

[그림5]에서 보는 바와 같이 VWSBC는 데이터 버스 선택기, 버퍼, 주소 생성기, 디코더로 구성된다.

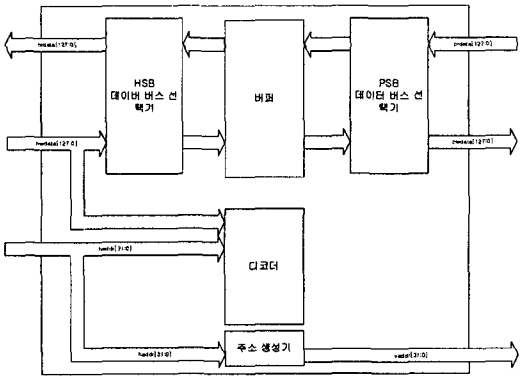
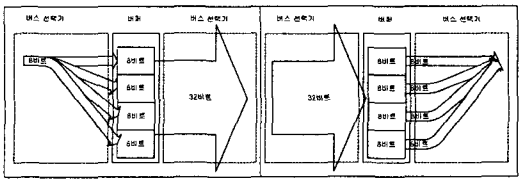


그림 5 VWSBC의 구조

데이터 버스 선택기는 VWSBC의 CSR에 설정된 값에 따라 디코더에 의해 데이터 버스의 비트를 선택한다. VWSBC는 최소 데이터 버스 크기가 8비트로, 버퍼는 8비트의 작은 버퍼들로 구성, 64개가 모여 128비트 크기의 버퍼를 갖게 된다. 쓰기와 읽기 버스가 분리가 되어 있으므로 두 개의 버퍼를 가지고 있다 총 256비트의 버퍼가 있게 된다. 데이터 버스 선택기는 [그림6]에서 보는 바와 같이 비트에 따라 데이터 버스를 선택하여 데이터를 버퍼에 정렬하게 된다. [그림6]은 버스 선택기의 동작을 결정하기 위하여 32비트 크기의 버퍼를 갖는 것으로 표현했지만 본 논문의 VWSBC는 128비트의 버퍼를 가지고 있다. narrow 버스 슬레이브[그림2]를 갖는 경우 PSB는 여러 개의 데이터를 나눠 전송하게 된다. 주소 생성기는 이때 필요한 주소를 생성하게 된다. 마지막으로 디코더는 VWSBC의 제어 상태 레지스터에 따라 각 VWSBC 내의 블록을 제어한다. 특히, 디코더는 데이터 버스 선택기에서 데이터 버스의 몇 비트를 사용할지를 결정하여 버퍼에 8비트 단위로 정렬시키도록 한다.



a. narrow->wide 데이터 버스 b. wide->narrow 데이터 버스

그림 6 버스 선택기와 버퍼의 구조

3.2 VWSBC의 타이밍

VWSB의 데이터 전송은 VWSB 버퍼 쓰기 위상(buffer write phase)과 VWSB 버퍼 읽기 위상(buffer read phase)으로 이뤄진다. 쓰기 위상은 버스 선택기에서 선택된 버스의 데이터가 버퍼에 쓰는데 걸리는 시간이다. 버퍼 읽기 위상은 데이터가 버스에 실리고 전송이 완료되기까지 걸리는 시간이다.

HSB는 기본적으로 HSB 주소 위상과 데이터 위상으로 이뤄지며 PSB는 PSB setup 위상과 enable 위상으로 이뤄진다[4].

1) VWSBC를 이용한 쓰기 동작

[그림7]은 HSB 데이터 버스 대역이 32비트, PSB 데이터 버스 대역이 16비트인 narrow 슬레이브의 쓰기 동작 타이밍이다. vwsel_in은 버퍼 쓰기 phase에서 버스 선택기의 제어 신호로 0x000f로 하위 32비트만을 사용한다는 의미이다. 버퍼 쓰기 phase에서 v_ale 시그널이 high로 주소 래치에 HSB의 주소를 저장하도록 한다. 32비트 데이터는 버퍼에 쓰여지게 된다. 다음 버퍼 읽기 phase에서 두 개의 16비트 데이터로 두 번에 걸쳐 실리게 된다. 이를 위해 vwsel_out 시그널에 의해 버스 선택기에서 0x0003의 하위 16비트를 읽고 다음으로 0x000C로 상위 16비트를 PSB 데이터 버스에 싣는다. 이때 각 16비트 데이터에 대한 어드레스는 vwsel_cnt에 의해 주소 래치에 저장된 주소를 이용한다. 데이터가 PSB 버스에 실리는 동안 HSB의 다른 데이터의 전송이 이뤄지는 것을 막기 위하여 vready를 두어 low일 때는 다음 데이터의 전송의 시작을 지연시키며 다음 전송이 이뤄질 준비가 된 시점에서 vready가 high가 되어 다음 데이터 전송을 시작한다.

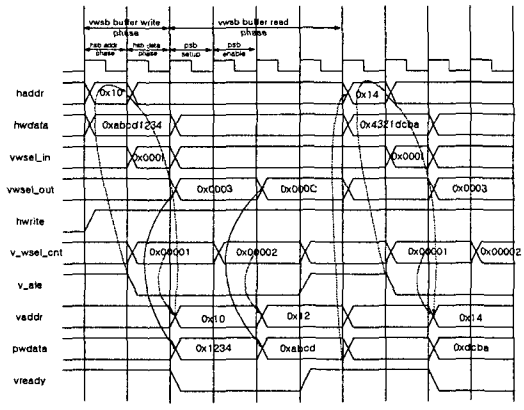


그림 7 쓰기 동작 (narrow slave)

[그림8]은 HSB 데이터 버스 대역이 32비트, PSB 데이터 버스 대역 64비트인 wide 슬레이브일 때 쓰기 동작의 타이밍이다. 이때는 HSB의 두 개의 데이터 전송이 이뤄진 후에야 PSB의 데이터 전송이 이뤄진다.

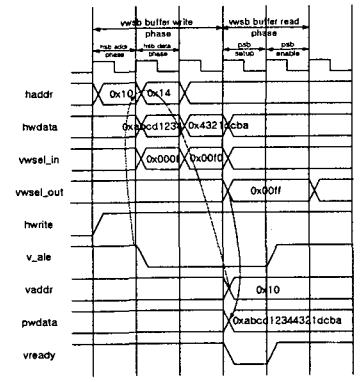


그림 8 쓰기 동작 (wide slave)

2) VWSBC를 이용한 읽기 동작

읽기 동작 역시 앞서 설명한 쓰기 동작과 크게 다르지 않다. 단, 데이터가 최종 실려야 하는 버스는 HSB의 데이터 버스이다.

[그림9]는 narrow 슬레이브로 HSB 데이터 버스 대역이 32비트, PSB 데이터 버스 대역이 16비트인 경우의 읽기 동자의 타이밍이다. HSB 주소 버스에 주소가 실리면 VWSBC 주소 래치를 통해 PSB 주소 버스에 주소가 실리게 되고 읽은 데이터는 VWSBC 버퍼에 저장된다. 두 번에 걸쳐 PSB에서 데이터를 읽게 되며 버퍼 쓰기 phase에서 HSB 실려 하나의 데이터 전송이 완료된다.

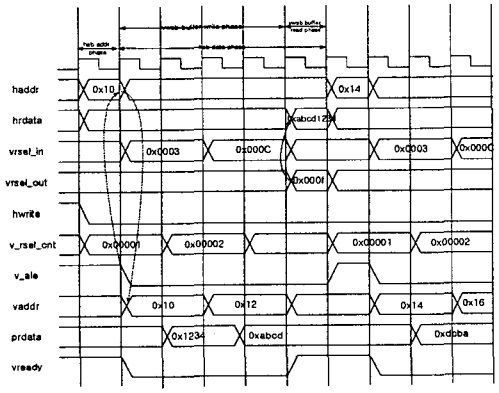


그림 9 읽기 동작 (narrow slave)

Ⅳ. VWSBC의 설계 및 검증

VWSBC의 설계는 기존의 설계된 HSB와 PSB 버스의 중계 역할을 하는 PSB 브릿지에 포함시키는 것으로 하였다. 내부 블록으로 추가할 수 있도록 추가적인 시그널을 추가하지 않고 VWSBC의 제어 상태 레지스터 값과 주소 그리고 데이터를 가지고 구현하도록 하였다. 그리고 AMBA의 AHB와 APB와 호환 가능한 자체 설계한 HSB와 PSB를 이용하도록 설계하였다.

검증은 HSB와 PSB 버스에서 쓰기, 읽기 동작이 정상적으로 이뤄지는지 테스트하였다. 0.35μm의 CMOS 표준 라이브러리를 이용해 합성하였으며 100Mhz로 동작한다.

표 1 VWSBC의 특징

특징	설명
동작 주파수	100Mhz @ 3.3v
공정	CMOS 0.35μm
버스 대역	8, 16, 32, 64, 128비트 데이터 버스 대역 지원
버스 프로토콜	AMBA Rev 2.0 지원(AHB, APB)
CSR	프로그램 가능한 제어 상태 레지스터

Ⅴ. 결론

마이크로컨트롤러와 다양한 주변장치는 시스템 버스를 통해 연결된다. 그러나 마이크로컨트롤러와 달리 주변장치는 여러 데이터 버스 대역을 갖는 경우가 있다. 가변 데이터 버스 대역을 지원함으로써 마이크로컨트롤러와 주변장치와의 확정성을 높일 수 있다.

본 논문은 서로 다른 데이터 버스 대역을 갖는 버스를 중계할 수 있는 VWSBC를 설계, 구현하였다. VWSB는 8, 16, 32, 64, 128 비트의 대역을 지원한다. 프로그램 가능한 제어 상태 레지스터를 사용함으로써 프로그램상에서 설정, 다양한 버스 대역을 사용할 수 있다. VWSBC는 AMBA 버스 프로토콜을 이용하여 자체 설계한 시스템 버스의 PSB bridge에 포함되어 동작을 검증하였으며 추가적인 버스 프로토콜이나 인터페이스 없이 시스템 버스에 적용할 수 있다. 본 버스는 100Mhz로 동작한다.

참고문헌(또는 Reference)

[1] 김영덕, "내장형 응용을 위한 인터럽트 제어기, DMA 제어기, 타이머 유닛 및 칩 선택 유닛의 VLSI 설계", Yonsei University. 1997

[2] David Flynn, "AMBA:Enabling reusable on-chip designs", IEEE Micro, 1997

[3] D W Flynn, "Modular bus design supports on-chip testability", IEE Colloquium: Systems Design for Testability, No. 1995/083, paper 2.

[4] "AMBA Specification Rev 2.0", http://www.arm.com