AMBA Platform을 기반으로 하는 SoC 상의 DMAC 설계

Implementaion of DMAC on SoC based on AMBA Platform

황인기*, 김정식**
InKi-Hwang, JungSik-Kim

Abstract - Because of the demands for high performance and high integrated system, the needs for optimal platform becomes more importance. Optimal platform can handle more data effectively with same resources. AMBA(Advanced Microprocessor Bus Architecture)™ defines on-chip communication standard for designing high performance embedded micro-controllers. It is consisted of AHB, ASB and APB. It can support fast implementation and reliability in system that is composed with reusable IPs. DMAC is one of master in system and generate master signals of AHB to communicate data from one slave(peripheral or memory) to another slave. It can reduce burden of CPU and increase system performance. We designed DMAC based on AMBA and it supports 13 channels. Each channel can be controlled by software program. It decides channel's priority using round-robin method. It can support P2P, P2M, M2P and P2P communication.

Key Words : AMBA, SoC, DMAC

1. 장 서 론

고성능 고집적 시스템의 요구가 증가됨에 따라, 동일한 Resource를 사용하여 보다 효율적인 Data communication을 가능하게 하는 bus architecture의 필요성이 강조되고 있다. AMBA (Advanced Microprocessor Bus Architecture)는 고성능 embedded-controller를 이용한 System을 위해 제안된 on-chip communication standard이다. AMBA는 AHB (Advanced High-performance Bus)와 APB (Advanced Peripheral Bus), 그리고 ASB (Advanced System Bus)로 구성되어 있으며, IP를 이용한 System 구현에 있어, 그 Backbone을 제공함으로써 사용자로 하여금 짧은 시간에 신뢰성 있는 시스템의 구현을 실현시켜준다.

DMAC (Direct Memory Access Controller)는 시스템의 Communication 효율을 증대시키고, CPU의 부담을 덜어주는 방법으로 사용된다. DMAC는 bus 의 마스터로써 source에서 destination으로의 communication에 필요한 제어 신호를 생성하고, Data communication 기능을 수행한다. 본 논문에서는 AMBA를 기반으로 하는 DMAC 의 구조와 설계 방법에 대하여 논한다.

구현된 DMAC는 13개 채널을 지원하고, AHB interface 규격에 맞도록 설계되었다. 각 채널 간 우선순위는 라운드 로빈 방식에 의해 결정된다. 채널의 방향, 전송 형태, 그리고 사용 가능/불가는 사용자의 요구에 따라 소프트웨어적으로 제어 가능하다. 구현된 DMAC 는 P2P (Peripheral), P2M (Memory), M2P, 그리고 M2M간 data communication을 지원한다.

저자 소개
* 황 인 기 : 한국전자통신연구원
** 김 정 식 : 한국전자통신연구원

2. 장 본 론

본 논문에서 구현된 DMAC 는 AMBA의 AHB의 규격을 따른다. AHB는 system backbone 기능을 수행한다. AHB는 Embedded processor 와 on-chip memory, off-chip memory controller, 그리고 peripheral 간의 효율적인 data communication을 위한 bus이다. 그림 1.은 AMBA를 이용한 system의 구현 예이다.

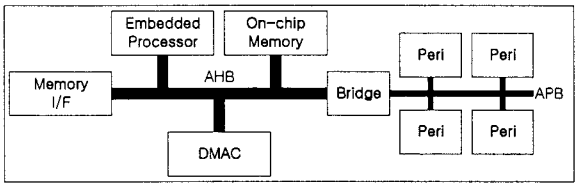
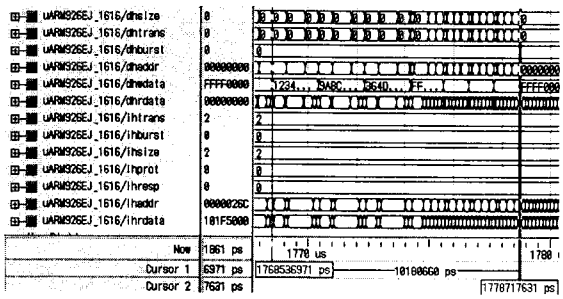


그림 1. AMBA를 이용한 System의 구조

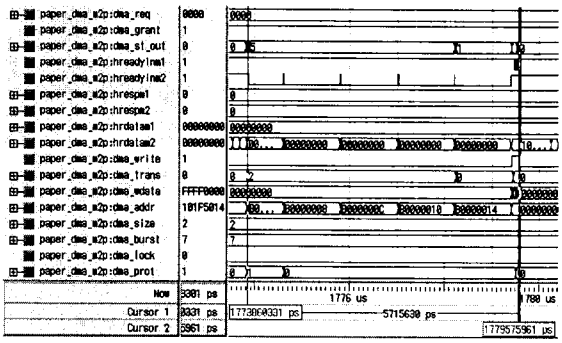
DMAC 는 peripheral과 memory, 또는 그 상호 간의 data communication을 효율적으로 수행하기 위해 구현된 블록이다. 동일한 source/destination에서의 data 입/출력과 같이 반복되는 작업을 수행하는 경우 CPU에서 처리하는 것보다 DMAC에서 처리하는 것이 보다 효율적이다. CPU는 data를 읽거나 쓰는 동작을 수행하기 위해 instruction read, fetch, decoding, operation 과 같은 과정을 수행하여야 하며, 동일한 instruction(동일 source/destination 에서의 data 입/출력)을

수행하는 경우에도 위와 같은 과정을 반복하여야 하므로 효율성이 떨어진다. DMAC는 이러한 data의 communication 효율을 높이며, 또한 CPU의 부담을 덜어주는 기능을 수행한다.

그림2는 M2P의 data communication을 수행하는 경우의 CPU/DMAC의 동작 비교도 이다.(burst mode로 동작하지 않는 경우)



(a) CPU의 M2P data communication



(b) DMAC의 M2P data communication

그림 2. CPU/DMAC의 비교도

그림 2에서 보면 동일한 동작을 수행함에 있어 DMAC에 의한 동작이 CPU에 의한 동작 보다 2배 빠른 것을 볼 수 있다.

DMAC는 AHB bus 에 대한 master 이며 동시에 slave의 특성을 가지고 있다. slave interface를 통하여 DMAC의 현 register 정보를 Embedded processor에 전달하고, 또한 사용자의 요구에 따라 register의 값을 입력받는다. Master interface를 통해선 communication에 필요한 제어 신호와 data를 입/출력 하게 된다. 그림 3은 구현된 DMAC의 내부 블록도이다. Slave interface 모듈은 Embedded processor에서 전달된 data와 제어 신호에 대한 응답 신호와 수신 결과를 Processor 쪽으로 전달하는 기능을 수행한다.

Register interface 모듈은 사용자 프로그램에 따라 DMAC를 제어하기 위한 정보들을 수신하거나, 그에 대한 결과 또는 DMAC의 현 상태 정보 등을 processor 쪽으로 전달하는 기능을 수행한다. 수신되는 제어 정보는 각 채널에 대한 정보, DMA 동작 가능 여부, 소프트웨어에 의한 DMA 채널 제어 정보, 인터럽트 제어 정보, 각 채널 별 전송 data 형태와 크기 등으로 구성된다.

Master interface 모듈은 DMAC가 bus 에 대한 master로 동작할 때, AHB 규격에 맞는 interface 신호의 전달 통로 역할을 수행한다. 또한 interrupt 제어 모듈을 포함하고 있어서, 전송의 종료, 전송 에러 등의 interrupt 정보를 생성하여 register interface 모듈에 전달하고, processor로 interrupt 신호를 송신하는 기능을 수행한다.

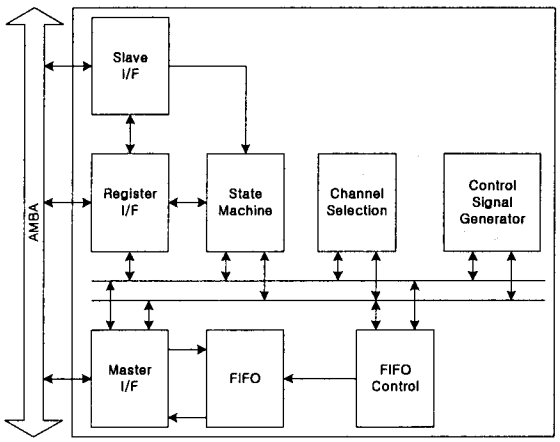


그림 3. DMAC 블록도

State machine 모듈은 register interface 모듈에서 전달된 각 채널 별 address, data 형태 및 크기, 전송 형태 등의 정보와 Channel selection 모듈에서 전달된 현 상태 선택 채널 정보를 바탕으로 DMAC의 동작 상태를 결정하는 기능을 수행한다. 그림 4는 State machine 모듈의 동작 천이 flow를 나타낸다. State machine에서 생성된 상태 정보는 control signal generator 모듈과 register interface 모듈로 전달되어 AHB master 신호 생성과 DMAC 상태 정보 생성 기능을 수행할 수 있도록 한다.

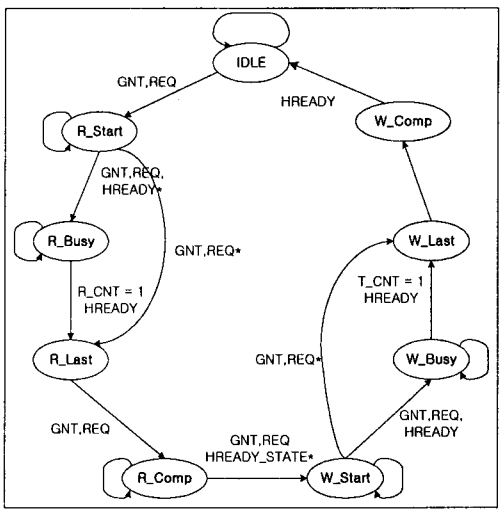


그림 4. State Machine 천이도

Channel selection 모듈은 각 채널의 우선순위를 결정하는

기능을 수행한다. 임의의 한 채널에서 DMA 요구가 발생하였을 경우엔 해당 채널의 data communication동작을 수행하지만, 두개 이상의 채널에서 동시에 DMA 요구가 발생할 경우 해당 채널들의 현 상태의 우선순위에 따라 한 채널에 우선권을 설정하는 기능을 수행한다. 구현된 DMAC에서는 라운드 로빈 방식을 이용하여 우선순위 결정 기능을 구현하였다. 라운드 로빈 방식은 초기에 각 채널의 우선순위를 나열해 놓고 선택되어 동작을 수행한 채널에 대한 우선순위를 최하위로 설정하는 방법이다. 예를 들어 현 상태의 우선순위가 채널 1,2,3,4 의 순이고 채널 2에서 요구가 발생하여 동작을 수행한 경우 다음 상태의 우선순위는 3,4,1,2 로 조정된다.

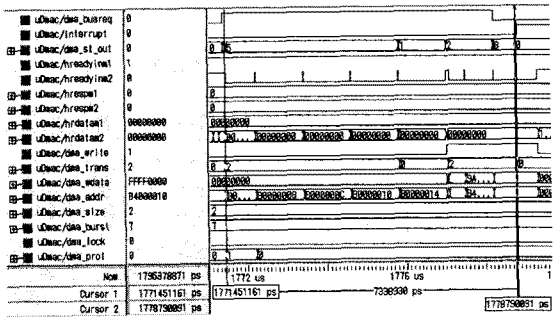
Control signal generator 모듈은 state machine 모듈에서 전달된 현 DMAC 정보와 register interface에서 전달된 각 채널 설정에 대한 정보, 그리고 channel selection 모듈에서 전달된 현 상태의 선택 채널의 정보를 바탕으로 AHB master interface에 맞는 제어 신호를 생성하는 기능을 수행한다. 이 때, 생성되는 제어 신호는 address, size, trans, prot, burst, ready 등이 있다. AHB 의 동작은 32bit bus architecture를 가지고 있어 control signal generator 모듈에서 생성되는 address 신호는 +4 씩 증가(memory) 하거나 고정(FIFO)된 형태로 생성된다. 또한 trans 신호는 전달 data의 정보에 따라 NON-SEQ/SEQ/IDLE/BUSY의 형태로 생성된다.

FIFO control 모듈은 state machine에서 전달되는 현 상태 정보와 master interface를 통해 전달되는 현 bus 상태 정보를 바탕으로 FIFO read/write 제어 신호와 pointer 정보를 발생하는 기능을 수행한다. 발생한 pointer 정보는 interrupt 제어 모듈로 전달되어 전송 완료 시 전송 완료를 나타내는 interrupt 신호를 생성하게 한다.

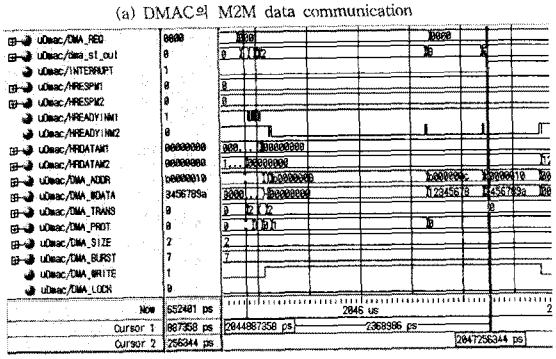
FIFO는 source에서 destination으로 전달되는 data 의 임시 저장소 역할을 수행한다. AHB 규격 상 동시 전달될 수 있는 최대 data 량이 1kbyte 이므로 FIFO의 크기는 1kbyte로 구성되어 있다.

구현된 DMAC 는 모두 13개 채널의 지원이 가능하게 설계되었으며, 채널 간 우선권 결정은 라운드 로빈 방식을 이용하여 결정한다. 사용자의 정의에 따라 각 채널의 정보를 임의로 설정할 수 있도록 설계되었으며, 또한 각 채널의 동작 가능 여부도 사용자의 정의에 따라 설정이 가능하다. AHB 규격에 맞는 master/salve port를 가지고 있으며, 32bit data 전송을 기본으로 한다. P2P, P2M, M2P, 그리고 M2M 의 data communication을 지원하며, 동시에 전달될 수 있는 최대 data의 량은 1kbyte로 제한되어 있다.

이와 같이 설계된 DMAC를 단일 AHB bus 구조를 갖는 AMBA platform 기반의 시스템과, 다중 AHB bus 구조를 갖는 AMBA platform 기반의 시스템에서 그 기능을 검증하여 보았다. 그림 5는 다중 AHB 구조를 갖는 시스템에서 P2M과 M2M의 DMAC 동작을 나타낸다. 그림에서 보는 바와 같이 AHB 규격에 맞는 제어 신호들이 생성되어, data 의 communication이 올바르게 진행되는 것을 볼 수 있다.



(a) DMAC의 M2M data communication



(b) DMAC의 P2M data communication

그림 5. DMAC의 M2M/P2M data communication

3. 장 결 론

본 논문에서는 AMBA platform을 기반으로 하는 시스템 상에서 구현한 DMAC의 구조와 동작에 대하여 논하였다. 구현된 DMAC는 AHB 규격에 맞춰 master/slave port를 가지며, 모두 13개의 채널을 지원한다. 각 채널의 우선순위는 라운드 로빈 방식을 이용하여 결정된다. P2P, P2M, M2P 그리고 M2M의 data communication을 지원하며, 기본적으로 DWORD 단위로 동작한다. 동시에 전송 가능한 최대 data의 크기는 1kbyte이며, 이는 AMBA 규격에 따른 것이다.

구현된 DMAC는 ARM922T, ARM926EJS를 embedded processor로 사용하여 AMBA platform을 기반으로 구현된 system 상에서 검증하였다.

참 고 문 헌

[1] AMBA™ Specification(Rev 2.0), May 1999