TLM 방법을 이용한 다양한 중재 방식의 특성 비교

이국표* · 고시영**

Characteristic comparison of various arbitration policies using TLM method

Kook-pyo Lee* · Si-Young Koh**

요 약

SoC(System on a Chip)는 버스 아키텍처 내에 여러 개의 마스터와 슬레이브, 아비터 그리고 디코더로 구성되어 있다. 마스터는 CPU, DMA, DSP 등과 같이 데이터 트랜잭션을 발생시키는 블록이고, 슬레이브는 SRAM, SDRAM, 레지스터 등과 같이 데이터 트랜잭션에 응답하는 블록이다. 또한 아비터는 마스터가 동시간대에 버스를 이용할 수 없기 때문에 이를 중재하는 역할을 수행하는데, 어떠한 중재 방식을 선택하는가에따라 SoC의 성능이 크게 바뀔 수 있다. 본 논문에서 우리는 아비터에 대해 TLM(Tranction Level Model) 방법을 이용하여 다양한 중재 방식의 특성을 비교하였다. 일반적으로 사용되는 중재 방식으로는 fixed priority 방식, round-robin 방식, TDMA 방식, Lottery bus 방식 등이 있는데, 이 중재 방식들의 장점과 단점을 분석하였다.

ABSTRACT

SoC(System on a Chip) has several masters, slaves, arbiter and decoder in bus architecture. Master initiates the data transactions like CPU, DMA and DSP and slave responses the data transactions like SRAM, SDRAM and register. Furthermore, as multiple masters can't use a bus concurrently, arbiter plays an role in bus arbitration. In compliance with the selection of arbitration method, SoC performance can be changed definitely. In this study, we compare the characteristics of various arbitration policies using TLM(Transaction Level Model) method. Fixed priority, round-robin, TDMA and Lottery bus policies are used in general arbitration method. We analyze the merit and demerit of these arbitration policies.

키워드

SoC, arbiter, Fixed priority, Round-robin, TDMA, Lottery bus

* 영진전문대학 전자정보통신계열
** 경일대학교 전자정보통신공학부(교신저자)

접수일자 2009. 02. 16
심사완료일자 2009. 03. 10

Ⅰ. 서 론

반도체 공정 기술의 발전으로 전자회로의 시스템화, 소형화가 가능해짐에 따라 소비자들은 점점 더 좋은 성능을 가지는 제품을 요구하고 있다. 이에 발맞춰 엔지니어들은 여러 다른 기능을 하는 칩들을 하나로 집적시키는 SoC(System on a Chip)를 연구하고 있다.[1] SoC는 단일버스에 여러 개의 마스터와 슬레이브, 아비터, 디코더로 구성되어 있는데, 마스터는 CPU, DMA, DSP 등과 같이 데이터 트랜잭션을 발생시키는 블록이고, 슬레이브는 SRAM, SDRAM, 레지스터 등과 같이 데이터 트랜잭션에 응답하는 블록이다. 또한 아비터는 마스터들이 동시간대에 버스를 이용할 수 없기 때문에 이를 중재하는 역할을 수행한다. 마지막으로 디코더는 마스터로부터 나오는 어드레스의 상위 비트를 가지고 적절한 슬레이브를 선택해주는 역할을 수행한다.[2] 특히, 아비터는 다양한 중재 방식들이 있는데, 각 중재 방식에 따라 시스템 자체의 성능이 바뀔 수 있으며, 시스템의 역할에 따라 각기 사용되는 중재 방식들이 존재한다.[3]

이에 본 논문에서는 TLM(Transaction Level Model) 방법[4]을 적용하여 다양한 중재 방식의 특성을 비교·분석을 하였다.

Ⅱ. 모 델

버스(Bus)는 CPU 또는 DMA와 메모리 및 입출력 장치 등을 연결하는 통신 채널을 의미한다. 실제 칩에서는 와이어로 이루어져 있으며 논리적인 의미에서는 마스터와 슬레이브 사이에 신호 또는 데이터 전송에 대한 프로토콜을 말한다. 마스터는 CPU나 DMA와 같은 버스의 사용에 주도적인 역할을 하는 장치를 말하며, 슬레이브는 메모리와 같이 마스터로부터의 명령에 해당하는 데이터를 저장하고 마스터의 명령에 따라 데이터를 전송하는 역할을 수행한다.

일반적인 버스 아키텍처의 경우는 단일 버스에 여러 개의 마스터와 슬레이브 그리고 아비터, 디코더로 구성되어 있다. 버스의 개수가 오직 하나이기 때문에 데이터 전송을 수행할 때, 동시간대에 마스터들이 버스를 이용할 수 없다. 그렇기 때문에 여러 마스터들이 버스를 이용하기 위해 서로 경쟁을 하고 아비터는 이를 중재하는 역할을 수행한다.

그림 1은 일반적인 버스 아키텍처의 데이터 트랜잭션을 보이고 있다. 마스터 M1과 M2는 각각 동시에 버스 사용 요청(req.)을 할 수 있다. 또한 요청한 신호가 아비터 AB에게 전송되면, 아비터 AB는 고유의 중재방식에 따라 선택된 마스터에게 버스 사용을 허가(grant)한다.

사용 허가를 받은 마스터는 해당 슬레이브 S1 또는 S2에 데이터 전송을 하고, S1 또는 S2는 wait 신호와 resp 신호를 통해 마스터와 아비터에게 데이터 처리에 관한 응답을 한다.

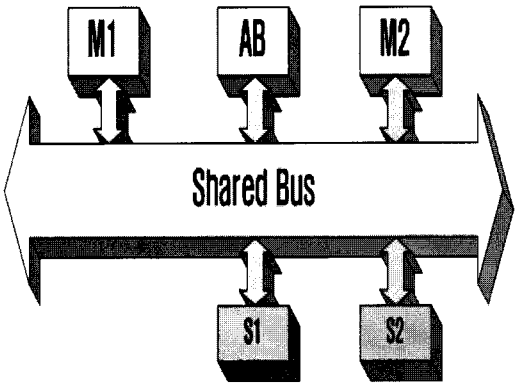

그림 1. 일반적인 버스 아키텍처의 데이터 트랜잭션
Fig. 1. Data transaction flow in general bus architecture.

성능분석을 위해 C++로 자체개발한 AMBA[5]를 사용하고 TLM(Transaction Level Model) 방법을 적용하였다.

마스터에서 발생하는 데이터는 싱글 데이터와 버스트 데이터가 있으며, 버스트 데이터 길이는 4, 8, 16까지 지원한다. 그리고 idle 사이클 지연 후 새로운 데이터를 발생시키는데, 버스트 데이터 길이와 idle 사이클에 대해서 랜덤 함수를 이용하였다.

Ⅲ. 결과 및 토의

3.1. Fixed priority 방식

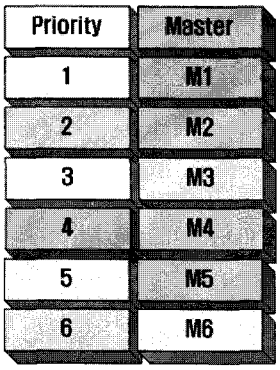

그림 2. Fixed priority 방식
Fig. 2. Fixed priority method.

Fixed priority 방식은 가장 일반적으로 사용되고 있는 중재 방식이다. 그림 2의 6개의 마스터 중 마스터 M1이 우선순위가 가장 높고 마스터 M6이 우선순위가 가장 낮다. 만약, 마스터 6개가 동시에 버스를 사용하려고 할 경우 각 마스터는 아비터에게 버스 요청을 한다. 이때, 아비터는 마스터 M1에게 버스 점유권을 줄 것이다. Fixed priority 방식의 장점은 중요한 데이터 처리가 필요한 마스터의 우선순위를 높여줌으로써 중요한 마스터가 손쉽게 버스를 이용할 수 있게 할 수 있다. 그러나 중요하지 않은 마스터의 경우엔 버스 점유권을 얻기가 상당히 힘이 들게 된다. 결국, 우선순위가 낮은 마스터의 경우 스타베이션(Starvation) 현상이 발생하게 된다.

그림 3은 TLM(Transaction Level Model) 방법을 이용한 결과를 보여주고 있다. 슬레이브는 데이터 레이턴시가 길고, 가변적인 SDRAM 컨트롤러로 하였으며, 총 마스터의 개수는 6개로 정하여 1,000,000 사이클까지 시뮬레이션을 수행하였다. 시뮬레이션 결과 우선순위가 가장 높은 마스터 M1의 데이터 트랜잭션은 150,438 사이클로 가장 높았으며, 마스터 M2는 135,308 사이클, M3은 112,519 사이클, M4는 73,819 사이클, M5는 30,341 사이클, M6은 6,078 사이클 순으로 데이터 전송을 수행하였다. 이는 마스터의 우선순위를 마스터 M1, M2, M3, M4, M5, M6 순으로 주었기 때문이며, 마스터 M1을 중요하게 생각해서 우선순위를 높인 결과 마스터 M1의 데이터 트랜잭션이 큰 반면에 마스터 M6의 데이터 전송은 거의 이루어지지 않고 스타베이션 현상이 발생하였다. 결국, fixed priority 방식의 결정적 문제점인 스타베이션 현상이 TLM 분석에도 발생하였음을 알 수 있었다.

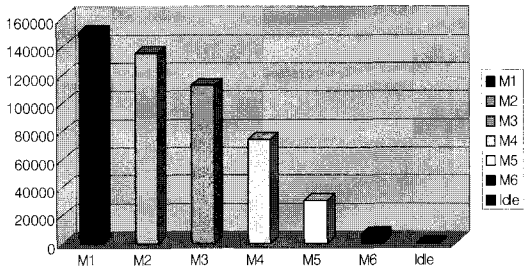

그림 3. Fixed priority 방식의 데이터 트랜잭션 사이클
Fig. 3. Data Transaction cycle in fixed priority.

3.2. Round-robin 방식

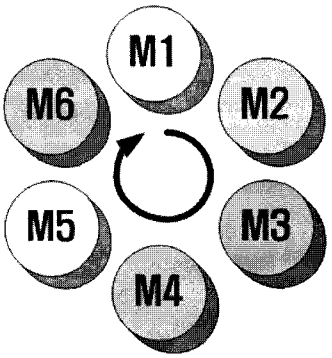

그림 4. Round-robin 방식
Fig. 4. Round-robin method.

Round-robin 방식은 fixed priority 방식과 함께 가장 일반적으로 알려진 방식이다. Round-robin 방식은 스타베이션 현상을 방지하기 위해 fixed priority 방식을 보완한 중재 방식이다. 즉, 모든 마스터에게 골고루 버스 점유권을 주게 된다면, 스타베이션 현상은 발생하지 않게 된다. 그러나 그림 4에서 마스터 M1, M2, M3, M4, M5, M6이 각각 순서대로 버스 점유권을 받게 된다고 가정할 때, 마

스터 M1이 다른 마스터보다 중요한 데이터 전송이라면 마스터 M1이 데이터 전송을 마치고 또 다른 중요한 데이터 전송을 해야 할 경우, 나머지 마스터 M2, M3, M4, M5, M6의 데이터 전송을 끝낼 동안에 기다려야 한다. 결국 round-robin 방식은 중요한 마스터의 데이터 처리 시간이 늦어질 수 있다는 단점이 있다.

그림 5는 TLM(Transaction Level Model) 방법을 이용한 결과를 보여주고 있다. 시뮬레이션 결과 마스터 M1, M2, M3, M4, M5, M6 모두 약 83,000-86,000 사이클 정도의 비슷한 값을 보였다. 이는 round-robin 방식이 스타베이션 현상을 방지하기 위해 모든 마스터에게 균일하게 버스 점유권을 부여했기 때문에 발생한 결과이다.

결국, round-robin 방식의 경우는 각 마스터에게 균등하게 버스 점유권이 돌아가기 때문에 각 마스터의 버스 요청 사이클이 수십 사이클로 안정적이라 할 수 있다. 그러나 우선순위가 고려되지 않기 때문에 중요한 마스터의 데이터 처리를 신속하게 수행할 수 없는 단점이 있다.

Fixed priority 방식과 round-robin 방식은 가장 먼저 제안된 방식이다. 그러나 TLM 분석 결과에도 알 수 있듯이 fixed priority 방식의 결정적인 문제점은 스타베이션 현상이다. 또한 이 문제점을 해결하기 위해 제안된 방식이 round-robin 방식인데, 결정적으로 스타베이션 현상은 방지하였지만, 중요한 데이터 처리를 신속하게 수행할 수 없었다. 이 문제점을 동시에 해결하고자 제안된 방식이 TDMA 방식이다.

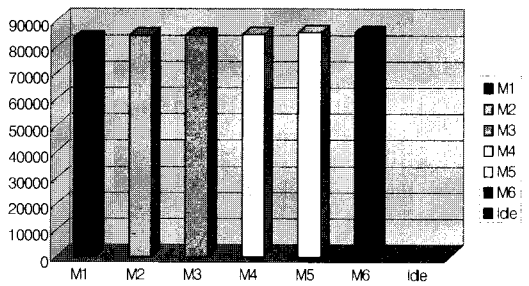

그림 5. Round-robin 방식의 데이터 트랜잭션 사이클
Fig. 5. Data Transaction cycle in round-robin.

3.3. TDMA 방식

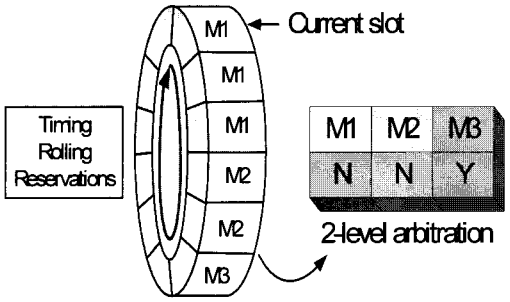

그림 6. TDMA 방식
Fig. 6. TDMA method.

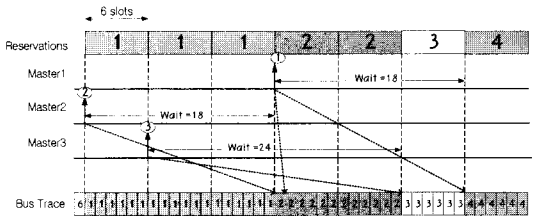

그림 7. TDMA 방식의 대기시간
Fig. 7. Waiting time of TDMA method.

TDMA 방식은 Time Division Multiple Access의 줄임말로 일반적으로 사용되고 있는 fixed priority 방식과 round-robin 방식을 보완하기 위해 제안된 방식이다. 그림 6은 TDMA 방식을 설명하고 있다. 이 방식은 예약된 마스터가 적힌 각 슬롯이 Timing Rolling을 통해 그 시간에 해당하는 마스터를 선택해 버스 점유권을 제공한다. Rolling의 단일 회전에서 한 슬롯 이상을 가지고 있는 마스터의 경우(마스터 M1, M2)는 많은 시간동안 버스를 사용할 수 있다. 만약 현재 슬롯에 관계된 마스터의 뚜렷한 요청이 있다면, 버스 점유권이 주어지고, Timing Rolling은 한 슬롯을 회전한다. 반면, 슬롯이 사용되지 않는다면, 대역폭의 낭비를 이끌 수 있다. 이 문제를 해결하기 위해, 2 순위 중재가 사용된다. 그러나 TDMA 방식은 그림7과 같이 좋은 경우엔 마스터의 대기시간이 1사이클로 이상적이 될 수도 있지만, 잘못된 경우엔 마스터의 대기시간이 수십 사이클까지 길어질 수도 있는 단점이 있다.

그림 8은 TDMA 방식[6]의 데이터 트랜잭션을 보여주고 있다. 마스터 M1에서 마스터 M6의 슬롯(slot)수를 각각 4, 1, 1, 1, 1, 1로 정하였으나, 각각의 데이터 트랜잭션은 마스터 M1이 132,276 사이클로 가장 높았으며, 마스터 M2는 102,611 사이클, M3은 82,338 사이클, M4는 82,105 사이클, M5는 77,821 사이클, M6은 29,173 사이클로 슬롯수와 정확하게 일치하지 않았다. 특히 마스터 M2에서 마스터 M6의 슬롯수가 같았지만 데이터 트랜잭션이 서로 크게 다름을 알 수 있다. 이는 현재의 슬롯에서 마스터가 버스 요청을 하지 않을 경우, 2순위 중재(2-level arbitration)에 의해서 다른 마스터가 버스 점유권을 받기 때문이다.

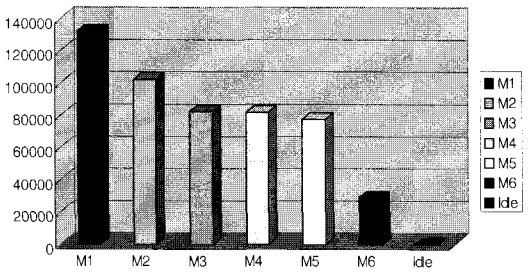

그림 8. TDMA 방식의 데이터 트랜잭션 사이클
Fig. 8. Data Transaction cycle in TDMA.

3.4. Lottery bus 방식

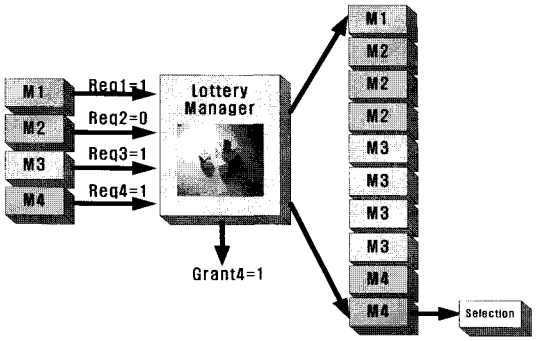

그림 9. Lottery bus 방식
Fig. 9. Lottery bus method.

Lottery bus 방식[7]은 TDMA 방식의 단점을 보완하기 위해 제안된 방식이다.

이 방식에는 Lottery 매니저를 포함하고 있는데, Lottery 매니저는 하나 또는 여러 마스터들의 버스 요청을 축적하고 있다. 각 마스터들에게는 티켓이 확률적으로 할당되어 있으며, 매니저는 무작위로 버스 점유권을 위해 경쟁하는 각 마스터들 중 하나의 마스터에게 버스 점유권을 제공한다. 결국, 이 경쟁에서 유리한 마스터는 티켓확률을 많이 할당 받은 마스터라 할 수 있다. 하지만, 이럴 경우 발생할 수 있는 마스터의 버스 점유 독점을 방지하기 위해 한 마스터가 이용할 수 있는 데이터 전송 크기에 제한을 두었으며, idle 사이클을 최소화하기 위해 Lottery 매니저 동작은 실제 데이터 전송과 함께 파이프라인 전송을 하는 특징을 가지고 있다.

$$P(M_i) = \frac{r_i \cdot t_i}{\sum_{j=1}^{n} r_j \cdot t_j} \tag{1}$$

식 (1)은 마스터가 버스 점유권을 얻을 수 있는 확률이다. Mi는 각 마스터들을 말하며, ti은 각 마스터에 의한 티켓 수이다. 어떤 버스 사이클에서 미결정의 버스 요청을 Boolean 변수 ri(i=1,2,3,...,n)로 표현한다. 만약, ri=0 이라면 마스터 Mi가 미결정 요청을 가지고 있다. 마스터들은 무작위로 버스 점유권을 받을 수 있는데, 더 많은 티켓 수를 가져야 버스 점유권을 받을 수 있는 확률이 높다.

그림 10은 Lottery bus 방식의 데이터 트랜잭션을 보여주고 있다. 마스터 M1에서 마스터 M6의 티켓확률을 4/9, 1/9, 1/9, 1/9, 1/9, 1/9로 정하였으나, 각각의 데이터 트랜잭션은 마스터 M1은 116,017 사이클, M2는 102,582 사이클, M3은 87,043 사이클, M4는 75,341 사이클, M5는 65,533 사이클, M6은 61,440 사이클이 소요되었는데, TDMA 방식과 마찬가지로 2순위 중재 때문에 버스점유 확률과 데이터 트랜잭션이 정확하게 일치하지 않았음을 알 수 있다.

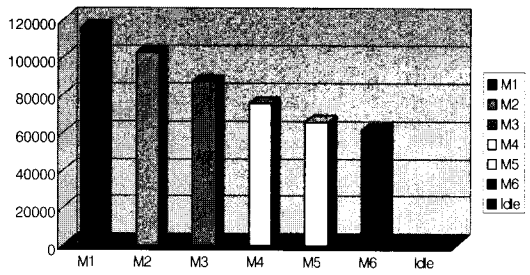

그림 10. Lottery bus 방식의 데이터 트랜잭션 사이클
Fig. 10. Data Transaction cycle in Lottery bus.

Ⅳ. 결 론

본 논문에서는 TLM 방법을 이용하여 아비터의 다양한 중재방식의 특성을 비교·분석하였다.

시뮬레이션 분석 결과, fixed priority 방식에서는 마스터의 버스 독점으로 인한 우선순위가 낮은 마스터의 스타베이션 현상이 발생하였다. 이를 보완하고자 제안된 round-robin 방식에서는 스타베이션 현상을 방지할 수 있었지만, 중요한 마스터의 데이터 처리를 신속하게 수행할 수 없는 문제점을 보였다. 또한 TDMA 방식과 Lottery bus 방식은 스타베이션 현상 방지와 중요한 데이터 처리를 신속하게 수행할 수 있었지만, 슬롯수에 따른 데이터의 처리의 정확성이 떨어지는 문제점을 보였다.

현재까지 사용되고 있는 버스중재방식의 장단점을 살펴봄으로서, SoC 칩 사용자 또는 애플리케이션 엔지니어가 칩이 사용되는 환경에 따라 최적의 버스 중재방식을 선택하는데 큰 도움이 되리라 생각된다. 뿐만 아니라, 앞으로의 중재 방식은 본 논문에서 다룬 fixed priority 방식, round-robin 방식, TDMA 방식, Lottery bus 방식의 장점을 살리고, 문제점을 해결하는 방식으로 발전할 것으로 생각되며, 이에 본 논문이 중요한 자료로 사용될 것이다.

참고문헌

[1] E. Salminen, V. Lahtinen, K. Kuusilinna and T. Hamalainen, "Overview of bus-based system-on-chip interconnections," in Proc. IEEE Int.Symp. Circuits Syst., pp.II-372-II-375, 2002.

[2] K. Lahiri, A. Raghunathan and S. Dey, "Design Space Exploration for Optimizing On-Chip Communication Architectures," IEEE Trans. Computer-Aided Design, vol.23, pp.952.961, June. 2004.

[3] A. B. Kovaleski, "High-Speed Bus Arbiter for Multiprocessor," IEEE Proc., Vol. 130, Pr, E, No.2, March 1983.

[4] 이국표, 윤영섭,"마스터와 슬레이브에 따른 싱글버스와 다중버스 토폴로지의 성능분석," 전자공학회논문지, 제45권 SD편 제9호, pp.96-102, 2008.

[5] AMBA TM Specification(AHB) (Rev 2.0), ARM Ltd, May 1999.

[6] Y. Xu, L. Li, Ming-lun Gao, B. Zhand, Zhao-yu jiand, Gao-ming Du and W.Zhang, "An Adaptive Dynamic Arbiter for Multi-Processor SoC," Solid-State and Integrated Circuit Technology International Conf., pp.1993-1996, 2006.

[7] K. Lahiri, A. Raghunathan and G. Lakshminarayana, "The LOTTERYBUS On-Chip Communication Architecture," IEEE Trans. VLSI Systems, vol.14, no.6, 2006.

저자소개

이 국 표(Kookpyo Lee)

한국해양정보통신학회논문지
제12권 제3호 참조

고 시 영(Siyoung Koh)

한국해양정보통신학회논문지
제12권 제3호 참조