논문 2009-46SD-11-5

랜덤함수 매니저를 이용한 버스중재 방식

(Bus Arbitration Policy Using Random Function Manager)

이 국 표*, 윤 영 섭*

(Kook-Pyo Lee and Yung-Sup Yoon)

요 약

본 논문에서는 일반적으로 사용되는 fixed priority 방식과 round-robin 방식의 장점을 살리고 단점을 극복하기 위해 위 두 방식에 랜덤 함수 매니저를 결합하여 두 방식을 모두 사용할 수 있는 랜덤 함수 중재 방식을 제안하였다. Fixed priority 방식에서는 우선순위가 낮은 마스터의 스타베이션 현상이 발생하는 문제점을 가지고 있으며, round-robin 방식은 중요한 마스터에게 우선권을 줄 수 없는 단점이 있다. 그러나 제안한 랜덤 함수 중재 방식은 F-F:R-R의 비율을 사용자가 임의로 조정할 수 있어서 기존 중재방식의 문제점을 해결할 수 있었다. TLM 시뮬레이션 결과, 마스터의 버스점유율을 32%에서 16%로 사용자가 임의 조절이 가능했으며, 마스터의 스타베이션 현상도 방지할 수 있었다.

Abstract

In this study, we have proposed the random function arbitration method using a random function manager in other to use the merits and overcome the demerits of fixed priority and round-robin. Starvation phenomenon can be generated from the master having low priority in fixed priority method. Also, round-robin method can't give a priority to important master. however, as the proposed random function arbitration method can be random set the rate of fixed priority and round-robin, we can solve the problems of conventional arbitration methods. From TLM simulation, we confirmed that the bus utilization of master could be controled from 32% to 16% and the starvation phenomenon of master could be prevented.

Keywords: fixed priority, round-robin, starvation phenomenon, priority

I. 서 론

첨단 과학기술의 발전은 인간의 삶을 편안하고 즐겁게 해주고 있다. 또한 과학기술은 계속적으로 진보하고 있으며, 앞으로 인간이 상상할 수 있는 것보다 더욱더 발전할 것이다[1].

현재 우리가 사용하고 있는 전자제품은 계속적으로 작아지고 기능은 향상되고 있다. 궁극적으로 칩의 SoC(System on a Chip) 화가 되고 있다. SoC는 단일 버스 아키텍처에 여러 개의 마스터, 슬레이브, 아비터, 디코더로 구성되어 있다[2]. 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 선택하는가에 따라 SoC의 성능이 크게 바뀔 수 있다. 일반적으로 사용되고 있는 중재 방식에는 fixed priority 방식, round-robin 방식이 있으며, 최근에는 두 방식의 단점을 보완하기 위해 TDMA[3] 방식, LOTTERY[4] 방식 등 여러 중재 방식들이 제안되고 있다[5].

Fixed priority 방식은 기본적으로 마스터마다 우선순위를 가지고 있다. 결국, 우선순위가 높은 마스터가 버스 점유권의 경쟁에서 이길 수밖에 없다. 따라서 이 방식은 가장 우선순위가 낮은 마스터의 버스 점유권을 보

* 정회원, 인하대학교 전자공학과
(Dept. of Electronics Engineering, Inha University)
※ 이 논문은 인하대학교의 지원에 의하여 발간되었음.
접수일자: 2009년4월3일, 수정완료일: 2009년10월6일

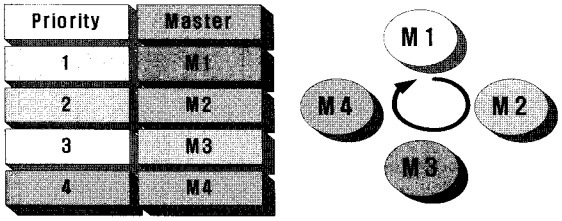
장할 수 없는 문제점을 가지고 있다. Round-robin 방식은 각각의 마스터들이 공평하게 버스 점유권을 가지고 있어 동등한 버스 점유율을 보이지만, 중요한 마스터의 데이터 처리를 빠르게 수행할 수 없는 단점이 있다.

이에 본 논문에서는 각각의 중재 방식이 가지고 있는 문제점을 극복하고자 fixed priority 방식과 round-robin 방식을 혼합하여 사용할 수 있으며, 시스템에 특성에 따라 fixed priority 방식과 round-robin 방식의 적용 선택 확률을 다르게 할 수 있는 랜덤 함수를 이용한 버스 중재 방식을 제안하고, TLM 분석을 이용하여 각각의 중재 방식과의 비교를 통해 성능을 분석 검증하였다[6].

Ⅱ. 본 론

1. 일반적인 중재 방식

그림 1은 일반적인 버스 중재 방식인 fixed priority 방식과 round-robin 방식을 보여주고 있다. 그림 1(a)는 fixed priority 방식이다. Fixed priority 방식은 가장 일반적으로 사용되고 있는 중재 방식이다. 그림 1(a)의 4개의 마스터 중 마스터 M1이 우선순위가 가장 높고 마스터 M4가 우선순위가 가장 낮다. 만약, 마스터 4개가 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 이용 할 수 있다. 그러나 중요하지 않은 마스터의 경우엔 버스 점유권을 얻기가 상당히 힘이 들게 된다. 결국, 우선순위가 낮은 마스터의 경우 스타베이션 현상이 발생하게 된다. 그림 1(b)에 보인 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 1(b)에서 마스터 M1, M2, M3, M4가 각각 순서대로 버스 점유권을 받게 된다고 가정할 때, 마스터 M1이 다른 마스터보다 중요한 데이터 전송이라면 마스터 M1이 데이터 전송을 마치고 또 다른 중요한 데이터 전송을 해야 할 경우, 나머지 마스터 M2, M3, M4의 데이터 전송을 끝낼 동안에 기다려야 한다. 결국 round-robin 방식은 중요한 마스터의 데이터 처리 시간이 늦어질 수 있다는 단점이 있다.



(a) fixed priority 방식 (b) round-robin 방식

그림 1. 일반적인 버스 중재 방식
Fig. 1. General bus arbitration methods.

2. 랜덤함수 매니저를 이용한 버스 중재 방식

그림 2는 본 논문에서 제안한 랜덤 함수를 이용한 버스 중재 방식을 보여준다. fixed priority 방식과 round-robin 방식 중 어느 방식을 결정할 것인가를 랜덤 함수 매니저가 선택하게 된다. 선택의 기준은 랜덤 함수 매니저가 가지고 있는 적용 확률이 있는데, 이는 시스템에 따라 사용자가 임의로 변화시킬 수 있다. 예를 들어, fixed priority 방식의 장점을 부각시키고 싶다면, fixed priority 방식의 선택 확률을 높여주고 round-robin 방식의 선택 확률을 낮추면 된다. 이는 그림 2의 *Ratio_In* 입력신호로 조절가능한데, 본 연구에서는 1~100의 값을 넣을 수 있도록 설계하였다. 예를 들어서 *Ratio_In*의 값을 '10'으로 설정하면 fixed priority 방식이 발생할 확률을 10%이다. 표 1에 자세한 내용이 나타나 있다.

표 2에는 그림 2의 *Mode* 입력신호에 따른 동작을 보여주고 있다. 본 연구에서는 그림 2의 중재방식으로 fixed priority, round-robin, 랜덤 매니저에 의한 중재방식을 *Mode* 입력신호를 이용하여 선택할 수 있도록 구성하였다. 만약 *Mode* 신호가 '00'일 경우는 Random Manager를 사용하지 않고 무조건 Fixed Priority를 사용하도록 *FF/RR Selection Block*의 *Enable* 출력값을 강제적으로 '1'이 나오도록 하였다. 그리고 Mode가 '01'일 경우는 무조건 Round-Robin을 사용하도록 *Enable* 출력값을 강제적으로 '1'이 나오도록 하였다. 마지막으로 *Mode*가 '10' 또는 '11'일 경우는 랜덤 매니저 블록을 적용하도록 하였다.

그림 2의 *Random Number Generation* 블록을 하드웨어적으로 구현하기 위해서, 그림 3과 같은 LFSR 블록도을 적용하였다. LFSR은 입력비트가 이전 상태에 대해 선형적인 시프트 레지스터이다. LFSR의 초기 값

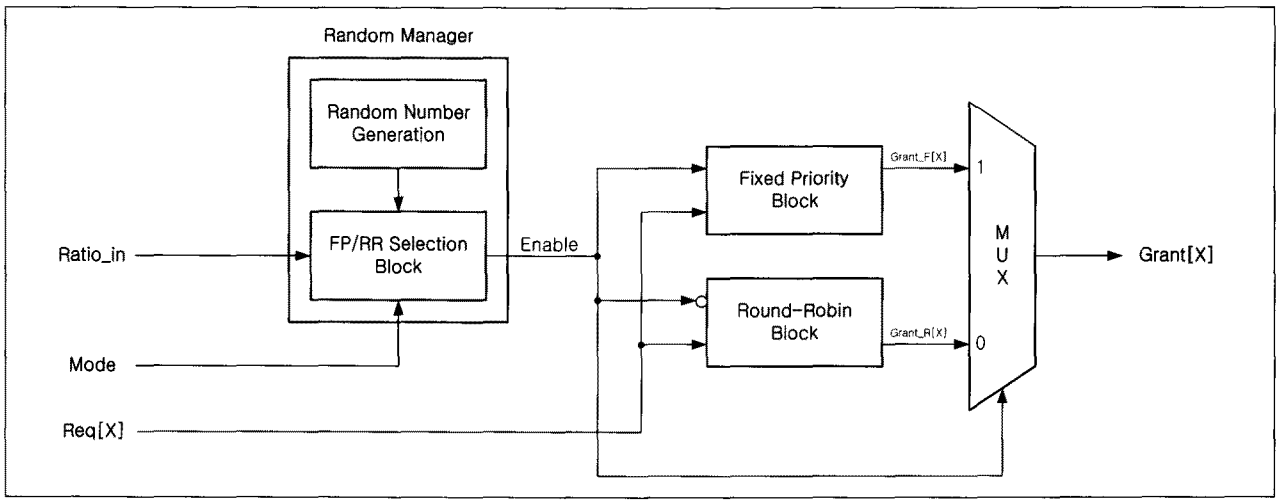


그림 2. 랜덤함수 매니저를 이용한 중재 방식
Fig. 2. Bus arbitration method using random function manager.

표 1. *Ratio_IN*에 의해 정해지는 fixed priority와 round-robin의 비율
Table 1. Ratio between fixed priority and round-robin decided by *Ratio_In*.

Ratio_In	전체 중 Fixed Priority 비율 [%]
1	1
2	2
...	...
100	100

표 2. *Mode* 입력신호 따른 동작
Table 2. Operation due to *Mode* input.

Mode	Enable	설명
00	1	무조건 Fixed Priority
01	0	무조건 Round-Robin
10, 11		Random Manager에 의해 버스중재방식이 결정

은 시드라고 불리고, 레지스터의 동작은 결정론적이기 때문에 레지스터로 발생되는 값의 수열은 현재 상태에 의해 결정되며, 레지스터가 가질 수 있는 상태의 수는 한계가 있어 일정한 주기를 갖고 반복하는데, 되먹임 함수를 잘 선택하면 LFSR은 거의 랜덤으로 긴 주기를 갖는 비트 수열을 생성하게 되어 디지털 논리 회로에서 의사 랜덤 함수(Pseudo Random Function)로 많이 사용된다.[4]

결국, 제안한 중재 방식은 LFSR을 사용한 랜덤 함수 매니저에 의해 fixed priority 방식과 round-robin 방식의 채택이 랜덤으로 결정되기 때문에 fixed priority 방식과 round-robin 방식을 모두 적용할 수 있는 장점을 가지게 된다.

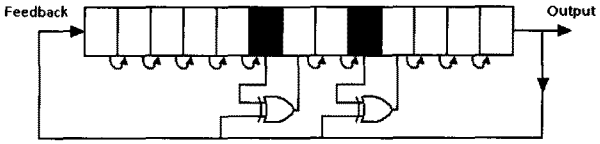


그림 3. LFSR(선형 되먹임 시프트 레지스터) 블록도
Fig. 3. LFSR(Linear Feedback Shift Register) block diagram.

3. 성능 분석

성능분석을 위해 C++로 자체개발한 AMBA TLM (Transaction Level Model)[7]을 사용하였다.

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.

시뮬레이션의 목적은 각 마스터의 버스 사용률과 버스 요청 사이클을 기초로 중재 방식의 효율성을 평가하기 위한 것이다. 시뮬레이션 모델은 6개의 마스터와 4개의 슬레이브, 단일 공용버스로 구성되어 있다. 시뮬레이션 사이클은 10,000,000 사이클로 수행하였다.

그림 4는 fixed priority 방식의 버스 사용률을 보여주고 있다. Fixed priority 방식에서 각 마스터는 마스터 M0, M1, M2, M3, M4, M5 순으로 고유의 우선순위를 가지고 있다. 마스터 M0는 가장 높은 우선순위를 가지고 있기 때문에 처음으로 버스 점유권을 얻을 수 있다.

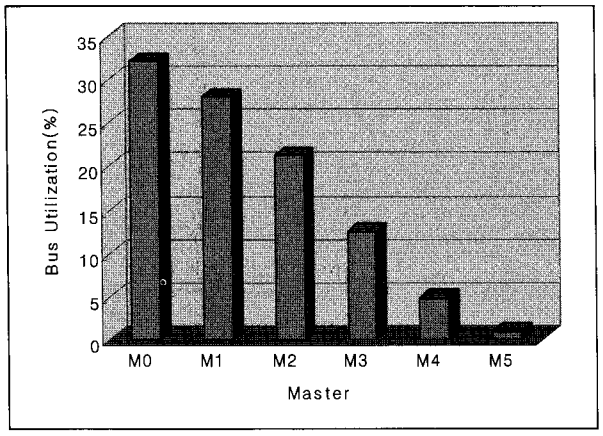


그림 4. Fixed priority 방식의 버스 사용률
Fig. 4. Bus Utilization of fixed priority.

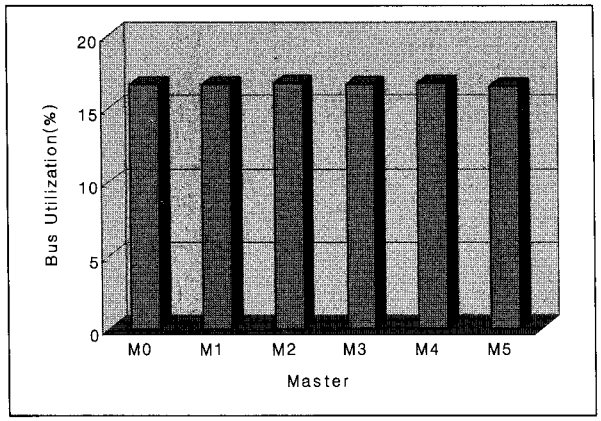


그림 5. Round-robin 방식의 버스 사용률
Fig. 5. Bus Utilization of round-robin.

만약, 마스터 M0이 버스 사용을 요청하지 않는다면, 마스터 M1-M5 순서대로 버스 점유권을 얻는다. 특히, 우선순위가 가장 높은 마스터 M0의 데이터 트랜잭션 사이클이 우선순위가 가장 낮은 마스터 M5보다 30배 이상 크다. 결국, 우선순위가 낮은 마스터는 버스 점유권을 얻기가 매우 어렵다. 가장 낮은 우선순위를 가지고 있는 마스터는 항상 다른 마스터와의 버스 점유권 경쟁에서 이기기 어렵기 때문에, 그 마스터는 스타베이션(starvation) 현상이 발생하게 된다.

그림 5는 round-robin 방식에서 버스 사용률을 보여주고 있다. Round-robin 방식은 각 마스터에게 골고루 분배되어 마스터 M0부터 M5 모두 버스 사용률이 약 16%로 동일함을 보여 fixed priority 방식의 문제점인 스타베이션 현상은 발생하지 않았지만, 마스터 중 중요한 데이터 처리를 해야 하는 마스터가 발생할 경우 해당 마스터에게 우선순위를 줄 수 없는 단점이 있다.

그림 6은 랜덤 함수 중재 방식의 버스 사용률을 보여

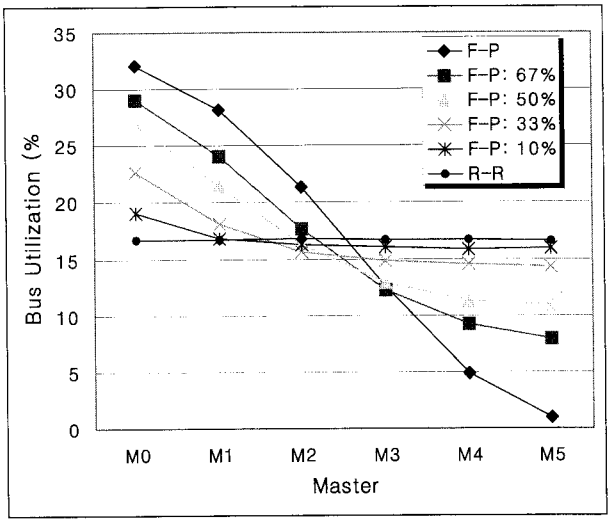


그림 6. 랜덤 함수 중재방식의 버스 사용률
Fig. 6. Bus utilization of random function arbitration method.

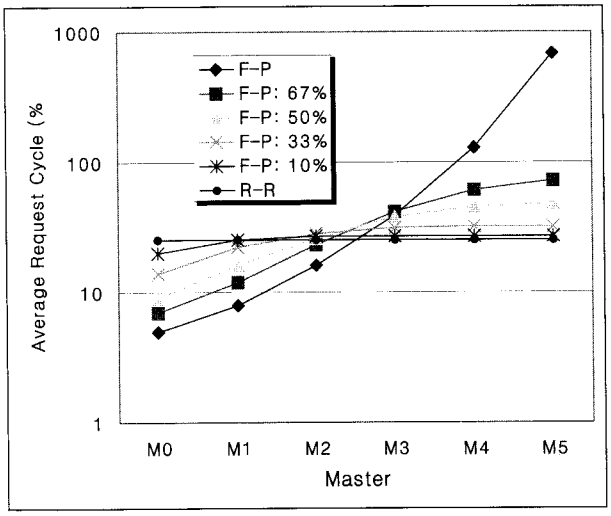


그림 7. 랜덤 함수 중재방식의 평균 요청 사이클
Fig. 7. Average request cycle of random function arbitration method.

주고 있다. F-P(Fixed priority) : R-R(Round-robin) 의 비율을 랜덤 함수를 이용하여 각각 1:0, 2:1, 1:1, 1:2, 1:9, 0:1로 하여 TLM 시뮬레이션을 수행하였다.

그 결과 F-P:R-R 비율이 1:0일 경우는 fixed priority 방식의 경우이며, F-P:R-R 비율이 0:1의 경우는 round-robin 방식을 적용한 경우와 동일해진다. 또한 F-P:R-R 비율을 2:1로 적용하였을 경우에는 우선순위가 높은 마스터 M0의 버스 사용률이 약 30%로 가장 높지만, fixed priority 방식의 문제점이었던 스타베이션 현상이 나타났던 마스터 M5의 버스 사용률이 약 9%로 문제점이 많이 개선되었다. 또한 round-robin 방식의

F-P:R-R 비율을 높게 줄수록, 마스터에 따른 버스사용률이 비슷하게 나타났다.

그림 7은 랜덤 함수 중재 방식의 평균 버스 요청 사이클을 보여주고 있다. 일반적으로 fixed priority 방식의 경우는 우선순위가 높은 마스터 M0의 요청 사이클은 5 사이클로 상당히 적은 데 반해 우선순위가 낮은 마스터 M5의 요청 사이클은 675사이클로 상당히 높음을 알 수 있다.

이는 마스터 M5가 버스점유권 경쟁에서 우선순위가 높은 마스터를 이길 수 없기 발생한 결과이다. 또한, round-robin 방식은 각 마스터들의 우선순위가 똑같기 때문에 평균 요청 사이클 또한 약 25 사이클로 일정함을 볼 수 있다. 이와는 달리 우리가 제안한 랜덤 함수 중재 방식은 F-P:R-R의 비율이 fixed priority 방식이 높을 경우엔 fixed priority 방식과 근접하지만, fixed priroity 방식과의 다른 점은 우선순위가 낮은 마스터 M5의 평균 요청 사이클이 100사이클을 넘지 않았다는 부분이다. 이는 fixed priority 방식의 문제점인 스타베이션 현상을 방지한 것으로 여겨진다. 또한 F-P:R-R 비율이 round-robin 방식으로 확률이 높아질수록 점점 round-robin 방식의 기울기와 유사해짐을 볼 수 있다.

그림 8은 랜덤 함수 중재 방식의 최대 요청 사이클을 보여준다. 그림 7과 마찬가지로 fixed priority 방식은 마스터 M0의 최대 요청 사이클이 상당히 낮음에 비해 우선순위가 낮은 마스터 M5의 최대 요청 사이클이 8,428사이클로 거의 10,000사이클에 가까움을 알 수 있다. 또한 round-robin 방식의 최대 요청 사이클은 각 마스터들이 약 100사이클 정도로 비슷하였다. 이에 비해 본 논문에서 제안한 랜덤 함수 중재 방식의 경우는 F-P:R-R 비율이 fixed priority 방식이 높은 경우에서 round-robin 방식으로 확률이 높아질수록 점점 round-robin 방식과 비슷해짐을 볼 수 있다. 이상의 시뮬레이션 결과로부터 랜덤 함수 중재 방식이 fixed priority 방식과 round-robin 방식보다 더 유연하게 버스사용율과 버스 요청사이클을 조절할 수 있음을 알 수 있다.

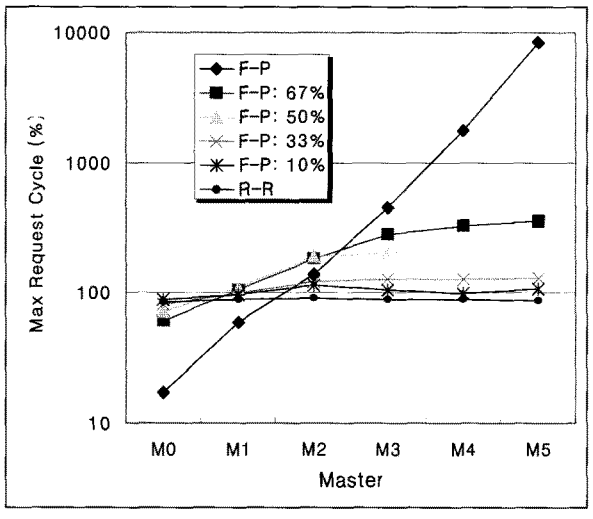


그림 8. 랜덤 함수 중재방식의 최대 요청 사이클
Fig. 8. Max request cycle of random function arbitration method.

Ⅲ. 결 론

버스 사용률을 효율적으로 제어하기 위해 우리는 fixed priority 방식과 round-robin 방식의 장점을 모두 얻을 수 있는 랜덤 함수 중재 방식을 제안하고 TLM 방법을 이용하여 성능을 분석하였다.

시뮬레이션 결과, 제안한 중재 방식은 fixed priority 방식의 문제점인 스타베이션 현상을 방지하였으며, round-robin 방식의 문제점인 중요한 마스터의 데이터 처리를 우선적으로 처리 할 수 있었다. 또한 F-F:R-R 비율을 사용자가 임의로 조정할 수 있어 기존의 두 방식보다 더 유연하게 중재할 수 있는 방식임을 알 수 있었다.

참 고 문 헌

[1] Gunar Schirner, Rainer D"omer, "System Level Modeling of an AMBA Bus", technical Report, 2005.
[2] K. Lee and Y. Yoon, "Architecture Exploration for Performance Improvement of SoC Chip Based on AMBA System", ICCIT, pp.739-744, 2007.
[3] Y. Xu, L. Li, Ming-lun Gao, B.Zhand, Zhao-yu Jiand, Gao-ming Du, W. Zhang, "An Adaptive Dynamic Arbiter for Multi-Processor SoC", Solid-State and Integrated Circuit Technology International Conf., pp.1993-1996, 2006.
[4] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "The LOTTERYBUS On-Chip Communication Architecture", IEEE Trans. VLSI Systems, vol.14, no.6, 2006.
[5] M. Jun, K. Bang, H. Lee and E. Chung, "Latency-aware bus arbitration for real-time embedded systems," IEICE Trans. Inf.& Syst.,vol .E90-D, no.3, 2007.
[6] 이국표, 윤영섭, "다양한 버스 중재방식에 따른 플라잉 마스터 버스아키텍처의 TLM 성능분석", 전

자공학회 논문지, vol. 45, CI, no.5, pp.1-7,2008.
[7] AMBA TM Specification(AHB) (Rev 2.0), ARM Ltd, May 1999.

저 자 소 개

이 국 표(정회원)
대한전자공학회 논문지
제45권 SD편 제4호 참조

윤 영 섭(정회원)
대한전자공학회 논문지
제45권 SD편 제4호 참조