논문 2009-46SD-2-8

하이브리드 버스 중재 방식

(The Hybrid Bus arbitration policy)

이 국 표*, 윤 영 섭*

(Kook-Pyo Lee and Yung-Sup Yoon)

요 약

SoC(System on Chip)는 버스 아키텍처 안에 여러 개의 마스터, 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 선택하는가에 따라 SoC의 성능이 크게 바뀔 수 있다. 일반적인 중재 방식에는 fixed priority 방식, round-robin 방식, TDM 중재 방식 등이 있다. 본 논문에서는 TLM 알고리즘을 구성하여 일반적인 중재방식을 TLM 시뮬레이션을 통해 비교 분석하였다. 또한 새로운 중재 방식인 하이브리드 버스 중재 방식을 제안하고 다른 중재 방식과 비교하여 성능을 검증하였다.

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. Fixed priority, round-robin, TDM arbitration are used in general arbitration method, In this study, we compose TLM algorithm and analyze general arbitration methods through TLM simulation. Consequently, we propose the hybrid bus arbitration policy and verify the performance, compared with the other arbitration methods.

Keywords: SoC, bus architecture, transaction, arbitration, fixed priority, round-robin, performance

Ⅰ. 서 론

현대 사회는 휴대용 전자제품의 홍수 속에 살고 있다. 휴대폰, PDA, MP3, 게임기 등 각종 전자제품이 일상생활에 깊이 침투해있다. 소비자들은 가격이 더 싸고 성능이 더 좋은 제품을 원하고 있다. 소비자들의 요구를 해결하고 수익을 얻기 위해 칩 제조 회사들은 경쟁사들보다 더 빨리 제품을 시장에 내놓기를 희망한다. 이를 해결하고자 대다수 회사들은 칩을 제작하기 전, 성능을 빠르고 정확하게 분석할 수 있는 High-Level 설계 방법을 찾기 위해 노력을 아끼지 않고 있으며, Transaction Level Modeling(TLM) 방법이 최근에 많이 연구되고 있다.[1]

또한 IP (Intellectual Property) 블록을 사용한 설계의 재사용 기법이 연구 개발되어, 이미 검증되고 설계에 사용된 제품이나 지적 재산권, 특허로 보호 받는 IP 블록의 사용은 계속 증가하고 있다. 그러나 IP 자체에 대한 설계와 더불어 중요한 사항으로서 이 IP들을 연결하는 버스 시스템의 구조를 들 수 있다. 버스 시스템은 IP들 간의 통신 순서와 방법을 정의하고 제어한다. 그러므로 버스 시스템과 IP 간의 호환성을 고려할 때, 버스 시스템 자체의 성능이 설계하는 SoC의 성능을 좌우하는 요소로 부각되고 있다[2]. 이 중 ARM 프로세서는 전 세계

* 정회원, 인하대학교 전자공학과
(Dept. of Electronics Engineering, Inha University)
접수일자: 2008년6월2일, 수정완료일: 2009년2월4일

의 SoC 시장의 70% 이상을 차지하고 있다. 결국 AMBA(Advanced Microcontroller Bus Architecture)[3] 시스템이 온 칩 통신의 표준이 되어 가고 있다. AMBA 시스템에는 AHB(Advanced High performance Bus)와 APB(Advanced Peripheral Bus)가 있는데, AHB는 고성능 버스를 의미하며 성능 향상을 위해 현재 많이 연구되고 있다. 일반적인 AHB는 단일 버스에 여러 개의 마스터와 슬레이브, 아비터, 디코더로 구성되어 있다.

마스터는 CPU, DMA, DSP 등과 같이 어드레스나 제어 신호를 내보냄으로 "read"나 "write"의 동작을 할 수 있도록 하는 주체를 말한다. 또한 슬레이브는 DRAM, SDRAM 등과 같이 주어진 어드레스 공간 내에서 "read"나 "write"를 가능하게 해주는 장치를 말한다. 그리고 디코더는 마스터로부터 나오는 어드레스의 상위 비트를 가지고 적절한 슬레이브를 선택해주는 역할을 수행한다. 마지막으로 아비터는 여러 개의 마스터가 동시간대에 버스를 사용할 수 없기 때문에 이를 중재하는 역할을 수행하고 중재하는 방식에 따라 버스의 효율적인 중재가 가능하기 때문에 성능 향상을 위해 현재 많이 연구되고 있다. 아비터의 일반적인 중재 방식에는 fixed priority 방식, round-robin 방식, TDM 중재 방식, Lottery bus 방식 등이 있다[4]. 이 방식을 일부 수정하여 priority with break 방식, priority with waiting time control 방식, short job first 방식, short job first with waiting time control 방식 등이 새롭게 개발되고 있다[5~6].

본 논문에서는 TLM 알고리즘을 구성하고 일반적인 중재 방식인 fixed priority 방식, round-robin 방식, TDM 중재 방식, Lottery bus 방식의 버스 점유율과 데이터 전송량, SDRAM 레이턴시, 버스 요청 사이클을 TLM 시뮬레이션 결과 값을 통해 비교해 보았다. 그리고 IEEE 802.11 네트워크 SoC[7]에 적합한 중재방식으로 하이브리드 버스 중재 방식을 제안하고, 시뮬레이션을 통해 일반적인 중재 방식과 비교하여 성능을 검증하였다.

Ⅱ. 모 델

1. 버스 아키텍처 모델구성

버스 아키텍처를 모델링하기 위해서 그림1과 같이 버스 내의 데이터전송 흐름도를 순서대로 나타내었다. 각각의 마스터(M1, M2 등)들은 동시에 버스 사용 요청

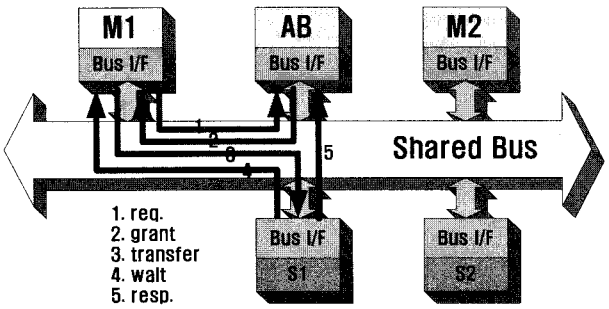


그림 1. 공용버스 내의 데이터전송 흐름도
Fig. 1. Data transaction flow in the internal shared bus.

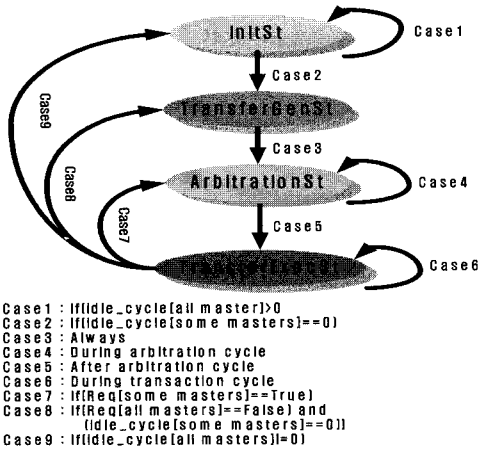


그림 2. 버스 아키텍처 모델의 상태도
Fig. 2. State machine of bus architecture model.

(req.)을 할 수 있으며 아비터 AB에 의해서 선택된 마스터에게 버스사용을 허가(grant)한다.

사용허가를 받은 마스터는 해당 슬레이브에 데이터 전송을 하고, 슬레이브는 "wait" 신호와 "resp" 신호를 통해 마스터와 아비터에게 데이터 처리에 관한 응답을 한다. 이와 같은 싱글 버스 아키텍처를 모델링하기 위해 그림 2와 같은 상태도를 구성하였다.

버스 아키텍처의 상태도는 InitSt, TransferGenSt, ArbitrationSt, TransferExecSt의 4가지 상태를 거치게 되는데, InitSt는 마스터를 통한 버스사용 요청이 발생하지 않아 버스의 동작이 없는 상태를 의미한다. 1개 이상의 마스터에서 버스를 사용하려고 할 경우, TransferGenSt 상태에서 버스요청 신호와 데이터 전송 관련 신호를 발생시키게 된다. 이후 ArbitrationSt 상태로 진입하게 되는데, 여러 가지 버스중재 알고리즘을 통해 하나의 마스터에게 버스 사용권(grant 신호)을 할당하게 되며 TransferExecSt 상태로 진입하게 된다. TransferExecSt에서 실제 데이터 트랜잭션을 시작하고

표 1. 버스 아키텍처 알고리즘
Table 1. Algorithm of bus architecture.

```
1:  single_bus_model(Arbitration_Type)
2:  begin
3:    Cur_Cycle=0
4:    State=InitSt
5:    Master_Signal[all masters]=Idle
6:    Slave_Signal[all slaves]=Idle
7:    while (Cur_Cycle<Final_Cycle) do
8:      if (State=InitSt) then
9:        Idle_Gen(Master_Signal[selected masters])
10:       while(Master_Signal.Idle_Cycle[all masters]--) do
11:         Cur_Cycle++
12:         Detail_Cycles_Cal(NULL,NULL,InitSt)
13:       end while
14:       Execute_Bus_Model(Master_Signal[all masters],NULL,InitSt)
15:       State=TransferGenSt
16:     end if
17:     else if (State=TransferGenSt) then
18:       Req_Gen(Master_Signal[selected masters])
19:       Addr_Gen(Master_Signal[selected masters])
20:       Data_Gen(Master_Signal[selected masters])
21:       Transfer_Signal_Gen(Master_Signal[selected masters])
22:       State=ArbitrationSt
23:     end else if
24:     else if (State=ArbitrationSt) then
25:       Arbitration(Req[selected masters],Arbitration_Type)
26:      while (ARBITRATION_CYCLE--) do
27:         Cur_Cycle++
28:         Detail_Cycles_Cal(NULL,NULL,ArbitrationSt)
29:         Execute_Bus_Model(Master_Signal[all masters],NULL,ArbitrationSt)
30:       end while
31:       State=TransferExecSt
32:     end else if
33:     else then  //State is TransferExecSt.
34:      while ((Master_Signal.Data_Size[selected master]+Slave_Signal.
         Slave_Latency[selected slave])--) do
35:         Cur_Cycle++
36:         Detail_Cycles_Cal(Master_Signal[all masters],Slave_Signal[all slaves],
            TransferExecSt)
37:         Execute_Bus_Model(Master_Signal[all masters],Slave_Signal[all slaves],
            TransferExecStSt)
38:       end while
39:       Master_Signal.Req[selected master]=False
40:       if     (Master_Signal.Req[some masters]==True)     State=ArbitrationSt
41:       else if(Master_Signal.Idle_Cycle[some masters]==0) State=TransferGenSt
42:       else                                              State=InitSt
43:     end else
44:   end while
45: end single_bus_model(Arbitration_Type)
```

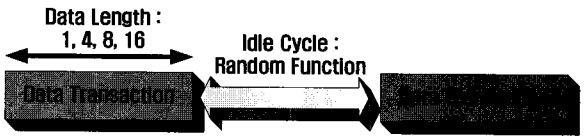
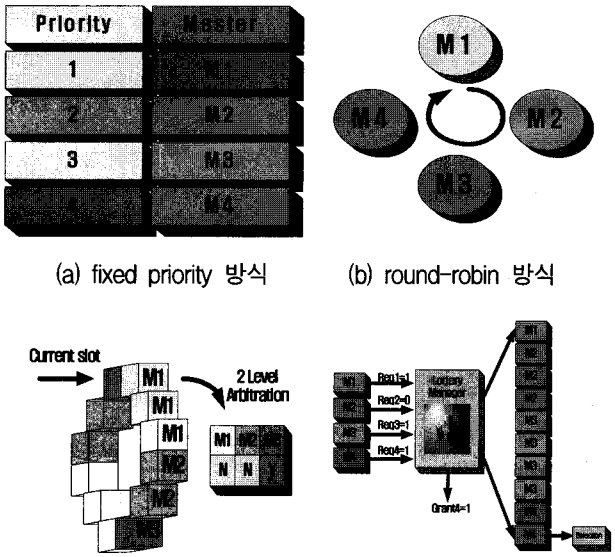


그림 3. 데이터 길이와 idle 사이클
Fig. 3. Data length and idle cycle.

해당 슬레이브에 데이터를 전송하게 된다. 데이터 전송을 마친 후, 다른 마스터에서 버스요청이 있으면 ArbitrationSt 상태로 진입하고, 마스터가 버스를 사용하려고 하면 TransferGenSt 상태로 진입한다. 만약 모든 마스터가 버스를 사용하지 않을 경우에는 InitSt 상태로 진입하게 된다. 이 상태도를 근거로 버스 아키텍처 알고리즘을 구성하였는데, 표 1에 자세히 나타나 있다.

전 세계의 70% 이상을 차지하고 있는 AMBA 시스템의 경우, 그림 3과 같이 마스터에서 발생하는 데이터는 싱글 데이터와 버스트 데이터가 있으며, 버스트 데이터 길이는 4, 8, 16까지 지원한다. 그리고 idle 사이클 지연 후 새로운 데이터를 발생시키는데, 본 모델에서는 idle 사이클에 대해서 랜덤 함수를 이용하였다.



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

(c) TDM 중재 방식 (d) Lottery bus 방식

그림 4. 다양한 버스 중재 방식
Fig. 4. Various bus arbitration policies.

그림 4에 보듯이 대표적인 버스 중재 방식은 fixed priority 방식과 round-robin 방식, TDM 중재방식, Lottery bus 방식이 있다. 그림 4(a)는 fixed priority 방식을 보여준다. Fixed priority 방식의 경우 각 마스터들은 고정된 우선순위를 가지고 있다. 예를 들어, 마스터 M1의 우선순위가 마스터 M2보다 높으면, 아비터는 버스 점유권을 마스터 M1에게 준다. Fixed priority 방식의 경우 중요한 마스터는 우선순위를 높일 수 있는 장점을 가지지만, 우선순위가 가장 낮은 마스터의 스타베이션을 일으킬 수 있는 단점을 가지고 있다. 그림 4(b)는 round-robin 방식을 보여준다. 이 방식은 마스터의 우선순위가 정해지지 않고 골고루 버스 점유권을 주기 때문에 마스터들이 공평하게 버스를 이용할 수 있어 스타베이션을 방지할 수 있다. 그러나 중요한 데이터 처리를 할 경우 이 방식은 한계를 가진다. 그림 4(c)는 TDM 중재 방식을 보여준다. TDM 중재 방식도 골고루 마스터의 버스 점유권이 돌아가기 때문에 스타베이션을 방지할 수 있는 장점을 가지고 있지만, 중요한 데이터 처리를 가져야 하는 마스터의 대기시간이 길어 질 수 있다는 단점을 가지고 있다[4]. 그림 4(d)는 Lottery bus 방식을 보여준다. Lottery bus 방식의 경우는 마스터들에게 버스 점유권을 확률적으로 주는 방식이다[4]. 중요한 마스터의 경우는 좀 더 많

은 버스 점유권을 주고 그렇지 않은 경우는 좀 더 적게 주는 방식으로, 최근에 TDM 버스중재를 개선하는 방식으로 제안되고 있다.

본 연구에서는 표 1의 알고리즘을 이용하여 다양한 버스중재 방식에 대한 성능을 비교분석하고, 새로운 버스중재 방식에 대해 제안하려고 한다.

Ⅲ. 결과 및 토의

TLM 방법으로 여러 버스 중재방식에 따른 싱글 버스 아키텍처의 성능을 분석해 보았다. 슬레이브는 데이터 레이턴시가 길고, 가변적인 SDRAM 컨트롤러로 하였으며, 총 마스터의 개수는 4개로 정하여 1,000,000 cycle까지 시뮬레이션을 수행하였다.

그림 5(a)와 (b)는 각각 fixed priority 방식과 round-robin 방식의 데이터 전송량을 보여주고 있다. Case1, Case2, Case3, Case4는 데이터 통신의 빈번함에 차이가 있으며, 표 2의 조건과 같이 Case1에서 Case4로 갈수록 idle cycle의 길이가 길어진다. 그림 5(a)의 fixed priority 방식을 보면, 평균 idle cycle이 짧은 Case1에서는 우선순위에 의한 마스터별 버스 점유율이 큰 차이를 보이고 있으나 평균 idle cycle이 긴 Case4에서는 우선순위에 의한 마스터별 버스 점유율에 큰 차이가 거의 없음을 알 수 있다. 그림 5(a)의 fixed priority 방식에서

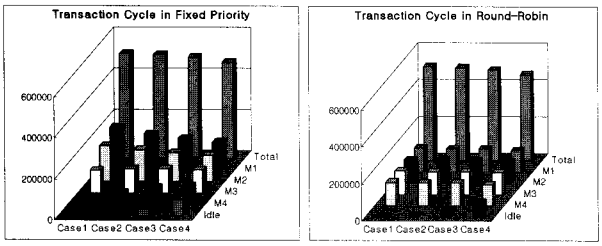


그림 5. (a) Fixed priority 방식의 데이터 트랜잭션,
(b) Round-robin 방식의 데이터 트랜잭션
Fig. 5. (a) Data transaction cycle in round-robin,
(b) Data transaction cycle in fixed priority.

표 2. TLM 시뮬레이션 조건
Table 2. TLM simulation condition.

Case	Data Transaction (Random Function)	Idle Cycle (Random Function)	
		Range	Mean
Case1	Single, Burst(4,8,16)	0 ~ 20	10
Case2	Single, Burst(4,8,16)	0 ~ 30	15
Case3	Single, Burst(4,8,16)	0 ~ 40	20
Case4	Single, Burst(4,8,16)	0 ~ 50	25

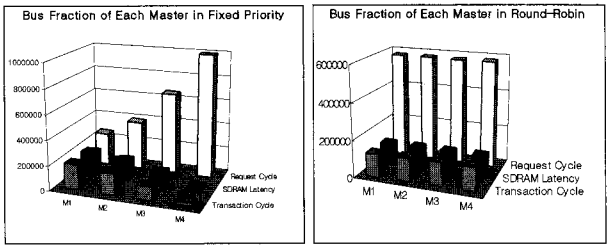


그림 6. (a) Fixed priority 방식에서 버스 사용률,
(b) Round-robin 방식에서 버스 사용률
Fig. 6. (a) Bus usage proportion in fixed priority,
(b) Bus usage proportion in round-robin.

Case1을 보면, 우선순위가 높은 마스터 M1의 경우 200,000 사이클의 데이터 전송이 발생한데 비해 우선순위가 낮은 마스터 M4는 390 사이클로 우선순위에 따른 데이터 처리의 차이가 크다. 그림 5(b)의 round-robin 방식의 경우는 Case1, Case2, Case3, Case4에 상관없이 마스터에 따른 버스 점유율이 균등하게 확보되고 있음을 알 수 있다.

그림 6은 fixed priority 방식과 round-robin 방식의 각 마스터 당 데이터 처리량, SDRAM 레이턴시, 버스 요청 사이클을 보여 주고 있다. 슬레이브는 SDRAM 컨트롤러이며, precharge 사이클, refresh 사이클, 로우 컬럼 엑세스 사이클 등이 고려되어 있다. 그림 6에서 보듯이 SDRAM 레이턴시가 데이터 처리 사이클(transaction cycle)과 유사할 정도로 상대적으로 큰 값을 나타내며, 그림 6(a)의 마스터 M4는 버스요청 사이클(request cycle)이 1,000,000 사이클에 근접할 정도로 상당히 큰 값을 갖아서 사실상 스타베이션 (starvation) 현상이 발생했음을 알 수 있다. 반면에 그림 6(b)의 round-robin 방식은 마스터에 따른 데이터 처리량이 균등하며, 각 마스터의 버스요청 사이클이 약 600,000 사

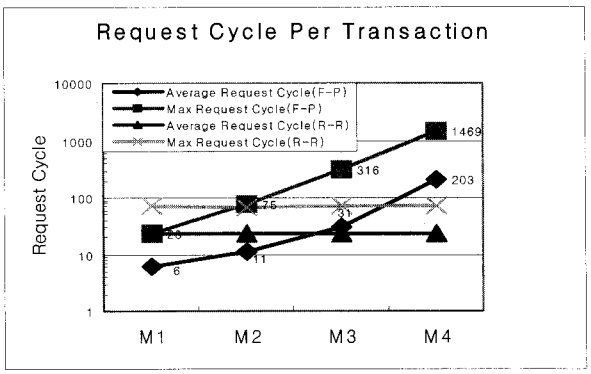


그림 7. 각 중재방식 전송량 당 요청 사이클
Fig. 7. Request cycle per transaction.

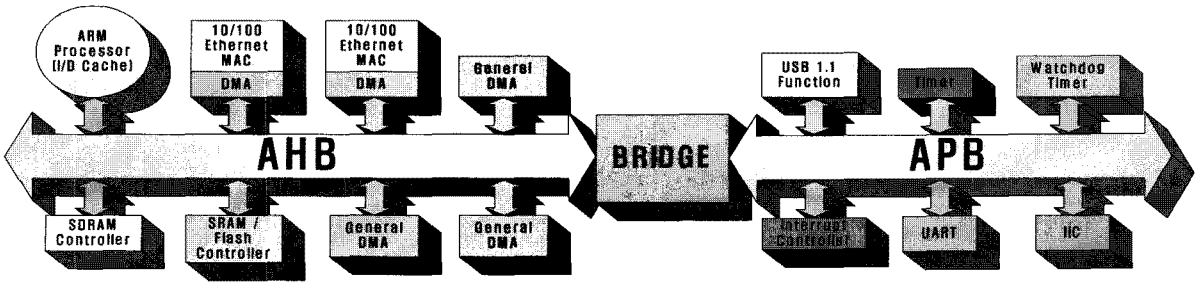


그림 8. 성능분석을 위한 SoC의 예
Fig. 8. SoC example for performance analysis.

이클을 보여 주고 있다.

그림 7은 fixed priority 방식과 round-robin 방식에서 하나의 데이터 트랜잭션 당, 사용되는 평균, 최대 버스요청 사이클을 보여준다. Fixed priority 방식의 경우는 마스터의 우선순위가 정해져 있기 때문에 버스 점유권을 많이 얻을 수 있는 마스터 M1의 경우의 평균, 최대 버스요청 사이클은 6, 23으로 상대적으로 작지만, 우선순위가 낮은 마스터 M4의 평균, 최대 버스요청 사이클은 1469, 203 사이클로 무척 크며, 사실상 스타베이션 현상이 발생하였음을 알 수 있다. Round-robin 방식의 경우는 각 마스터에게 균등하게 버스 점유권이 돌아가기 때문에 각 마스터의 버스요청 사이클이 수십 사이클로 안정적이지만, 우선순위가 고려되지 않음을 알 수 있다.

그림 8은 실제로 성능분석에 사용할 IEEE 802.11 네트워크 SoC[8] 예이다. 고성능 버스인 AHB와 주변 장치에 주로 사용되는 APB 사이에는 브리지가 결합되어 있다. 마스터는 ARM 프로세서와 2개의 이더넷과 3개의 DMA로 구성되어 있다. 본 연구에서는 그림 8의 예에 적합한 버스 중재방식에 대해 파악해 보려고 한다.

그림 8과 같은 일반적인 SoC에서 프로세서는 실제 칩을 운영하므로 우선순위가 가장 높아야 한다. 그 외의 마스터는 사용자의 목적에 따라 중요도가 달라지는데, 그림 8의 SoC는 실제 데이터 패킷을 전송하는 2개의 이더넷이 우선순위가 높고, 일반 DMA 3개는 우선순위가 상대적으로 낮다.

그림 9는 그림 8의 마스터 우선순위를 고려한 하이브리드 버스 중재 방식(Hybrid bus arbitration policy)을 보여주고 있다. 하이브리드 버스 중재 방식은 fixed priority 방식과 round-robin 방식을 사용자의 필요성에 따라서 그룹별로 혼재시킨 방식으로 사용자의 요구에 정확히 부응하는 방식이다.

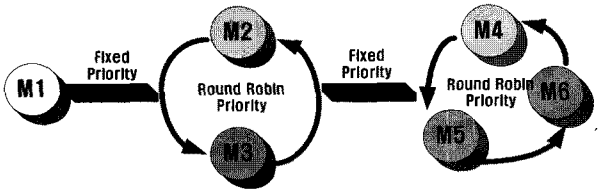


그림 9. 하이브리드 버스 중재 방식
Fig. 9. Hybrid bus arbitration policy.

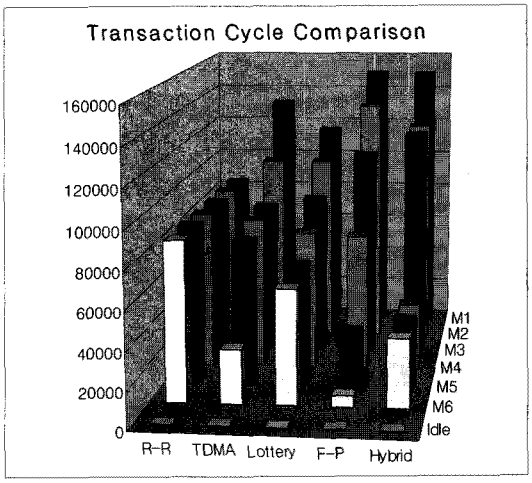


그림 10. 데이터 전송량 비교
Fig. 10. Transaction cycle comparison.
(R-R: round-robin, F-P: fixed priority)

그림 10은 각 중재 방식과 우리가 제안한 하이브리드 중재 방식의 데이터 전송량을 비교하고 있다.

Round-robin 방식의 경우는 각 마스터 당 약 84,000 사이클로 비슷함을 보이고 있다. Fixed priority 방식의 경우는 우선순위가 가장 높은 마스터 M1이 150,438 사이클로 가장 많은 데이터 전송량을 보였지만, 마스터 M6의 경우는 데이터 전송이 거의 발생하지 못했다.

그림 10에서 TDM 중재 방식은 마스터 M1에서 마스터 M6의 슬롯(slot) 수를 각각 4, 1, 1, 1, 1, 1로 정하였으나, 데이터 트랜잭션 사이클은 이와 정확하게 일치하

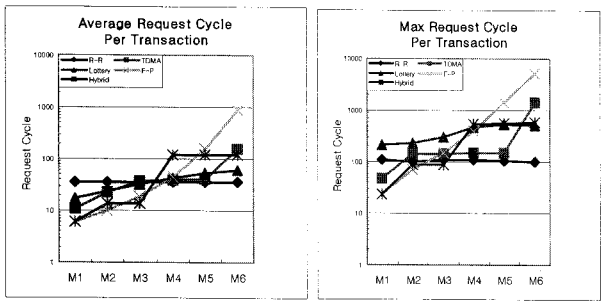


그림 11. (a) 데이터 전송량 당 평균 요청 사이클,
(b) 데이터 전송량 당 최대 요청 사이클
Fig. 11. (a) Average Request Cycle Per Transaction,
(b) Max Request Cycle Per Transaction.

지 않았다. 특히 마스터 M2에서 마스터 M6의 슬롯 수가 같았지만 데이터 트랜잭션 사이클이 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 10에서 Lottery 중재 방식은 마스터 M1에서 마스터 M6의 버스점유 확률을 4/9, 1/9, 1/9, 1/9, 1/9, 1/9로 정하였으나, TDM 버스중재와 마찬가지로 2순위 중재 때문에 버스 점유 확률과 데이터 트랜잭션 사이클이 정확하게 일치하지 않았다.

그림 9와 같이 fixed priority와 round-robin 중재를 혼재한 하이브리드 버스 중재 방식의 결과도 그림 10에 보여주고 있다. 가장 중요한 마스터 M1의 경우 약 150,000 사이클의 데이터 전송이 이루어졌고 마스터 M2, M3의 경우 또한 두 번째로 높은 데이터 전송이 이루어졌다. 나머지 마스터들 또한 균등한 데이터 전송이 이루어졌다.

그림 11(a)과 (b)는 데이터 전송량 당 평균 요청 사이클과 최대 요청 사이클을 보여준다. 우리가 제안한 하이브리드 중재 방식은 계단 모양을 보이고 있다. 가장 많은 데이터 전송량을 보인 마스터 M1의 요청 사이클이 가장 적었으며, 가장 적은 데이터 전송량을 보인 마스터 M4, M5, M6의 요청 사이클이 가장 많음을 알 수 있다.

VI. 결 론

본 논문에서 우리는 TLM 알고리즘을 구성하고 새로운 중재방식인 하이브리드 버스 중재 방식을 제안하고 성능을 분석하였다. 하이브리드 버스 중재는 fixed priority의 장점인 중요한 마스터의 데이터 전송을 할 수 있는 특징과 round-robin 중재방식의 장점인 스타베이션을 방지할 수 있는 특징을 혼재한 방식이다. TDM 중재 방식이나 Lottery bus 방식은 마스터의 사용비율을 임의적으로 줄 수 있는 특징이 있는데, 2순위 중재에 의해 실제 마스터의 사용비율과 정확하게 일치하지 않았다. 그러나 본 하이브리드 버스 중재방식은 버스사용 우선순위에 따라 정확하게 동작하였으며, 우선순위가 같은 그룹 내의 버스사용비율이 거의 비슷함을 알 수 있었다.

앞으로 버스 중재방식을 다양한 종류의 하이브리드 방식으로 계속 발전할 것으로 파악되며, 본 논문이 중요한 자료로 역할을 할 것으로 여겨진다.

참 고 문 헌

[1] K. Lee and Y. Yoon, "Architecture Exploration for Performance Improvement of SoC Chip Based on AMBA System", ICCIT, pp.739-744, 2007.
[2] Gunar Schirner, Rainer D"omer, "System Level Modeling of an AMBA Bus", technical Report, 2005.
[3] AMBA TM Specification(AHB) (Rev 2.0), ARM Ltd, May 1999.
[4] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "The LOTTERYBUS On-Chip Communication Architecture", IEEE Trans. VLSI Systems, vol.14, no.6, 2006.
[5] 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.
[6] K.Sekar, K.Lahiri, A.Raghunathan, and S.Dey, "FLEXBUS:Ahigh-performance system-on-chip communication architecture with a dynamically configurable topology," Proc. Des. Autom. Conf., pp.571-574, 2005.
[7] http://www.samsung.com/global/business /semiconductor/productInfo.do?fmly_id=234&partnum=S3C2510A

저 자 소 개

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

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