스코어 버스 중재방식의 설계 및 성능 분석

이국표* · 고시영**

Design and Performance Analysis of Score Bus Arbitration Method

Kook-pyo Lee* · Si-Young Koh**

요 약

버스 시스템은 하나의 버스 내에 여러 개의 마스터와 슬레이브, 아비터 그리고 디코더로 구성되어 있다. 마스터는 CPU, DMA, DSP 등과 같은 데이터의 명령을 수행하는 프로세서를 말하며, 슬레이브는 SRAM, SDRAM, 레지스터 등과 같이 명령에 응답하는 메모리를 말한다. 또한 아비터는 마스터가 동시간대에 버스를 이용할 수 없기 때문에 이를 중재하는 역할을 수행하는데, 어떠한 중재 방식을 선택하는가에 따라 버스 시스템의 성능이 크게 바뀔 수 있다. 일반적인 중재 방식에는 fixed priority 방식, round-robin 방식이 있으며, 이를 개선한 TDMA 방식과 Lottery bus 방식 등이 현재까지 제안되었다. 본 논문에서는 새로운 중재 방식인 스코어 중재 방식을 제안하고 RTL 디자인후 하이닉스 0.18um 공정 라이브러리를 이용하여 설계 합성하였으며, 일반적인 중재방식과 시뮬레이션을 통해 성능을 비교 분석하였다.

ABSTRACT

Bus system consists of several masters, slaves, arbiter and decoder in a bus. Master means the processor that performs data command like CPU, DMA, DSP and slave means the memory that responds the data command 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, bus system performance can be changed definitely. Fixed priority and round-robin are used in general arbitration method and TDMA and Lottery bus methods are proposed currently as the improved arbitration schemes. In this study, we proposed the score arbitration method and synthesized it using Hynix 0.18um technology, after design of RTL. Also we analyze the performance compared with general arbitration methods through simulation.

키워드

SoC, 버스 구성, 중재 방식, 처리량

Key word

SoC, bus architecture, arbitration policy, Throughput

* 정회원 : 영진전문대학 전자정보통신계열 (kplee@yjc.ac.kr)

** 종신회원 : 경일대학교 전자정보통신공학부 (교신저자)

접수일자 : 2011. 09. 29

심사완료일자 : 2011. 10. 12

Ⅰ. 서 론

현대 사회가 점점 정보화 사회로 진보하는 데에는 반도체 산업이 매우 중요한 역할을 하고 있다. 인터넷 시대의 등장과 함께 텔레비전, 냉장고와 같은 가전에서부터 노트북, 휴대용 정보기기 및 스마트 전자기기에 대한 수요가 증가하면서 전자회로의 임베디드(Embedded) 시스템화, 소형화, 저전력화되어 가고 있다. 또한 반도체 공정 기술의 발달로 기존의 여러 다른 기능을 하는 칩들을 하나에 집적시키는 SoC(System on a Chip) 기술이 각광받고 있다.[1]

SoC 버스 시스템은 IP들 간의 통신 순서와 방법을 정의하고 제어한다. 그러므로 버스 시스템의 성능이 SoC의 성능을 좌우하는 중요한 요소로 부각되고 있다. 버스 시스템에는 ISA, PCI, MCA 등 여러 종류가 있지만, 이 중에서 ARM 프로세서의 AMBA(Advanced Microcontroller Bus Architecture) 가 온 칩 통신의 표준이 되고 있다.[2]

AMBA는 AHB(Advanced High-performance Bus), ASB(Advanced System Bus) 그리고 APB(Advanced Peripheral Bus)가 있으며, AXI(Advance eXtensible Interface)가 현재 새롭게 개발되었지만, 아직까지 고성능 버스인 AHB가 성능 향상을 위해 많이 연구되고 있다. 전형적인 AHB는 하나의 단일 버스 내에 여러 개의 마스터와 슬레이브, 아비터, 디코더로 구성되어 있다. 마스터는 CPU, DMA, DSP 등과 같은 프로세서들을 말하며, 슬레이브는 마스터와는 다르게 DRAM, SRAM과 같은 메모리를 의미한다. 또한, 아비터는 여러 개의 마스터가 동시간에 버스를 이용할 수 없기 때문에 이를 중재하는 역할을 수행하고 중재하는 방식에 따라 버스의 효율적인 중재가 가능하기 때문에 전체 시스템의 성능 향상을 위해 많이 연구되고 있는 분야이다. 마지막으로 디코더는 마스터로부터 나오는 어드레스의 상위 비트를 가지고 적절한 슬레이브를 선택해주는 역할을 한다.

기존의 아비터 중재 방식에는 fixed priority 방식, round-robin 방식, TDMA 방식, Lottery bus 방식 등 여러 가지가 있다.[3～8] 본 논문에서는 새로운 방식인 스코어 중재 방식을 제안하고 Verilog로 설계 하고, 하이닉스 공정 라이브러리를 이용하여 합성 및 최대주파수를 측정하였다. 그리고, TLM 알고리즘을 구성하여 여러 가지 중재 방식과 비교하여 성능을 분석 및 검증하였다.

Ⅱ. 시뮬레이션 및 성능분석

효율적인 버스 중재 방법을 위해서는 마스터의 대역폭, 스타베이션, 고성능, 사용자의 특별한 요구 등을 고려해야 한다. 첫 번째 마스터의 대역폭의 경우엔 현재 해결 방안으로 fixed priority, TDMA, Lottery 방식 등이 있다.[9] 이는 각 마스터에게 우선순위를 정해줌으로써 대역폭을 조절할 수 있다. 두 번째 스타베이션에 관한 경우, 해결 가능한 중재 방법으로는 Latency-award bus arbitration 방식과 round-robin 방식이 있으며, Latency-award arbitration 방식의 경우엔 마스터가 일정한 대기시간을 넘을 경우 그 마스터의 우선순위를 높여줌으로써 스타베이션을 방지할 수 있다. 세 번째 고성능에 관한 경우는 슬레이브 레이턴시나 버스트 길이에 대해 우선순위를 할당함으로써 성능이 향상될 수 있다. 슬레이브 레이턴시는 슬레이브의 종류에 따라 다르기 때문에, 속도가 빠른 SRAM을 다른 메모리보다 우선순위를 높임으로써 성능이 향상될 수 있으며, 버스트 길이의 경우는 버스트 길이가 길수록 우선순위를 높여줌으로써 각 마스터의 데이터 처리가 끝난 후에 발생하는 레이턴시를 줄일 수 있어 성능을 높일 수 있다. 네 번째 사용자의 특별한 요구에 대해서는 현재 데이터 크기(바이트, 워드 등)에 따라 우선순위를 할당할 수 있다.

일반적으로 네 가지 고려사항에 대한 각각의 해결책은 있으나, 모두를 한 번에 해결할 중재 방법은 아직까지 제안되지 않았다. 이에 본 연구는 우선순위를 점수화하고, 합산하여 종합 점수에 의해 버스 점유권을 부여하는 스코어 중재 방식을 제안한다.

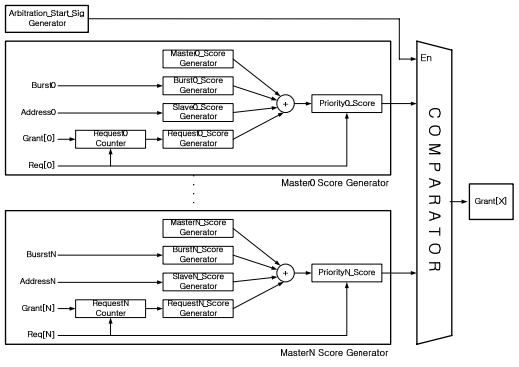

그림 1. 스코어 중재 방식
Fig. 1 Score arbitration method.

그림 1는 스코어 중재 방식을 보여주고 있다. 아비터 내부에 각 마스터 당 점수를 계산하는 *MasterN Score Generator* 가 있다. 이에 버스요청신호 *Req[N]* 이 입력되면, 각 마스터는 해당 마스터의 스코어를 계산한다. 또한 비교기를 통해 점수가 높은 마스터에게 버스 점유권 Grant[X] 신호를 준다.

$$S_{Tot} = S_{MP} + S_{SL} + S_{LL} + S_{DL} + S_{SR} \quad (1)$$

식 (1)은 표 1의 내용을 포함하여 버스 중재 우선순위에 대한 전체 스코어를 합산하는 방법을 보여준다. 여기서 STot는 버스 중재 우선순위에 대한 전체 스코어, SSL는 슬레이브 레이턴시에 대한 스코어, SLL는 장시간 버스 요청하는 마스터에 대한 스코어, SDL는 데이터 길이에 대한 스코어, SSR는 사용자의 특별한 요구에 대한 스코어이다.

표 1. 고려사항에 대한 점수 할당
Table. 1 Score assignment about the consideration item.

연번	목록	스코어 할당
1	Master	Master0 : 6 score Master1 : 5 score Master2 : 4 score Master3 : 2 score Master4 : 1 score Master5 : 0 score
2	Starvation	100 cycle 이상 마스터 대기시 : 10 score
3	성능	1. 슬레이브 레이턴시 짧은 슬레이브 레이턴시 : 4 score 2. Burst length 16 Burst length :4 score 8 Burst length : 3 score 4 Burst length : 2 score 1 Burst length : 1 score
4	데이터	데이터 크기 Word : 2 score Else : 0 score

표 1는 스코어 중재 방식을 위해 실제 적용된 각 고려사항에 대한 점수의 예를 보여주고 있다. 첫 번째, 마스터의 대역폭을 보면 마스터 Master0의 경우는 6점, Master1은 5점, Master2는 4점, Master3은 2점, Master4는 1점, Master5는 0점으로 총 6개의 마스터에 각각의 점수가 할당되어 있다. 두 번째 고려사항인 스타베이션의 경우는 버스 요청 사이클이 100이상일 때, 10점이 상승한다. 세 번째의 경우는 고성능의 경우인데, 이때에는 두 가지의 기준으로 점수가 할당된다. 첫 번째 기준은 슬레이브 레이턴시이다.

즉, 짧은 슬레이브 레이턴시인 SRAM (Static Random Access Memory)의 경우는 4점의 가산점이 부여되며, 그 외의 슬레이브인 SDRAM, DRAM의 경우는 0점이 부여된다. 네 번째 고려사항은 사용자의 요구로서 데이터 크기가 워드일 경우엔 2점의 점수가 부여된다.

표 2. 스코어 중재 방식의 알고리즘
Table. 2 Algorithm of Score arbitration method.

```
1: Priority_Score_Arbitration ()
2: {
3: int Priority_Score              [master_number];
4: int Master_Score                [master_number];
5: int Long_Wait_Score             [master_number];
6: int Burst_Score                 [master_number];
7: int Slave_Score                 [master_number];
8: int first_req=0;
9: int pre_granted;
10:// Priority Score Calculation
11:for(i=0;i<master_number;i++)
12:{
13:  // Master_Score Assignment
14:  Master_Score[i] = master_number-i;
15:  // Long Wait Score Assignment for Starvation Prevention
16:  if(request_cycle[i]>=100) Long_Wait_Score[i] = 10;
17:  else Long_Wait_Score[i] = 0;
18:  //Burst Length Score Assignment
19:  if(burst[i]==SINGLE)        Burst_Score[i] = 1;
20:  else if (trans_cycle[i]==4) Burst_Score[i] = 2;
21:  else if (trans_cycle[i]==8) Burst_Score[i] = 3;
22:  else if (trans_cycle[i]==16)Burst_Score[i] = 4;
23:  // Slave Score Assignment
24:  if(slave_type[i]==SDRAM) Slave_Score[i] = 0;
25:  else                                Slave_Score[i] = 4;
26:  // Total Score Summation
27:  Priority_Score[i]=Master_Score[i]+Long_Wait_Score[i]+Burst_Score[i]+
     Slave_Score[i];
28:} //for(i=0;i<master_number;i++)
29:// Bus Grant Decision
30:for(i=0;i<master_number;i++)
31:{
32:  if(req[i])
33:  {
34:    if(first_req!=0) first_req=1;
35:    else         first_req=0;
36:    if(first_req==0)
37:    {
38:              first_req=1;
39:              pre_granted=i;
40:    } //if(first_req==0)
41:    else
42:    {
43:      if(Priority_Score[pre_granted]<Priority_Score[i]) pre_granted=i;
44:    } // else
45:  } //if(req[i])
46:} //for(i=0;i<master_number;i++)
47:// Bus Grant Assignment
48:granted[pre_granted]=TRUE;
49:} //Priority_Score_Method ()
```

본 연구에서는 일반적인 우선순위의 기준을 고려하여 표1와 같은 점수할당으로 성능을 분석하였으나, 시스템이나 사용자의 요구에 의해 점수 가중치를 변경할 수 있다.

표 2은 이 절에서 논한 스코어 중재 방식의 알고리즘을 보이고 있다. 알고리즘을 구성할 때 조건은 표 1의 기준을 적용하였다. 각각의 마스터에 따른 우선순위는 변수 *Master_Score[master]*에 할당되었고, 스타베이션 방지를 위해 100 사이클 동안 버스점유를 받지 못한 마스터에게 변수 *Long_Wait_Score[master]*에 10점을 할당하였다. 또한 버스트 크기에 따라 버스트 점수 *Burst_Score [master]*를 할당하였다. 또한 슬레이브의 종류(SDRAM, SRAM, 레지스터)에 따라 슬레이브 점수 *Slave_Score [master]*를 할당하였다. 마지막으로 모든 점수를 합산하여 변수 *Priority_Score[master]*에 최종점수를 할당하였다.

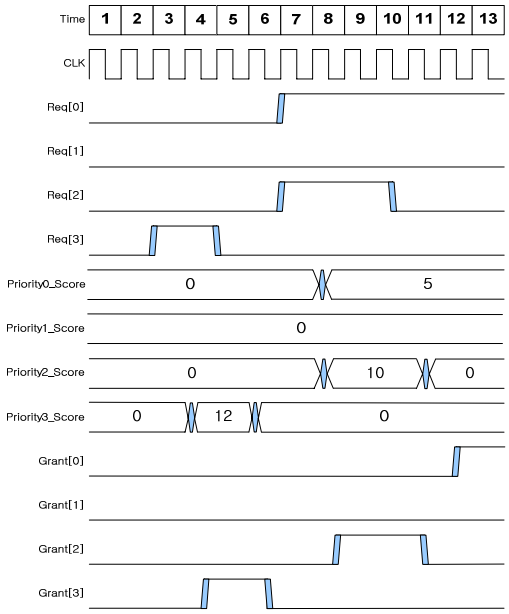

그림 2. 스코어 중재 방식의 타이밍 다이어그램
Fig. 2 The timing diagram of Score arbitration method.

아비터는 전체 점수인 *Priority_Score[master]* 크기를 비교하여 버스 허가 신호 *grant[master]*를 해당 마스터에게 주어 버스를 이용할 수 있는 마스터가 선택되어진다.

그림 2는 스코어 중재 방식의 타이밍 다이어그램이며, Priority_score의 점수값은 임의의 값으로 표기하여 버스 점유를 받는 예를 표현하였다. 시간 3에서 마스터3에 의해 요청된 *Req[3]* 신호에 의해 *Grant[3]* 신호가 할당된다.

이 경우 마스터3만 버스요청을 하였으므로 스코어와 무관하게 *Grant[3]* 신호가 할당되었다. 그러나 시간 7에서 마스터 0과 마스터2가 동시간대에 버스 요청을 하였으며, 각각의 스코어는 10과 12점이다. 이 경우에는 스코어가 큰 마스터2가 버스 허가 신호를 먼저 받고, 마스터2가 처리 완료된 후, 마스터1에 버스 허가 신호가 할당된다. 본 논문은 이와 같이 버스 점유권을 줌으로써, 여러 가지 중요한 요소들을 모두 고려하는 중재 방식을 제안한다.

표 3. 마스터에 따른 버스트길이, 슬레이브 타입 설정
Table. 3 Burst length and slave type due to each master.

No.	Master	Burst Length[cycle]	Slave Type
1	Master0	Single(1), Burst(4, 8, 16) (Using random function)	SDRAM, SRAM (Using random function)
2	Master1	16	SDRAM
3	Master2	16	SRAM
4	Master3	Single(1), Burst(4, 8, 16) (Using random function)	SDRAM, SRAM (Using random function)
5	Master4	1	SDRAM
6	Master5	16	SRAM

일반적으로 각각의 마스터가 전송하는 버스트 길이와 슬레이브는 미리 결정된다. 본 연구에서는 성능검증 시뮬레이션을 위해 각각의 마스터가 전송하는 버스트 길이와 슬레이브 타입을 표 3과 같이 설정하고, 시뮬레이션을 수행하였다. 예를 들어 마스터 2의 경우는 버스트 길이가 16이며, 슬레이브 타입은 SRAM이기 때문에, 버스트 길이 점수 4점과 슬레이브 타입점수 4점을 합하여 스코어는 8점의 가산점이 부여되는 것이다.

또한 최종 사이클(*Final_Cycle*)은 충분히 긴 시간인 10,000,000 사이클로 하여 정확도를 높였으며, 마스터에서 발생시키는 데이터 트랜잭션 사이의 간격은 평균값을 20으로 하였고 랜덤함수로 0-40까지의 값이 발생하도록 하였다.

그림 3은 각각의 중재 방식에 따른 데이터 전송량을 보여주고 있다. Fixed priority 방식의 경우는 마스터 M0의 우선순위가 마스터 M1보다 높음에도 데이터 전송량은 마스터 M1과 M2가 더 높게 나타났다. 이는 마스터의 버스트 길이가 마스터 M1, M2가 마스터 M0보다 크기 때문에 실질적인 데이터 전송량이 많은 이유이다.

특히, 제안한 스코어 중재 방식의 경우는 마스터 M2가 상대적으로 데이터 전송량이 많다. 이는 마스터 M2의 조건 중 슬레이브 타입이 SRAM이기 때문에 속도가 빠른 SRAM에 큰 점수가 할당되었기 때문이다. 또한 이 부분에서 전체적인 성능 향상이 이루어졌다.

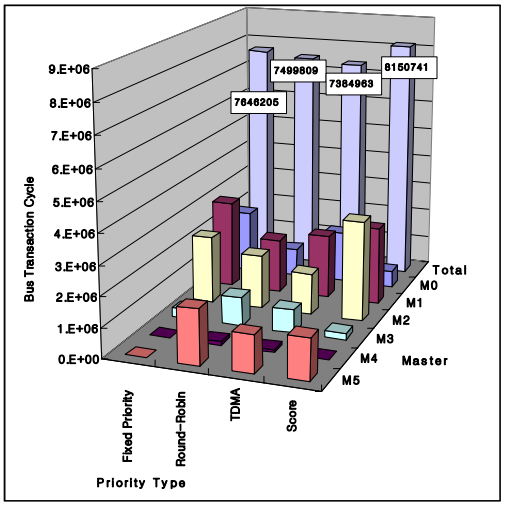

그림 3. 중재방식에 따른 데이터 전송량
Fig. 3 Data transaction due to arbitration methods.

결국, 일반적인 중재방식인 fixed priority 방식과 round-robin 방식 그리고 TDMA 방식의 대략적인 데이터 전송량은 7,100,000-7,700,000 사이클이었으나 우리가 제안한 스코어 중재 방식의 경우는 약 8,150,000사이클이었다. 결국, 스코어 중재 방식이 다른 방식보다 약 6-12%의 성능이 향상되었음을 알 수 있다.

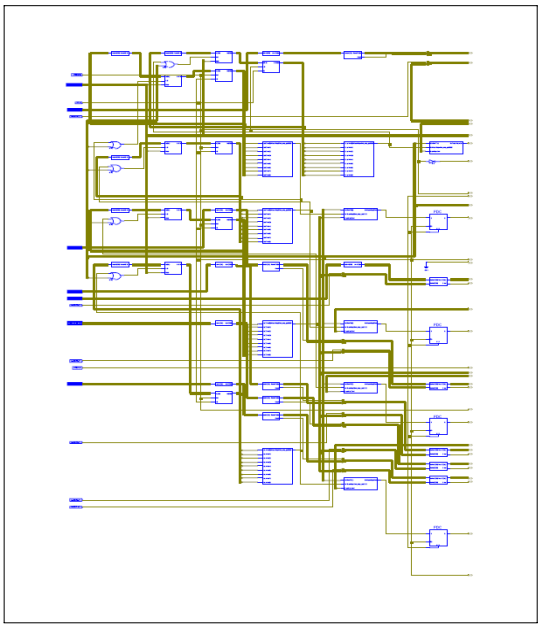

그림 4. 스코어 버스중재 블록 합성 네트리스트
Fig. 4 Synthesis netlist of score bus arbitrtaion block.

표 4. 최대 주파수와 게이트 카운트
Table. 4 Max frequency and gate count.

버스중재방식	최대주파수	게이트 카운트
Fixed Priority	862MHz	157.56
Round-Robin	847MHz	210.10
TDMA	862MHz	259.62
SCOREBUS (proposed)	847MHz	1004.37

마지막으로, 제안하는 스코어 중재방식의 타이밍 마진과 디자인 오버헤드를 파악하기 위해서, Verilog RTL 설계를 수행하고, 하이닉스 0.18um 공정기술을 맵핑하여 Synopsys Design Compiler로 합성하였다. 그 결과 합성 네트리스트는 그림 4와 같으며, 각각의 버스중재방식에 따른 최대주파수는 표 4에서 보듯이 최대주파수의 차이는 847MHz에서 862MHz로 미미한 수준이었다. 그리고, 디자인 오버헤드를 의미하는 게이트 카운트의 경우, 스코어 중재방식이 새로운 scheme이 추가되어 1000 개 정도 되었으나, 실제 칩크기 증가는 무시할 정도 수준이었다.

Ⅲ. 결 론

본 논문에서 우리는 TLM 알고리즘을 구성하고 새로운 중재방식인 스코어 중재 방식을 제안하고, 성능을 분석하였다. 스코어 중재 방식은 다른 중재 방식과는 다르게 각각의 마스터의 우선순위를 나눌 때, 필요한 기준들을 점수로 할당하여 우선순위를 구분하였고, 또한 우선순위가 점수 할당에 따라 수시로 변화할 수 있게 구성하였다. 결국 스코어 중재 방식의 장점은 fixed priority 방식에서 문제시 되는 스타베이션을 극복하였고, 또한 약 6-12%의 시스템 성능을 향상시켰다. 이는 본 스코어 중재 방식이 버스 시스템의 성능과 효율성 측면에서 기존의 다른 중재 방식보다 앞서고 있음을 말해준다. 마지막으로 스코어 중재방식을 설계하여 최대주파수와 칩사이즈를 측정하였는데, 다른 버스중재 방식과 비교하여 주파수 감소와 칩크기 증가는 무시할 수준임을 알 수 있었다.

참고문헌

[1] K. Lee and Y. Yoon, “Architecture Exploration for Performance Improvement of SoC Chip Based on AMBA System”, ICCIT, pp.739-744, 2007.

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

[3] L. N. Bhuyan, ″Analysis of interconnection networks with different arbiter designs″, J.Parallel Distrib. Comput., vol.4, no.4, pp.384-403, 1987.

[4] J. G. Delgado-Frias and R. Diaz, ″A VLSI self-compacting buffer for DAMQ communication switches″, in Proc. IEEE 8th Great Lakes Symp. VLSI, pp.128-133, Feb. 1998.

[5] A. Bystrov, D.J .Kinniment and A. Yakovlev, ″Priority Arbiters″, in Proc. IEEE 6th internation Symp. ASYNC, pp.128-137, April. 2000.

[6] 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.

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

[8] 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.

[9] 이국표, 윤영섭, 대한전자공학회, 전자공학회논문지-SD, 제46권 SD편 제2호 2009.2, pp. 50~56

저자소개

이국표(Kookpyo Lee)

한국해양정보통신학회논문지
제12권 제3호 참조

고시영(Siyoung Koh)

한국해양정보통신학회논문지
제12권 제3호 참조