논문 2010-47SD-1-7

효율적인 버스점유율 관리를 위한 새로운 하이브리드 버스 중재방식의 제안

(Proposal of a Novel Hybrid Arbitration Policy for the Effective Bus Utilization Control)

이 국 표*, 윤 영 섭*

(Kook-Pyo Lee and Yung-Sup Yoon)

요 약

우리가 제안한 새로운 하이브리드 버스 중재 방식은 기존의 Fixed priority 방식과 Round-Robin 방식을 혼재한 것으로 고정된 우선순위로 인한 버스 우선권 독점현상을 방지하고 각 마스터에 효율적으로 버스 우선권을 할당한다. 제안한 중재 방식과 기존의 방식들은 verilog와 하이닉스 0.18um 공정 라이브러리를 이용하여 합성하고, 게이트 카운트와 면적 용적을 비교함으로써 검증하였다. 성능 분석 결과, 우리가 제안한 중재방식이 기존의 방식들보다 성능이 우수하고, 버스 점유율의 효율적인 관리가 가능함을 확인할 수 있었다.

Abstract

We propose the novel Hybrid bus arbitration policy that prevents a priority monopolization presented in fixed priority and effectively assigns a priority to each master by mixing fixed priority and round-robin arbitrations. The proposed arbitration policy and the others were implemented through Verilog and mapped the design into Hynix 0.18um technology and compared about gate count and area overhead. In the results of performance analysis, we confirm that our proposed policy outperforms the others and effectively controls the bus utilization.

Keywords: arbitration, bus architecture, SoC

I. Introduction

We have lived in portable electronic products in our modern societies. All kinds of electronic products like cell phone, PDA, MP3 and electronic games have deeply penetrated our daily life, caused by the development of electronic communication. The one of important electronic communication parts is SoC (System on a Chip) having several masters, slaves, arbiter and decoder in bus architecture. Masters initiate the data transactions like CPU, DMA and DSP, and slaves response the data transactions like SRAM, SDRAM and register. When several masters request a permission to use the shared bus having the attribution that can not process the multiple master data concurrently, arbiter interfaces between shared bus and masters. Fixed priority, round-robin, TDMA (Time Division Multiple Access) and LOTTERYBUS schemes are developed in general arbitration policies, which lead the effectiveness of bus use on shared bus.[1~4]

In the fixed priority policy, each master has a previously defined priority. The master having top

* 정회원, 인하대학교 전자공학과
(Dept. of Electronics Engineering, Inha University)
접수일자: 2009년7월3일, 수정완료일: 2009년11월27일

priority first gets a bus grant. If the top priority master does not request a bus grant, the next highest priority master get a bus grant. It is difficult for lower priority masters to get a bus grant in the fixed priority policy. As the master having the lowest priority always loses in the bus grant competition with the other masters, it falls in the starvation phenomenon that a master cannot get a bus grant for long time.

TDMA and LOTTERYBUS policies are recently developed and can control a master priority using slot number and probability respectively. However, these policies have the difficulty to obtain the expectable bus utilization. (The bus utilization of these policies is described in section 3.)

In this study, we develop the novel Hybrid bus arbitration policy that prevents a priority monopolization presented in fixed priority and effectively assigns a priority to each master. Moreover, it is possible for the proposed policy to control the bus utilization effectively, compared with TDMA and LOTTERYBUS policies.

Our concept is that similarly operated masters or same priority masters are grouped as round-robin scheme and given same priority. Figure 1 shows the example of proposed Hybrid arbitration. In this example, master M1~M2 and M3~M5 are assigned to same groups that are divided by fixed priority respectively. The master number of each group can easily be modified by a master priority.

It is simple to implement only due to mixing fixed priority and round-robin arbitrations. Furthermore, we find that this policy outperforms the others beyond expectation, described in section 3.

The key contributions of this paper are summarized as follows. First, in order to analyze the arbitration schemes, we make the TLM (Transaction Level Model) method of bus architecture that is recently used to explore the bus architecture deeply described in section 2.[5] Second, we compare the max frequency and the design overhead of various arbitrations. Third, the Hybrid arbitration policy is described in detail and we evaluate the performance of Hybrid arbitration compared with the other arbitration methods. Finally, we analyze the effectiveness of proposed arbitration policy.

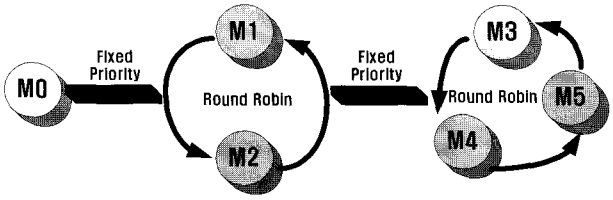

그림 1. 하이브리드 버스 중재 방식의 예시
Fig. 1. Example of Hybrid bus arbitration policy.

II. Model for performance analysis

In order to implement the TLM (Transaction Level Model) method of bus architecture, we make the algorithm shown in table 1. There are InitSt, TransferGenSt, ArbitrationSt and TransferExecSt states in our normal bus architecture model. InitSt is the initial state that the bus requests of masters do not happen. When a master tries to initiate the data transaction, our state moves on TransferGenSt where all kinds of bus signals related with bus communication are created. Next state is ArbitrationSt that the master assigning a bus grant is selected by using the previously defined arbitration policy. Next, our state moves on TransferExecSt state that the data of selected master communicate on the shared bus. After data communication is finished, our state moves on InitSt, TransferGenSt or ArbitrationSt state due to the bus request of master. It directly moves on ArbitrationSt in the case that the bus request signal Req[master] is '1', If Req[master] and idle cycle idle_cycle[master] are all '0', TransferGenSt state is selected. For the residue, it moves on InitSt state. Table 1 gives a full detail of our normal bus model operation.

In this paper, we apply this bus model to AMBA system that has been used in more than 50% embedded system of all over the world. [1] The data length of transaction can be random changed to 1, 4,

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

```
1: 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[allmasters]--)do
11:         Cur_Cycle++
12:         Detail_Cycles_Cal(NULL,NULL,InitSt)
13:      endwhile
14:      Execute_Bus_Model(Master_Signal[all masters],NULL,InitSt)
15:      State=TransferGenSt
16:   endif
17: elseif(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: endelseif
24: elseif(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:   endwhile
31:   State=TransferExecSt
32: endelseif
33: elsethen//StateisTransferExecSt.
34: while((Master_Signal.Data_Size[selectedmaster]+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:   endwhile
39:   Master_Signal.Req[selected master]=False
40:   Master_Signal.Granted[selected master]=False
41:   if(Master_Signal.Req[somemasters]==True)State=ArbitrationSt
42:   elseif(Master_Signal.Idle_Cycle[somemasters]==0)State=TransferGenSt
43:   elseState=InitSt
44:   endif
45: endelse
46: endbus_model(Arbitration_Type)
```

8 and 16. The idle cycle between the data transactions of master is also changeable using random function that is operated with the range and the mean values of idle cycle. The model of SDRAM controller is used as slave component. SDRAM has relatively long latency due to pre-charge, refresh, row/column access cycles and so on.

Ⅲ. Results and discussion

1. Comparison of max delay and gate count

In order to compare timing margin and design overhead of proposed Hybrid arbitration with those of the others, the various arbitration blocks were implemented through Verilog and mapped the design into Hynix 0.18-m technology through the Synopsys Design Compiler. However, LOTTERYBUS arbitration policy is not implemented through Verilog, as we already know that LOTTERYBUS policy is enough complicated and require more gate count, using the table of random function[2].

The max delay due to each arbitration block along the bus is shown in table 2. The max delay difference between arbitration policies is not wide from 1.12ns to 1.18ns. In the synthesized netlists, the various arbitration blocks have around 158~260 logic gate counts. However, the area overhead is still small considering the typical area of modules in SoC designs.

Finally, we find that it is simple and easy to implement our Hybrid arbitration block and its design overhead is rather small in comparison with the other arbitration block.

표 2. 최대 지연시간과 게이트 카운트
Table 2. Max delay and gate count.

Arbitration block	Max delay	Gate count
Fixed Priority	1.16ns	157.56
Round-Robin	1.12ns	210.10
TDMA	1.18ns	259.62
Hybrid (proposed)	1.12ns	219.84

2. Performance analysis

The goals of our simulation are to evaluate the effectiveness of five arbitration protocols based on the both bus utilization and bus request cycle of each master. Our simulation model consists of six master modules, four slave modules and one shared bus. In

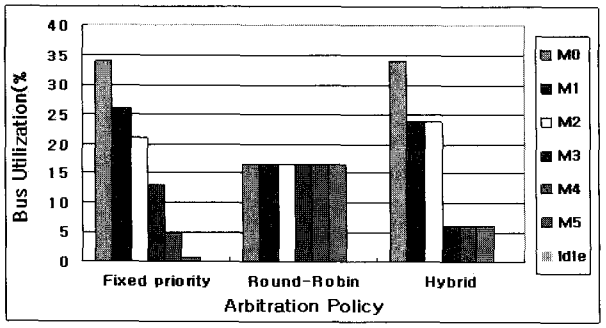

그림 2. fixed priority, round-robin 방식과 하이브리드 버스 중재 방식의 버스 점유율 비교

Fig. 2. Bus utilization of fixed priority and round-robin, compared with Hybrid arbitration.

order to accurately confirm the results, final simulation cycle is set to 10,000,000. In figure 2, each master has a previously defined priority in the order named such as M0, M1, M2, … , M5 in fixed priority. As the bus utilization of master M5 is less than 1%, we assume that master M5 falls in the starvation phenomenon that the lowest priority master is locked out from the bus. Furthermore, as the bus utilization of master M0 is around 30 times as large as that of master M5, we confirm that it is difficult to achieve the fairness in fixed priority. The bus utilization of each master is same as around 17% in round-robin without a priority. On the other hand, in Hybrid policy shown in figure 1, as M1~M2 masters and M3~M5 masters are grouped as round-robin, these utilizations are almost same as

표 3. LOTTERYBUS 방식의 티켓 확률과 TDMA 방식의 슬롯 넘버 수

Table 3. Slot number of TDMA and ticket probability of LOTTERYBUS.

Case	Slot number			Ticket probability		
	Master M0	Master M1,M2	Master M3,M4, M5	Master M0	Master M1,M2	Master M3,M4, M5
Case1	3	2	1	3/10	2/10	1/10
Case2	5	2	1	5/12	2/12	1/12
Case3	5	1	1	5/10	1/10	1/10
Case4	7	2	1	7/14	2/14	1/14

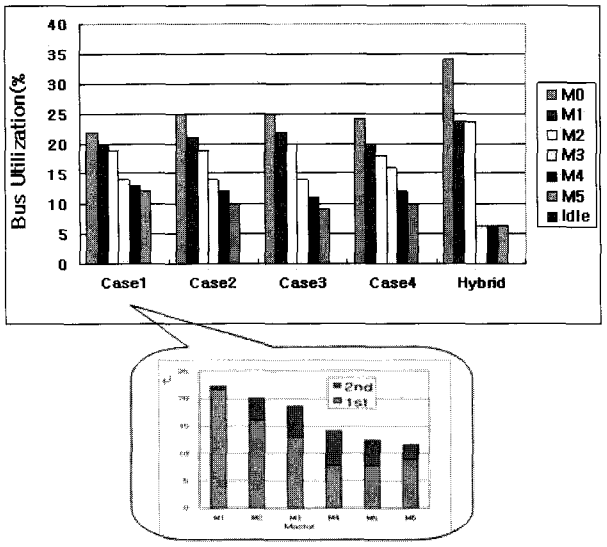

그림 3. TDMA와 하이브리드 버스 중재 방식의 첫 번째, 두 번째 중재에 따른 버스 점유율 비교

Fig. 3. Bus utilization of TDMA, compared with Hybrid arbitration containing utilization of second and first arbitration.

around 24% and 6% respectively. Conclusively, the proposed Hybrid policy prevents a starvation phenomenon happening in fixed priority, due to grouping the M3~M5 masters that have lower priority. Furthermore, the utilizations of master M0 having a top priority can be maximized as 34%, like fixed priority.

TDMA policy provides each master with the time slot. For our TLM simulation about TDMA policy, the slot numbers of each master from M0 to M5 are variously set, shown in table 3. In order to be compared with Hybrid policy, slot numbers are assigned due to the master group used in Hybrid. Bus utilization is not matched with the slot number shown in figure 3, which is caused by the second-level arbitration happening in case that the master of current slot does not request a bus grant. [2] For instance, though we set same slot number in order to give M3~M5 masters an equal priority, the difference of bus utilization is widely presented from 8% to 20%. Moreover, the utilizations of master M0 having a top priority are only from 22% to 25%, in comparison with 34% of Hybrid policy. The maximum utilization difference between masters

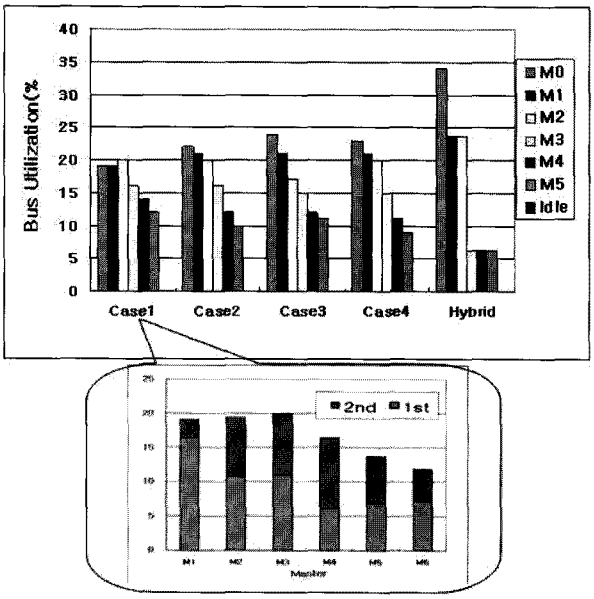

그림 4. LOTTERYBUS와 하이브리드 버스 중재 방식의 첫 번째, 두 번째 중재에 따른 버스 점유율 비교

Fig. 4. Bus utilization of LOTTERYBUS, compared with Hybrid arbitration containing utilization of second and first arbitration.

having the highest and the lowest priorities is only 14% in TDMA, in comparison with 28% of Hybrid policy.

Next, we analyze the LOTTERYBUS policy that a bus grant is divided statistically using a LOTTERYBUS ticket. LOTTERYBUS policy is proposed as the improvement of TDMA scheme.[2] The ticket probabilities of each master from M0 to M5 are variously set such as TDMA, due to table 3. Figure 4 shows the bus utilization of LOTTERYBUS policy that is similar to that of TDMA. Bus utilization is not matched with the ticket probability due to second-level arbitration and the maximum utilization difference between masters having the highest and the lowest priorities is only 14%.

From the results of figure 3 and figure 4, we confirm that Hybrid policy effectively controls a priority according to master's importance, compared with TDMA and LOTTERYBUS policies.

Figure 5 shows the average bus request cycle per transaction. The bus request cycle means the wait cycle until a bus grant is received. Generally, the

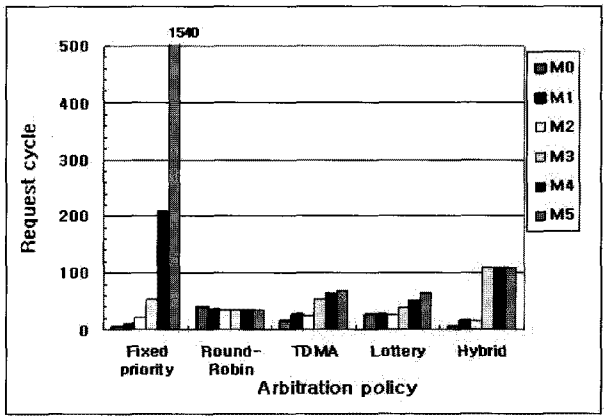

그림 5. 다양한 중재 방식들의 평균 요청 사이클

Fig. 5. Average request cycle vs. various arbitration policies.

request cycle of higher priority masters is smaller than that of lower priority masters. The difference of request cycle due to master priority is definitely presented in fixed priority shown in figure 5.

Though the request cycle of master M1 is only 6, that of master M6 is 1540. It means that master M6 falls in starvation phenomenon hardly getting a bus grant for long time. The other arbitration policies present relatively smaller request cycle. Specially, though the maximum request cycle of Hybrid policy is around 110, we consider the reasonable results caused by the utilization of M3~M5 masters having the lowest priority.

Ⅳ. Conclusions

To improve the utilization of conventional arbitration policies, we propose the novel Hybrid bus arbitration and analyze its performance. The experiment results of proposed arbitration are discussed in previous section and are conclusively summarized in table 4. Finally, we confirm that proposed policy outperforms the others and can be applied for the effective bus utilization control.

표 4. 하이브리드 버스 중재 방식과 다른 방식들의 비교 결과
Table 4. Compared results of Hybrid policy and the others.

Compared item	Hybrid Policy	Fixed Priority	Round-Robin	TDMA	LOTTERY BUS
Easy to design	○	○	○	△	X
Timing margin and design overhead	○	○	○	○	X
Giving same bus utilization to same priority groups	○	X	○	X	X
Protecting starvation phenomenon	○	X	○	○	○
Maximizing the difference of bus utilization about each master	○	○	X	X	X
Minimizing bus request cycle	○	X	○	○	○

Reference

[1] ARM, Limited. AMBA Specification, 1999.
[2] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "The LOTTERYBUS On-Chip Communication Architecture," IEEE Trans. VLSI Systems, vol.14, no.6, 2006.
[3] 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.
[4] Chiung-San Lee, "High-Fair Bus Arbiter for Multiprocessors," IEICE Trans. Inf. & Syst., vol.E80-D, no.1, 1997.
[5] K. Lee and Y. Yoon, "Architecture exploration for performance improvement of SoC chip based on AMBA system," ICCIT, pp.739-744, 2007.

저 자 소 개

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

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