LETTER

Jitter-Conscious Bus Arbitration Scheme for Real-Time Systems

Jong-Ho ROH[†], **Minje JUN**[†], **Kwanhu BANG**[†], *Nonmembers*, *and* **Eui-Young CHUNG**[†a)], *Member*

SUMMARY Jitter is the variation of latencies, when real-time Intellectual Properties (IPs) are accessing data from the data storages. It is a critical factor for such IPs from the Quality-of-Service (QoS) perspective. Jitter of a real-time IP can be measured by how frequently it experiences the underflows and overflows from its data queue in read mode and write mode, respectively. Such failures critically depend on the bus arbitration scheme which determines the bus acquisition order of IPs. The proposed idea allows IPs to inform the bus arbiter of the status of their data buffers when they assert bus requests. Such information helps the bus arbiter to determine the bus acquisition order while greatly reducing the jitter. The experimental results show that our method effectively eliminates the overflows and underflows of real-time IPs by dynamically preempting the jitter-critical bus requests.

key words: *jitter, QoS, arbitration, queue, real-time*

1. Introduction

Many System-on-Chips (SoCs) for multimedia applications include real-time constrained IPs such as the controllers for video streaming and display. For these IPs, constant rate of data consumption or production is important, since their behavior is periodic in nature. Real-time IPs have internal queues (FIFOs) to be robust to the variation of the data transfer latency called jitter. For a large variation of the data transfer latencies, the queue can be overflow in write mode and underflow in read mode. The variation is mainly due to the bus, since it is shared by multiple IPs competing with each other for the bus acquisition. The proper bus arbitration scheme should allocate sufficient bandwidth to each IP without causing any overflows or underflows for each real-time IP. But it is hard to satisfy both constraints when the IPs produce heavy workloads dynamically. In real-time systems, the overflow or underflow of real-time IPs due to the jitter is more severe than the insufficient bandwidth from a QoS perspective, since the former is translated to the failure of completing the given mission, while the latter is usually translated to the performance degradation. The proposed method effectively trades off these two factors by monitoring the quantized queue length of IPs representing the urgency of overflow or underflow. More precisely, our method arbitrates the bus requests from the bandwidth perspective unless there is a jitter-critical request, but it changes its arbitration policy when it observes a jitter-critical request such that the highest bus acquisition priority is given to the request with the sacrifice of other IPs' bandwidths.

In Sect. 2 we address the previous QoS arbitration schemes and the proposed scheme. In Sects. 3 and 4, we describe our method and its applications, respectively. Finally, the experimental results are shown in Sect. 5 followed by a conclusion in Sect. 6.

Manuscript received April 7, 2008.
Manuscript revised September 9, 2008.
[†]The authors are with Yonsei University, Seoul, Korea.
a) E-mail: eychung@yonei.ac.kr
DOI: 10.1587/transfun.E92.A.643

2. Previous QoS Arbitration Schemes

Many research groups have worked on the QoS-guaranteed arbitration scheme for shared bus architectures. The methods in [1] and [2] aimed at optimizing the average cases rather than the peak cases for the latency as well as bandwidth. In [1], LOTTERYBUS was introduced and it improved the latency problem of Time-Division Multiple Access (TDMA) and Round-robin arbitration schemes by allocating the bandwidth in a statistical manner. It showed outstanding results for high bandwidth IPs with a short latency constraint, but it could not handle efficiently for low (high) bandwidth IPs with a short (long) latency constraint. The methods for real-time IPs are also proposed in [3] and [4]. However, their performance drastically decreases when the total requested bandwidth from the IPs reaches the system bandwidth. The aforementioned techniques focused on the latency satisfaction under the bandwidth constraint. On the other hand, our approach does not consider the latency constraint, but a jitter for fully exploiting the advantage of internal buffers (data queues) of IPs. Hence, our method is more aggressive by redistributing the bandwidth of non jitter-critical IPs (non real-time IPs or real-time IPs with a data queue filled with moderate amount of data) to the jitter-critical IPs (real-time IPs whose queue is almost empty or full).

3. Jitter-Conscious QoS Arbitration Scheme

3.1 Overall Architecture

The proposed jitter-conscious QoS arbitration scheme consists of two blocks — jitter detection block and arbiter block. Figure 1 shows a shared-bus architecture adopting our method, where we denote master IP i as M_i. The arbiter block can be implemented with any arbiters such as fixed-priority, round-robin, and so on. However, our major contribution is on the jitter Detection Block (JDB) which is the core of our method. In Fig. 1, M_1 and M_2 are non real-time IPs whose requests can be delayed without violating

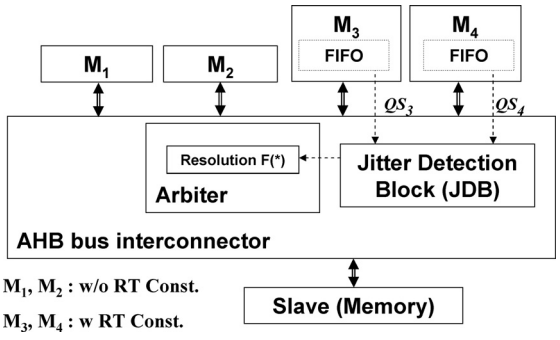

Fig. 1 Jitter-conscious QoS arbiter.

any system requirements. On the other hand, M_3 and M_4 are real-time IPs. Each real-time M_i has additional signals called QS_i to deliver its queue status to JDB which computes the dynamic weight of each request based on QS_i. The larger the dynamic weight is, the more critical the corresponding bus request is from the jitter perspective. Finally, the dynamic weight is delivered to the arbiter, and then the arbiter performs a resolution function which resolves the decision of JDB and the decision of arbiter.

3.2 Queue Status Information

QS_i, the queue status of M_i is computed by the IP itself. Whenever M_i asserts a request, it computes QS_i using Eq. (1) in read operation and Eq. (2) in write operation. $QC_i = \{1, 2, \cdots, QD_i\}$ and QD_i are the current queue length and the queue size of M_i, respectively. Also, N represents the bit width of QS_i.

$$\begin{aligned} QS_i &= \lceil (QD_i - QC_i)/QD_i \times 2^N \rceil \\ &= (QD_i - QC_i - 1) \ll (N - \log_2 QD_i) \end{aligned} \quad (1)$$

$$QS_i = QC_i \ll (N - \log_2 QD_i) \quad (2)$$

In Eq. (1), the lower line expression is a cost-effective version by replacing the division with logical shift operation. Note that the logarithm term is a constant, since QD_i is fixed at design time. In Eq. (1) for read operation, large QC_i means that future data to be consumed by M_i is buffered enough. Hence, large QS_i generates small QD_i, meaning that the jitter urgency is low. On the other hand, they have a opposite relation in Eq. (2) for write operation. The resolution of QS_i is determined by its bit width, N. The finer resolution of QS_i is preferred when many real-time IPs are competing, since the jitter-urgency can be finely distinguished with area increase (larger N).

3.3 Dynamic Weight

We denote the dynamic weight as WD_i for brevity. WD_i is a weighted QS_i to consider the difference of value system between QS_i and the arbiter. Typically, an arbiter prioritize the connected masters using a range of fixed-point numbers. For instance, a fixed-priority arbiter prioritize the masters from 0 to 15 in case AHB bus which allows up to 16 masters to be connected. On the other hand, the range of QS_i is from 0 to 2^N, where N is typically set to 1 or 2 to minimize the bus wiring overhead. If the value range of an arbiter is large, the impact of QS_i on arbitration will be marginal due to the large difference of value range. To compensate such mismatches, we introduce a dynamic weight denoted by WD_i as shown in Eq. (3).

$$WD_i = QS_i \times w \quad (3)$$

where, w scales QS_i to be comparable to the arbiter's value system. To implement Eq. 3 in a simple manner, we limit w to be 2^k, which replaces the multiplication in Eq. (3) by a shift operation. Also, k is determined using Eq. (4).

$$k = \lceil \log_2 V \rceil - \log_2 A \quad (4)$$

where, V is the magnitude of the arbitration value range and A is the maximum number of masters to be connected to the target bus. Hence, w is V quantized by A to scale QS_i for target arbitration scheme.

3.4 Resolution Function

Resolution function is located inside the arbiter. When the arbiter receives WD_i from JDB, it performs the resolution function for its final decision. The resolution function can be defined depending on the characteristic of the arbitration algorithm as shown in Eq. (5).

$$p'_i = F(WD_i, p_i) \quad (5)$$

where, function F is an arbitrary function for the resolution, WD_i is the dynamic weight of M_i. Also, p_i is the priority of M_i given by the conventional arbiter and p'_i is the modified priority of M_i after the resolution function is performed. Note that most of arbitration algorithms have the priority list for the masters connected to the corresponding shared bus. For instance, fixed priority arbitration scheme has a static priority list of masters, while the round robin arbitration scheme has a dynamic priority list of masters which is updated whenever an arbitration is performed. The arbiter selects the final winner among the requesting masters based on p', the modified priorities of IPs. In our work, we define the resolution function by either a multiplication or a addition. Multiplication is appropriate for more tightly constrained real-time systems, while the addition is appropriate for relatively softly constrained real-time systems. Note that the resolution function using multiplication more aggressively forces the arbiter to manage the jitter-critical requests (higher QS_i) with higher priority.

4. Extended Arbitration Schemes for Jitter Reduction

4.1 Jitter-Conscious Fixed Priority Scheme (JCFP)

The conventional fixed priority arbitration scheme (CFP) is one of the most popular arbitration schemes used in on-chip

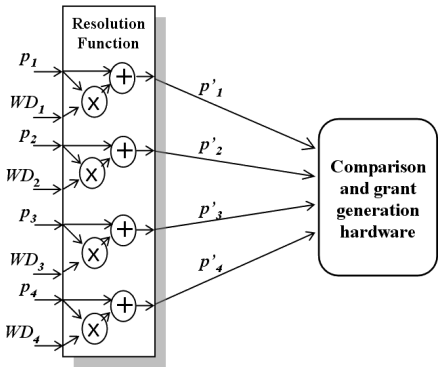

Fig. 2 Jitter-conscious fixed priority scheme.

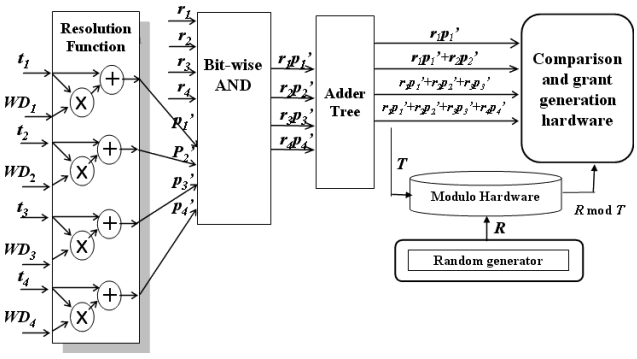

Fig. 3 Jitter-conscious LOTTERYBUS.

bus architectures. The bus access priority is statically defined *a priori*. The bus arbiter grants the bus access right to the master with the highest priority among the requesting masters. The proposed method incorporates with CFP by changing the priority dynamically for the jitter-critical requests based on WD_i. In JCFP case, we set $w = 2^k$ to 1, since k becomes 0 based on Eq. (4), where the value range of a fixed-priority scheme cannot be larger than the maximum number of masters to be connected. If the multiplication is used as a resolution function, it can be represented as shown in Eq. (6).

$$p'_i = \begin{cases} WD_i * p_i & \text{if } WD_i \; != \; 0, \\ p_i & \text{otherwise.} \end{cases} \tag{6}$$

where, p_i is the priority value of M_i in the conventional fixed priority scheme. The condition that $WD_i = 0$ only occurs when $QS_i = 0$, since w cannot be 0 in any case. In other words, M_i is never urgent from the jitter perspective when $WD_i = 0$. In such situation, we let the arbiter operate in its own nature by avoiding the multiplication On the other hand, the resolution function using addition is represented as Eq. (7).

$$p'_i = WD_i + p_i \tag{7}$$

The overall implementation of JCFP is shown in Fig. 2 with a resolution function of Eq. (6), where X represents multiplication and "+" represents "logical or" operation. After computing p_i using Eq. (6), the Comparison and grant generation hardware block selects the winning master based on the p_i. If a master M_i is not in jitter-urgent state then WD_i will be zero and it is acted as the conventional priority scheme. Our method with a fixed priority scheme is especially effective for low bandwidth jitter-critical IPs which are the major problem makers when we adopt CFP, since p'_i can be easily enlarged by QS_i or WD_i compared to the small arbitration value range of CFP.

4.2 Jitter-Conscious LOTTERYBUS (JC-LOTTERY)

The LOTTERYBUS [1] is a probabilistic arbitration algorithm implemented in a lottery manager for the bus arbitration. The major benefit of LOTTERYBUS is bandwidth fairness, since the arbiter can allocates the bus bandwidth to each master by predefined amount in a probabilistic manner using the concept of tickets. For this reason it has a large attention from bandwidth-conscious real-time systems. However, it does not consider the latency issue (and jitter issue) which can be solved with our method proposed in this paper. The lottery tickets acting as the weight are accumulated through the lottery manager in a shared bus architecture. Let $T = t_1, t_2, \cdots, t_n$ is the set of ticket values assigned to each master. Note that t_i represents the ratio of the bandwidth allocated to M_i over the system bandwidth. Let $R = r_1, r_2, \cdots, r_n$ is the set of bus requests. If M_i has a pending request, $r_i = 1$. Otherwise $r_i = 0$. The master M_i is granted with the probability of $P(M_i)$ by favoring components with larger ticket values based on Eq. (8).

$$P(M_i) = \frac{r_i \times t_i}{\sum_{j=1}^{n}(r_j \times t_j)} \tag{8}$$

where, $P(M_i)$ is the granting probability of M_i. Since LOTTERYBUS is a stochastic method, a master with lower $P(M_i)$ may be granted.

For jitter-conscious LOTTERYBUS, our method incorporates with LOTTERYBUS by increasing t_i dynamically for the jitter-critical requests using WD_i. For this purpose, t_i is used as p_i for the resolution function which is defined as Eq. (9).

$$p_i = t_i$$
$$p'_i = \begin{cases} WD_i * p_i & \text{if } WD_i \; != \; 0, \\ p_i & \text{otherwise.} \end{cases} \tag{9}$$

Note that p_i is multiplied by WD_i as in JCFP, but the value range of p_i is much wider than JCFP, since t_i needs to be large enough to consider the given probability precision. For instance, if the probability precision is 0.001, the range of t_i is from 0 to 999. Using Eq. (4), w becomes 64. Other part of the resolution function is similar to that in JCFP. The resolution function can be also implemented with an addition. But it less effective, since the wide value range of t_i.

JC-LOTTERY is implemented as in Fig. 3. After computing p'_i, the Bitwise-AND block filters out the weights of the idle masters in the current cycle. Adder Tree generates

the boundary condition of each request for selecting the winning master based on the number from the random number generator. To summarize, t_i is weighted in our scheme depending on the jitter criticality of requests using WD_i, hence the bandwidth-conscious ticket value is adjusted to minimize the jitter, which is not possible in the LOTTERYBUS.

5. Experimental Results

We implemented the proposed scheme with the cycle-level accuracy using SystemC. The exemplary architecture shown in Fig. 1 is explored in this experiment. Note that M_1 and M_2 are non real-time IPs, while M_3 and M_4 are real-time IPs. The bus protocol is AMBA AHB and the slave is a SDRAM controller with an external SDRAM. We show the experimental results only for read operations, since the results for write operations are similar to those of read operations.

5.1 Result of Jitter-Conscious Fixed Priority Scheme

We compared JCFP to CFP in this experiment. The required bandwidth from each master is shown in Table 1. The priority of masters is in the order of M_4, M_2, M_3, and M_1, since $M4$ and $M2$ require higher bandwidth than the other two. Also, $M4$ has a higher priority than $M2$ due to the real-time constraint. $M3$ has a higher priority than $M1$ for the same reason. Finally, we set N to 2 for the JCFP scheme.

We compared both schemes when the total required workload is varying from 80% to 180% of the system bandwidth as shown in Fig. 4. For the light workload (e.g. 80%), both schemes well allocate the system bandwidth as requested. However, the allocated bandwidth ratio does not match to the required bandwidth ratio as the total workload becomes lager (from 100%) for both cases. It is expected result, since the arbitration scheme loses its control over the system bus due to the heavy workloads beyond the system bus capacity. Nevertheless, there is a big difference between these two schemes. As the input workload increases, CFP allocates more bandwidth to M_4 and M_2, since they have higher priorities than the other two. Also, the other masters (M_3 and M_1) rarely have chances to be granted for the bus, hence M_3 frequently experiences the underflows as shown in Fig. 5. On the other hand, JCFP allocates more bandwidth to M_3 than CFP, while sacrificing the bandwidth allocated to M_2 for protecting jitter violation. For this reason, M_3 rarely violates the given real-time constraint with JCFP. Note that M_1 and M_2 are non real-time IPs, hence underflow or overflow does not mean the system malfunction, but performance degradation. When an underflow occurs for these IPs, they are stalled until the data is ready. On the contrary, the occurrence of underflow for M_3 and M_4 eventually causes the system malfunction due to the constraint violations. Therefore, JCFP performs better arbitration than CFP by favoring real-time IPs.

Even though JCFP outperforms FCP from the jitter perspective, the bandwidth allocation is not satisfiable, since non real-time IPs can significantly degrades the overall system performance due to their long data latencies. The bandwidth fairness issue can be resolved by integrating our scheme with better bandwidth-conscious arbiter. The LOTTERYBUS is one of the bandwidth conscious arbitration scheme and we will demonstrate the experimental results of our scheme which is integrated with LOTTERYBUS rather than CFP in Sect. 5.2.

Table 1 The bandwidth requirements from IPs.

IP	Bandwidth	Real-time
M_1	16.7%	no
M_2	33.3%	no
M_3	16.7%	yes
M_4	33.3%	yes

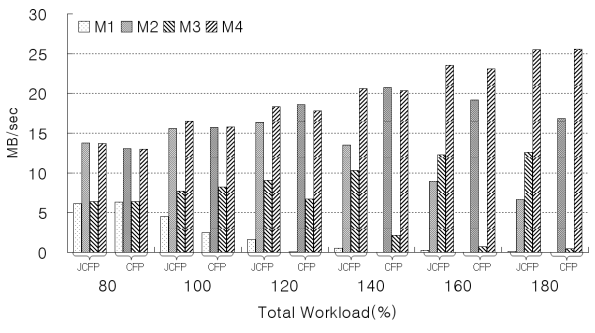

Fig. 4 Bandwidth allocation of JCFP and CFP.

5.2 Result of Jitter-Conscious LOTTERYBUS

The similar comparison was performed for JC-LOTTERY and LOTTERYBUS. The allocated bandwidth and the underflow counts are shown in Fig. 6 and Fig. 7, respectively. First, JC-LOTTERY does not incur any underflows for M_3 and M_4 like JCFP as shown in Fig. 7. Thus, it is shown that our method minimizes the jitter of real-time IPs with any existing arbitration schemes effectively. On the other hand, in Fig. 6, we could observe that the mismatch of allocated bandwidth with JC-LOTTERY and LOTTERYBUS against the required bandwidth has become less critical than JCFP and CFP, respectively. For instance, when the total input workload is 180% of system bandwidth, JC-LOTTERY allocates 3MB/sec and 4MB/sec to M_1 and M_2, respec-

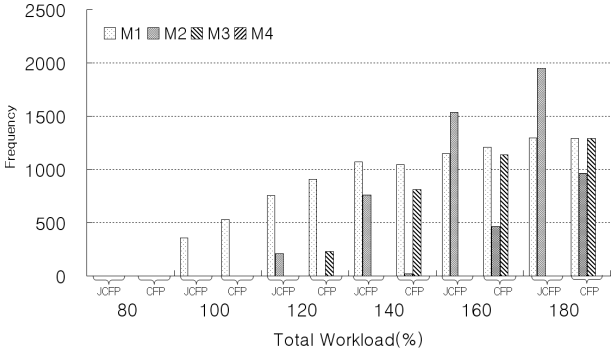

Fig. 5 Underflow counts of fixed priority scheme.

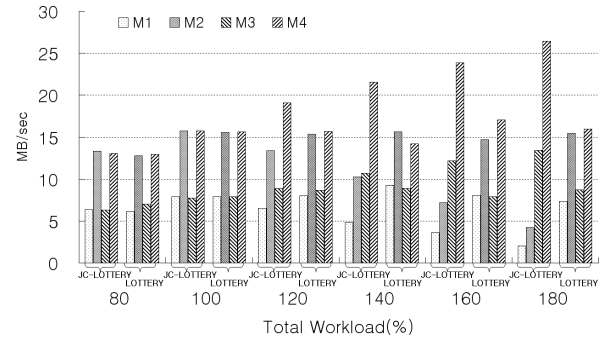

Fig. 6 Result of bandwidth allocation of LOTTERYBUS.

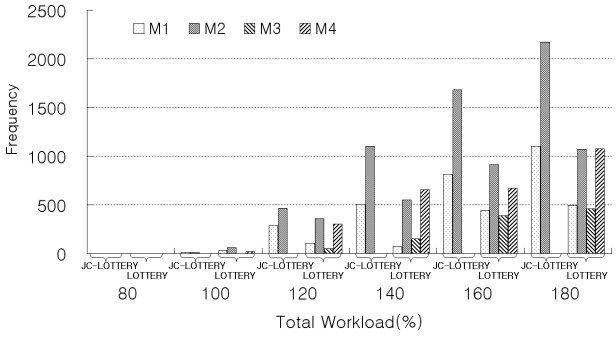

Fig. 7 Underflow counts of LOTTERYBUS.

tively. JCFP, however, rarely allocates bandwidth to M_1 which eventually experiences the starvation. It means that our method becomes more effective when it is integrated with more bandwidth-conscious arbitration scheme by protecting the bandwidth starvation of non real-time IPs while showing the same jitter effect for real-time IPs.

6. Conclusions

We propose an arbitration scheme for minimizing the variation of access latency called jitter. Our method reduces the jitter not to be larger than a certain level so that real-time IPs are free from the failures such as underflow and overflow. Ideally, our method can control the failures as zero even when the input workload requires larger than the system bandwidth.

Acknowledgments

This work was supported in part by "System IC 2010" project of Korea Ministry of Knowledge Economy, by Korea Research Foundation Grant funded by the Korean Government (MEST) KRF-2007-313-D00578, and by IDEC (IC Design Education Center).

References

[1] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "The LOTTERYBUS on-chip communication architecture," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.14, no.6, pp.596–608, June 2006.
[2] W.D. Weber, J. Chou, I. Swarbrick, and D. Wingard, "A quality of service mechanism for interconnection networks in system on chips," DATE 2005, pp.1232–1237, 2005.
[3] B. Lin, G. Lee, J. Huang, and J. Jou, "A precise bandwidth control arbitration algorithm for hard real-time SoC buses," ASP-DAC'07, pp.165–170, Jan. 2007.
[4] C.-H. Chen, G.-W. Lee, J.-D. Huang, and J.-Y. Jou, "A real-time and bandwidth guaranteed arbitration algorithm for SoC bus communication," ASP-DAC'06, pp.600–605, Jan. 2006.
[5] AMBA Specification (Rev 2.0), ARM IHI 0011A.