Arbitration Latency Analysis of the Shared Channel Architecture for High Performance Multi-Master SoC

Jisuhn Suh, Hoi-Jun Yoo
Department of Electrical Engineering and Computer Science,
Korea Advanced Institute of Science and Technology,
Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
Email : jissuh@sipac.org, hjyoo@kaist.ac.kr

Abstract

We propose new analysis method to estimate the traffic performance of communication channel for system-on-chip (SoC). We define the channel utilization ratio, and we analyze the traffic performance of multi-master system-on-chip on shared channel architecture by measure of the arbitration latency. This method is efficient to evaluate the traffic characteristics of the shared channel architecture. And this results offer the methods to optimize the parameter of the components to achieve high performance channel.

To verify the efficiency of this method, we experiment the latency of single shared channel architecture by various conditions of components composing SoC. We simulated the effect of number of masters and 2 types of arbitration algorithm by using defined channel utilization ratio. In this analysis, it is found that the arbitration latency increases with the number of masters and channel utilization ratio. The arbitration algorithm affects the arbitration latency according to the number of masters. The throughput of data transaction is proportional to channel utilization ratio.

1. Introduction

System-on-chip is getting more complex and requires more multi-processor architecture. However the capability for SoC design not increases as we expect. The one method to improve the capability is platform-based design. The central technology of platform is the communication channel architecture. Shared channel architectures have been widely used as communication channel of IP-based SoC. For example, the AMBA and CoreConnect bus are typical shared channel architecture. Shared channel is a bundle of data, address, control lines that are commonly shared by many IPs. The advantages of shared channel architecture are relatively simple controllers and small area because of uncomplicated architecture than point-to-point or network types of channel. However the demerit of shared channel is to have low performance because of relatively un-parallel processing than other types. Therefore the shared channel architecture has reached the limit as more complex of SoC functions and the increase of number of masters. To overcome this obstacle, various architectures are proposed such as multi-layer bus, segmented bus and network architecture [1, 2].

A general architecture for multi-master SoC on the shared channel architecture is shown in Figure 1. The components of this architecture have *n* masters, *m* slaves and *l* channels. Master is a processing element that gives a command to the other processing element for sending or receiving a data. Slave is a processing element that receives command from masters and responds to the command. Channel is a route of data and control signal for interconnection between masters and slaves.

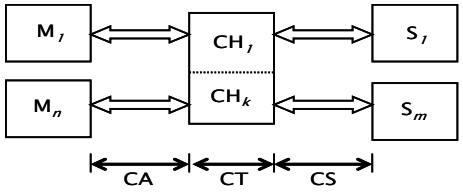

Figure 1. General architecture for multi-master SoC that has *n* masters, *m* slaves and *k* channels

A channel is composed of arbiters for controlling the channel contention among masters, an address decoder for decoding destination, and routes for the data passing.

The performance of a SoC can estimate by analyzing the communication channel architecture. And also the performance of SoC can be improved by adjusting or changing the parameters or components based on analysis of the communication architecture. Therefore the analysis of communication channel traffic is very important. And various methods to analyze the communication channel architecture have been reported for a long time [3, 4, 5]. The previous methods analyzed the performance in the respect of total system such as execution time or total throughput. Therefore the previous methods have made the conclusion that the characteristics of components composing SoC depend on applications. However because how to use the components

can determine the performance of a SoC, more detail analysis about the effect of each components is required. For example, arbitration algorithm was compared with other algorithms by performance metrics of bandwidth and latency, however the analysis how the arbitration algorithms affect directly to the performance of SoC was absent. Just it is reported that arbitration algorithm is closely depends on its application [6, 7]. It is required more analytical evaluation for arbitration algorithms because of increasing of number of masters. The mapping algorithms were evaluated by the comparison of the execution time [5], but the analysis how much the various types of channel architectures directly affect to the performance was not considered. And also it's required how many masters can be mapping to the single shared channel without performance degradation.

2. Proposed analysis method

We propose the analysis method to concentrate on channel utilization and arbitration latency to analyze the shared channel architecture. The data transaction between masters and slaves starts from a request by a master. An arbiter receives request signals from masters and selects one master to use channel by priority policy.

The completion time of data transaction through channel is defined as the time from request to slave access. In Figure 1, the data completion time (CC) is the sum of arbitration time (CA), data transmission time through routes (CT), and access time of slave (CS). Here, we use clock cycle as unit of time.

$$CC = CA + CT + CS$$

CA: arbitration latency which is affected by contention of masters

CT: data transmission time which is affected by operation type, transaction size and channel width

CS: slave access time which is affected by latency of slave and conditions of input/output

CT and CS can be estimated by parameters of components composing system. As a system have more masters, the probability of contention increases. Therefore the arbitration latency increases also. The arbitration latency (CA) directly effect to the latency of transaction and the performance of system. CA is good parameter to evaluate the system performance by channel traffic.

However it is hard to estimate the CA because it is hard to estimate the contention conditions for the shared channel. We analyze the CA with CT and CS. We define CTS as the sum of CT and CS. The CTS can be calculated by components of system and operation conditions. And this value affects the channel utilization.

The timing diagram example of data transaction occurred between master 1 and slave 1 is shown in Figure 2. CR1 and CR2 are the request cycle times from master 1. This value means how often master request to use channel for data transaction. The less CR, the faster operation is required. CTS_{write} and CTS_{read} are transmission time including slave access time for write and read operation respectively. This means the time means that channel is in use.

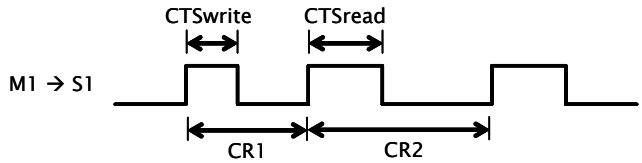

Figure 2. Timing diagram of request cycle time and transmission time at the channel for transaction data from master 1 to slave 1

CTS_{read}: channel transmission time including slave access time by read operation

CTS_{write}: channel transmission time including slave access time by write operation

CTS_{avg}: average channel transmission time

CR_{avg}: average request interval time

Channel utilization is the number of cycles that the channel was in use divided by the total cycles of running. We define the channel utilization ratio (CUR) of a master as average channel transmission time (CTS_{avg}) divided by average request interval time (CR_{avg}).

If a SoC has *n* masters, *m* slaves and *k* channels, CUR of channel *k* can calculate as below.

▪CUR between Master n and Slave *m* :

$$CUR_{nm} = CTS_{nm}/CR_{nm}$$

▪CTS of master *n* :

$$CTS_n = \sum_{j=1}^{m} CT_{nj} \text{ , where } j \text{ : slave ID}$$

▪CR of master *n* :

$$CR_n = \sum_{j=1}^{m} CR_{nj} \text{ , where } j \text{ : slave ID}$$

▪CUR of channel *k* :

$$CUR_k = \sum_{i=1}^{m} (CTS_i / CR_i) \text{ , where } i \text{ : master ID}$$

CUR of multi-master and multi-slave is the sum of CUR_i of all masters. Example, if SoC have 2 masters, 2 slaves and 1 channel, then we have 4 CURs, CUR_{11}, CUR_{12}, CUR_{21}, CUR_{22}. Therefore, total CUR is sum of them, $CUR = CUR_{11} + CUR_{12} + CUR_{21} + CUR_{22}$. The maximum of CUR is the number of channels. It means full channel utilization.

To evaluate the efficiency of CUR for estimating the system performance, we measure the arbitration latency, CA by simulation.

The throughput of a channel is proportional to CUR. The slope of throughput, α, is affected by channel transmission time, CT. If CT is low, total throughput (THP_{tot}) is high.

3. Simulation

3.1. Simulation environments

To evaluate the CA according to the CUR, we simulate the multi-master SoC with VisualElite from Summit Design and ModelSim from Mentor. Masters and channel are modeled by c++ code, and memory is modeled by VHDL code. The VisualElite provides co-simulation environment for c++ and VHDL.

The simulation conditions are as follows.

- arbitration algorithm:
  - round-ring arbiter (default : 1 clock cycle latency)
  - fair arbiter (default : 1 clock cycle latency)
- number of slave: 1 (memory, latency 2 or 5)
- number of channel: 1 (shared bus)
- channel width: 32bits
- request cycles distribution: Gaussian (standard deviation: 30% of mean)
- burst length: 4
- number of masters: 2 ~ 5
- probability of read operation: 0.1 ~ 0.7

3.2. Simulation results

First, we simulate SoC traffic performance according the number of masters. The results are shown in Figure 3.

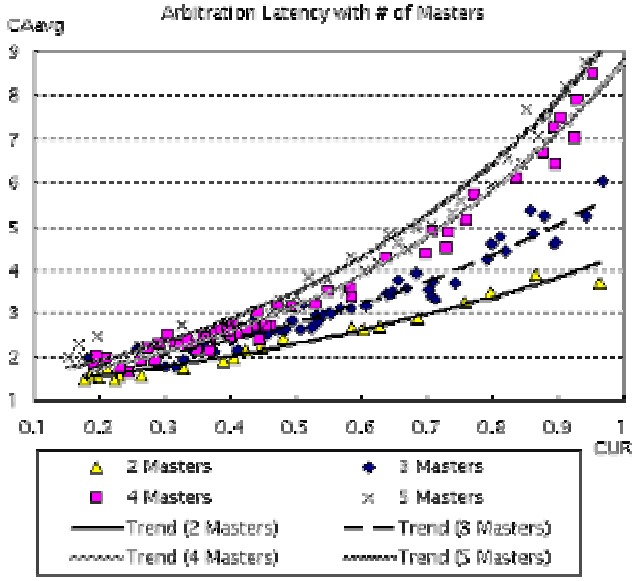

Figure 3. Average of arbitration latency, CA, at multi-master SoC in single channel architecture

The *y* axis is average arbitration latency. The *x* axis is sum of CUR at each master. The default arbitration latency is 1 clock cycle.

From Figure 3, we make the following observations.

- Arbitration latency increases exponentially with CUR.
- Arbitration latency increases in proportion to the number of master. This means that more the number of masters, more optimal mapping architectures are required.

Second, we simulate the arbitration latency with two arbitration algorithms: round-ring algorithm and fair algorithm. The round-ring algorithm is very popular in shared bus architecture because it is simple to implement, low latency, and uniform bandwidth at each master. The fair arbiter has the priority algorithm that all masters have equal ratio of number of grant per request [8]. The fair algorithm is hard to implement, in spite of merit for bandwidth guarantee without any adjusting of specific parameter.

The results for arbitration latency are shown in Figure 4 for 2, 3, 4 masters.

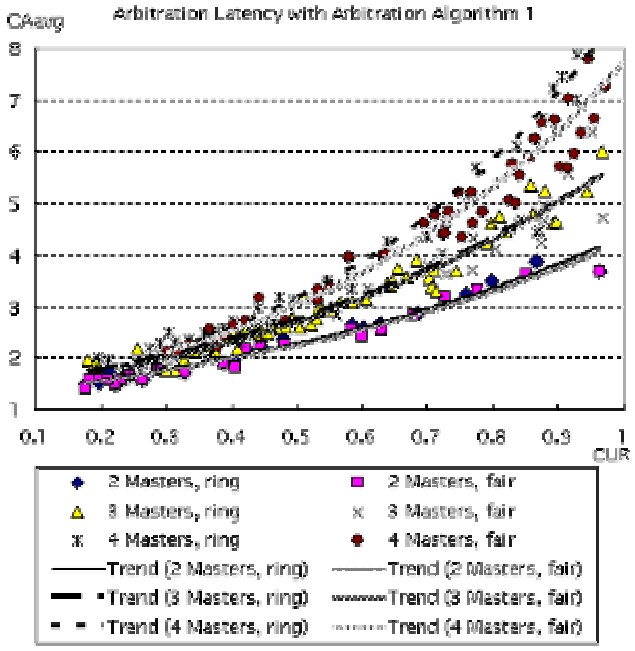

Figure 4. Average of arbitration latency, CA, at multi-master SoC with 2 types arbitration algorithm in single channel architecture

From Figure 4, we make the following observations.

- When the number of masters are 2 and 3, arbitration latency is not affected by the arbitration algorithm.
- When the number of masters are 4, arbitration latency is smaller at the fair arbiter than the round-ring arbiter.

We define the difference of request intervals among masters as $dCR = (CR_{max} - CR_{min}) / CR_{avg}$. Then original CA curve of master 4 is separated by curve1, where $dCR < 1$, (upper curve

in Figure 5) and curve2, where dCR > 1, as shown in Figure 5.

- dCR ≤ 1: asymptotic round-ring arbiter behavior
- dCR > 1: lower latency at CUR > 0.6 than round-ring arbiter
- When the request intervals have a large difference among masters, fair arbiter is better than round-ring arbiter. The arbitration latency by a round-ring arbiter is more increased at the master that has big request intervals. The fair arbitration algorithm gives a first priority to a master that uses the channel with a relatively less. Therefore the arbitration latency by a fair arbiter reduces at the master that has big request intervals, so total arbitration latency of system reduces.

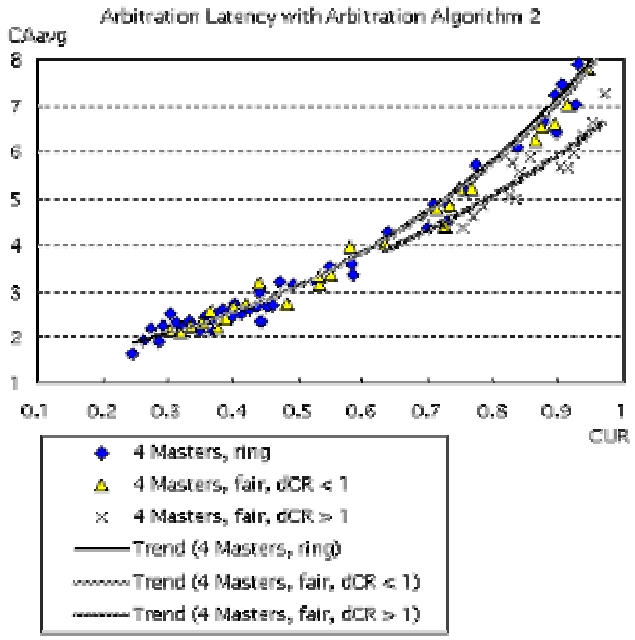

Figure 5. Average arbitration latency, CA, at 4 masters with 2 types arbitration algorithm in single channel architecture

The performance degradation is serious if the mapping of components to the channel is not optimized. For example, the channel utilization with 4 masters decreases 2.5 times each master than with 2 masters at the large CUR. Therefore if CUR is large value, then it is better to use more channels. However system has restricted resources, the optimal mapping of components to the channel is required toward decreasing of channel utilization ratio.

4. Conclusion

We defined the channel utilization ratio, and we proposed the analyzing methods for evaluating the arbitration latency for multi-master SoC on the shared channel architecture. By using proposed methods, we analyzed the performance of a channel from the viewpoint of the number of masters and arbitration algorithm.

Arbitration latency has a trend as below formula with number of masters and channel utilization ratio.

$$CA = c1 \times (N_m - 1) \times \exp(c2 \times CUR) + CA_{min}$$

where c1, c2 : constant

N_m : number of masters

CA_{min} : default arbitration latency

The analyzing method using channel utilization ratio is possible to look for the peculiar characteristics of components composed of SoC. And we can inquire the architecture of channel and parameterize the components for improving performance of SoC base on analyzed results. Also to maximize the performance of SoC in multi-channel architecture, the mapping algorithm for masters and slaves can be made by satisfying the requirements of CUR.

References

[1] A. Brinkmann, "On-Chip Interconnects for Next Generation System-on-Chips, " In *Proc. of the 15th Annual IEEE International ASIC/SOC Conference*, pp. 212-215, Sep. 2002

[2] Se-Joong Lee, "An 800MHz Star-Connected On-Chip Network for Application to System on a Chip," *ISSCC Dig. Of Tech. Papers*, pp. 468-469, 2003.

[3] Amer Baghdadi, "An Efficient Architecture Model for Systematic Design of Application-specific Multiprocessor SoC," *Proceedings of the Design Automation and Test in Europe Conference*, pp55-62, 2001

[4] Kyeong Keol Ryu, "Automated Bus Generation for multiprocessor SoC Design," *Proceedings of the Design Automation and Test in Europe Conference*, pp. 282-287, Mar. 2003

[5] Kanishka Lahiri, "Evaluation of the Traffic-Performance Characteristics of System-on-Chip Communication Architectures," in Proc. Intl. Conf. on VLSI Design, pp.21-35, Jan. 2001

[6] E. S. Shin, "Round-robin arbiter design and generation," *Proceedings of the International Symposium on System Synthesis* , pp. 243-248, Oct. 2002

[7] K. Lahiri, "Lottery: High performance communication architecture for system-on-chip designs," in Proc. DAC, pp 15-20, Jun. 2001

[8] Jisuhn Suh, "An Analysis and Implementation of High Fairness Arbitration Mechanism by Using Level-table and Static Priority Orders in Shared Bus Architecture," Proceedings of IP based SoC design 2003