Scalable distributed memory embedded system with a low-cost hardware message passing interface

Ha-young Jeong[a], **Won Hur, and Yong-surk Lee**
Processor Laboratory, School of Electrical and Electronic Engineering, Yonsei University,
262 Seongsanno, Seodaemun-gu, Seoul 120–749, Korea
a) *hyjeong@mpu.yonsei.ac.kr*

Abstract: In this paper, we propose a scalable distributed memory system with a low-cost hardware message-passing interface. The proposed interface improves the communication performance between nodes to decrease the overhead synchronization with a receiver reservation technique. The simulation results indicate that the performance is increased by 20% on 4x4 communications. The synthesis result of the proposed MPI indicates that the area was only 4.49% of each computing node. As a result, the proposed system is a useful embedded MPSoCs (Multiprocessor System on a Chip) for its low-cost implementation and scalability.

Keywords: distributed memory system, MPSoC, multiprocessor, message-passing interface
Classification: Science and engineering for electronics

References

[1] L. Benini and G. de Micheli, "Networks On Chips: A New SoC Paradigm," *IEEE Comput.*, 2002.
[2] F. Poletti, A. Poggiali, D. Bertozzi, L. Benini, P. Marchal, M. Loghi, and M. Poncino, "Energy-Efficient Multiprocessor Systems-on-Chip for Embedded Computing: Exploring Programming Models and Their Architectural Support," *IEEE Trans. Comput.*, 2007.
[3] F. Dumitrascu, I. Bacivarov, L. Pieralisi, M. Bonaciu, and A. A. Jerraya, "Flexible MPSoC platform with fast interconnect exploration for optimal system performance for a specific application," *Proc. Conf. Design, automation and test in Europe: Designers' forum*, pp. 166–171, 2006.
[4] S. Han, A. Baghdadi, M. Bonaciu, S. Chae, and A. A. Jerraya, "An efficient scalable and flexible data transfer architecture for multiprocessor SoC with massive distributed memory," *Proc. 41st annual Conf. on Design automation*, San Diego, CA, USA, June 7–11, 2004.
[5] AMBA AXI Specification, ARM Limited 2003.
[6] S. Mahadevan, F. Angiolini, M. Storgaard, R. G. Olsen, J. Sparso, and J. Madsen, "A Network Traffic Generator Model for Fast Network-on-Chip

© IEICE 2009
DOI: 10.1587/elex.6.837
Received March 16, 2009
Accepted May 20, 2009
Published June 25, 2009

Simulation," *Proc. Conf. Design, automation and test in Europe*, Munich, Germany, vol. 2, pp. 780–785, 2005.

1 Introduction

As of late, the MPSoCs (Multiprocessor Systems on a Chip) are the general solution to reduce power consumption of digital devices. With this trend, many commercial MPSoC have been designed, i.e. Cortex-A9 from ARM and OMAP from Texas Instruments Inc. However, these MPSoCs use additional hardware support to keep memory coherence because their design is based on using shared memory architecture. Consequently, this indicates a limitation on scalability. By using distributed memory architecture on MPSoCs, however, we can reduce the scalability problem on conventional shared memory MPSoCs [1].

The most recent research on distributed memory architecture systems is focused on using a hardware MPI unit to improve communication performance, which connects the network interface to communication hardware and manages the messages from tasks in a queue [2, 3, 4]. Once the queued messages are completely sent, the MPI unit automatically completes them, since MPI hardware research is now designed based on performing one transaction per operation. Yet when applying the conventional MPI hardware to a non-blocking interconnection, it has an optimization issue where a little effort to the synchronization mechanism and the scheduling algorithms between tasks improve the total bus performance. Thus, in this paper we propose a novel low-cost MPI hardware unit for distributed memory architecture system which can manage synchronization through the unit itself and reduce the burden of implementing a software MPI library.

2 Message passing interface hardware

In this paper, we propose an MPI unit that can support the general MPI functions and exchange messages from units for a distributed memory system. The proposed MPI unit can then synchronize, send, and receive messages between processor nodes. Since the MPI unit controls the message exchange operation between processor nodes, the MPI unit can support the non-blocking functions. As a result, the non-blocking operation can reduce the data transfer overhead of the processors. The preservation and control modules in the proposed MPI unit can reduce stall operations from programs, as well as status mismatch between processors while reducing unnecessary communication. Additionally, modules for non-blocking support and multiple outstanding data transaction support are implemented in the proposed MPI unit to optimize the data communication bandwidth on data dependent operation between processors.

© IEICE 2009
DOI: 10.1587/elex.6.837
Received March 16, 2009
Accepted May 20, 2009
Published June 25, 2009

2.1 Data transfer and synchronization mechanism

The proposed MPI unit uses send and receive functions to communicate between two processors. We describe the data transfer and synchronization process in steps:

Sender side:

1. The sender calls the "send()" function to initiate the transaction status in the message-box.
2. The sender reserves the local message buffer and once the necessary size of the buffer is guaranteed, it then writes data on the message buffer.
3. The sender waits for a data request signal from the receiver. Once the sender receives the data request signal from the receiver node, it sends a data reply grant signal to the receiver node. The receiver then reads the message buffer of the sender's message-box.
4. During data communication, the sender waits for the transaction terminate signal. If it receives the termination signal, it terminates the data transaction and deletes all involved entries.

Receiver side:

1. The receiver node calls the "receive()" function to guarantee enough message buffer. It initiates the transaction status in the message-box.
2. The receiver sends a data request signal and waits for a response. Once the data reply grant signal arrives from the sender node, it then prepares a data transmission. The transferred data is stored to the local memory.
3. After transmission, it then sends a transaction termination signal. Transactions are terminated and all involved entries are deleted.

On distributed memory architecture there are synchronization issue between receive and send signal due to an imperfect synchronization. So to compensate the imperfect synchronization, the MPI units should have additional buffers. When the receiver node calls the receive function despite the sender not transmitting any data, the sending processor node checks the reserve entry if it is possible to reserve the request signal. If there is enough space on the reserve entry, the request information is stored at the reserve entry. After the sending processor prepares the data for the transaction, it calls a send function to operate the message-box and notify the receiving processor node by the order on the reserve entry. As a result, the usage of a reserve entry, the node communication traffic and the communication delay time can be reduced. To support multiple outstanding data transactions, the MPI unit allocates a id tag per transaction on the AXI bus [5].

© IEICE 2009
DOI: 10.1587/elex.6.837
Received March 16, 2009
Accepted May 20, 2009
Published June 25, 2009

3 Simulation

To measure the data communication bandwidth of the proposed MPI unit, we have designed a BFM (Bus Functional Model) based on System C [6]. The BFM considers delay time of each block and can generate communication traffic for specific simulation environments. With the generated communication traffic, the performance of the proposed MPI unit was simulated. The buffer size constraint, which can affect the data communication performance, was set to a value of five words, where a word is 32-bits long.

3.1 One-to-many communication

An one-to-many communication is a communication where one processor node deals with various processor nodes parallel. It can be said as a MPI broadcasting of a collective communication, a gather communication, or a master-slave communication. We have simulated a message communication from one processor to many processor nodes with various message size from 4 bytes to 4096 bytes per message. Once the messages are fully sent, the total data transfer bandwidth and the amount of data transfer were measured.

In figure 1, the results indicates a performance increase as the number outstanding entries increase. This is because on one to many communication, the communication is processed as one master processor nodes sends data to large number of slave nodes.

3.2 Many-to-many communication

In this section, we have measured the bandwidth on many-to-many communications using the proposed MPI unit. The simulation was performed as each processor nodes sends a random but a significant length data to random processor nodes, while the average bandwidth was recorded for results. The simulation was done on various numbers of outstanding-entries.

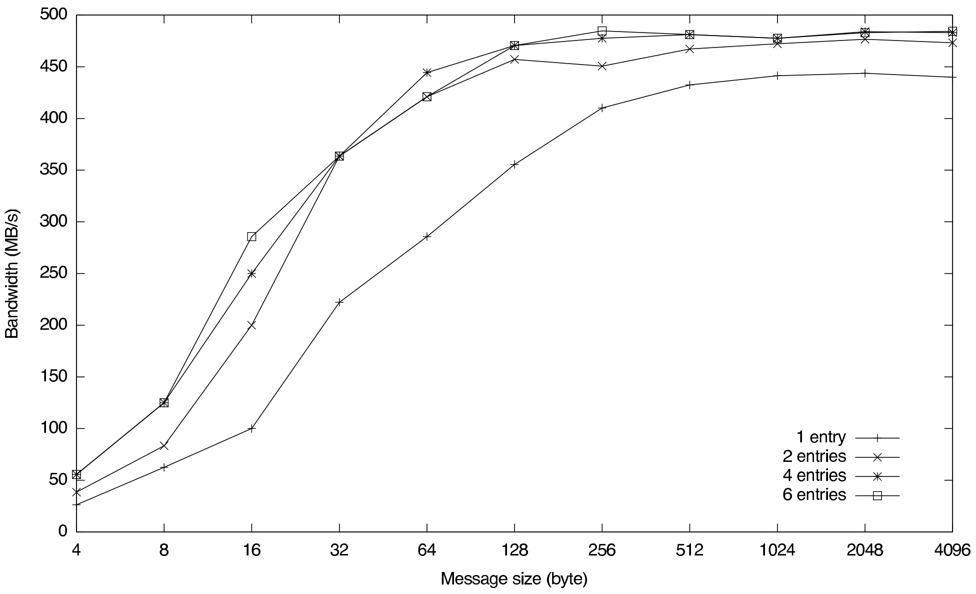

Fig. 1. Simulation results of 1 versus 4 communication

© IEICE 2009
DOI: 10.1587/elex.6.837
Received March 16, 2009
Accepted May 20, 2009
Published June 25, 2009

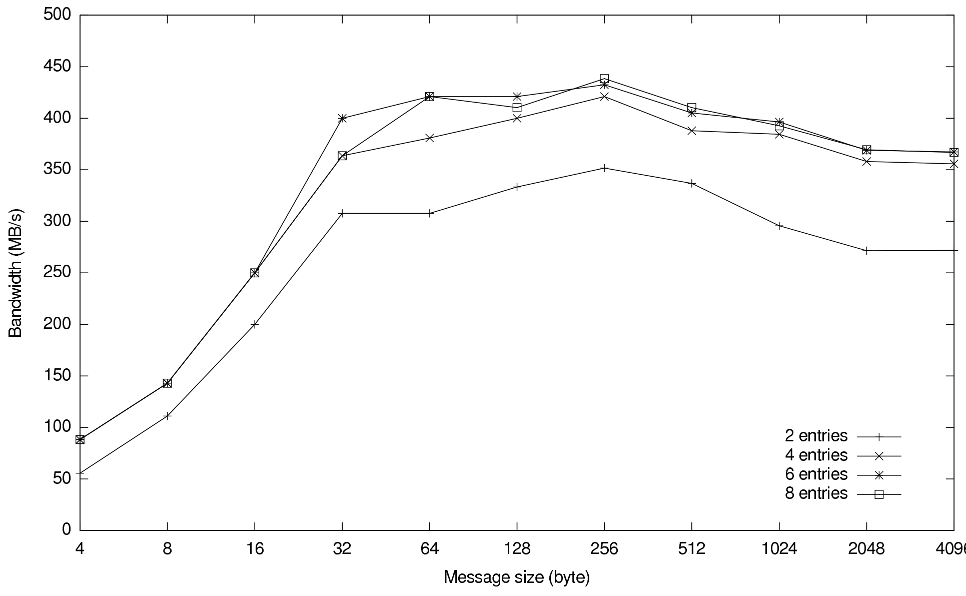

Fig. 2. Simulation results of 4 versus 4 communication

Figure 2 shows the simulation results of 4 versus 4 nodes communication. The results indicated that 20% to 25% of the performance has decreased when compared to 1 versus 1 node communication. This was obvious because, as the number of nodes communicating simultaneously increases, the dependencies between communication and the delay time of communication would increase. So as the number entries for outstanding increases, the results indicated a larger communication bandwidth which was from the reduced communication delay by reducing data dependency among processor nodes. And also the results on 4 versus 4 nodes communication indicated that by increasing the number of entries from 2 to 4, the performance increased by 24%. However, entries more than 4 did not show any significant performance increase.

On 8 versus 8 node communication, the simulation results related to entry numbers indicated a performance increase by increasing the number of entry until 8 and the performance started to decrease as the numbers go up.

Figure 3 shows the simulation results on communication through 4 versus 4 nodes, 8 versus 8 nodes, and 12 versus 12 nodes. As shown in the figure, the bandwidth increases as the number of outstanding entries increase, but the bandwidth did not increase further when the entry number exceeded the number of the nodes. It is obvious that the performance should decrease as the number of communicating nodes would increase, but by controlling the number of outstanding entries it can increase the performance proportional to number of communicating nodes.

4 Proposed MPI unit hardware implementation and verification

As to verify the proposed MPI unit, we have designed an embedded multiprocessor system based on a distributed memory architecture. The each

© IEICE 2009
DOI: 10.1587/elex.6.837
Received March 16, 2009
Accepted May 20, 2009
Published June 25, 2009

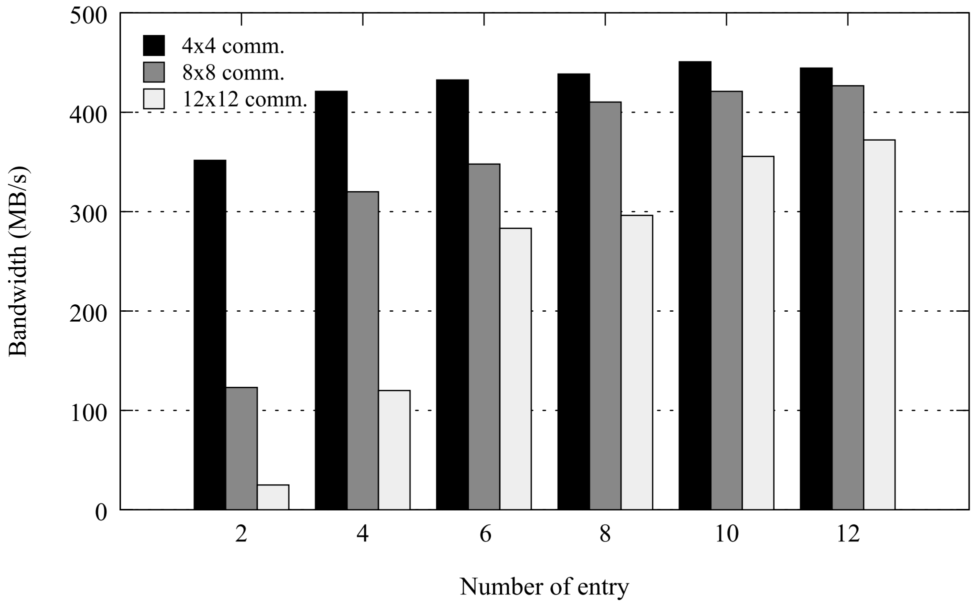

Fig. 3. Simulation results on communication through 4 versus 4 nodes, 8 versus 8 nodes, and 12 versus 12 nodes

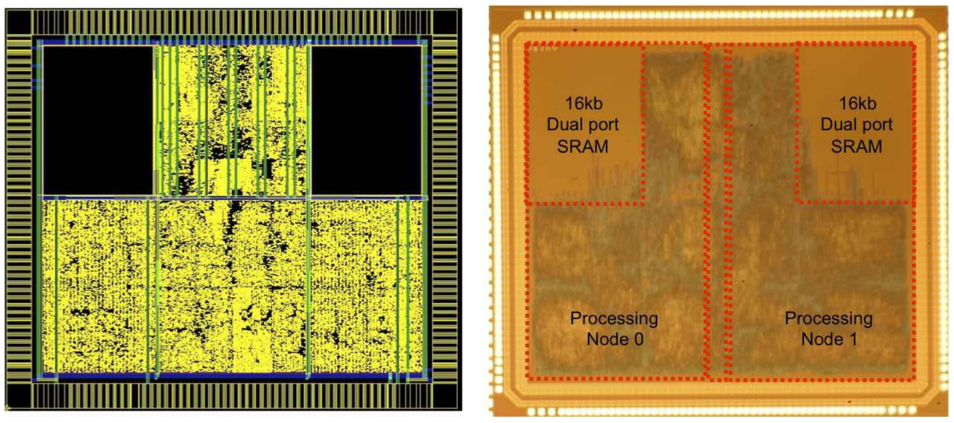

Fig. 4. Chip layout : two processing nodes with the proposed MPI unit

processors on processor nodes were designed based on a RISC MIPS DLX architecture. Each nodes were consisted with one RISC core and a message box containing 16 kb private memory.

From figure 4, it indicated that the area for the memory took the most part of the area. The processor core, as well as the MPI unit, did not take a large portion of the total area. A 30% of area increase of the MPI unit itself is negligible when considering a total area increase of the total chip area is under 1%. Thus, with a negligible area increase effort, the total system performance can increase by the proposed MPI unit.

5 Conclusion

In a recent developed embedded system, MPSoC's are used and the number of processor nodes are increasing. As the number of processor nodes in a MP-

© IEICE 2009
DOI: 10.1587/elex.6.837
Received March 16, 2009
Accepted May 20, 2009
Published June 25, 2009

SoC increases, the communication overhead is a considerable bottleneck. To reduce the bottleneck due to communication overheads from multiprocessor environment, the use of distributed memory architecture over shared memory architecture is a solution because the communication overhead due to increasing the number processor nodes is lower. Consequently, in this paper we propose an MPI unit as to optimize the MPI performance in distributed memory architecture. And the proposed MPI unit was designed to lessen the burden of generating the MPI library. The proposed MPI unit uses the non-blocking bus system and a mechanism by using more entry module to enhance the message-passing performance between nodes. By using the proposed MPI unit, an MPSoC system using distributed memory system can be designed even more efficiently.

© IEICE 2009
DOI: 10.1587/elex.6.837
Received March 16, 2009
Accepted May 20, 2009
Published June 25, 2009