On-Chip Communication Buffer Architecture Optimization Considering Bus Width

Salita Sombatsiri, Keishi Sakanushi, Yoshinori Takeuchi and Masaharu Imai
Graduate School of Information Science and Technology
Osaka University, Osaka, Japan 565-0871
Email: s-salita, sakanusi, takeuchi, imai@ist.osaka-u.ac.jp

***Abstract*—This paper studies the on-chip communication buffer optimization method for design space exploration, considering bus width. For embedded multicore system-on-a-chip (MCSoC), there usually are many buses on the system to handle a vast amount of data communications between several processing cores. Therefore, buffer architecture optimization has become one of the most important topics in this area as a parameter for communication architecture.**

This paper proposes an SRAM optimization method to construct buffer architecture candidates through architecture exploration. Moreover, the design quality of each system architecture candidate is evaluated. The experiment of the proposed method is applied to a JPEG encoder system. The result shows that the proposed exploration method can explore a variety of buffer architecture with trade-off between transfer time and area. Moreover, the result shows that buffer architecture optimization through exploration with pruning can reduce the computation time by approximately 94% .

***Index Terms*—Interconnection Architectures, Optimization, Multiprocessor Systems**

I. Introduction

The advancement in process technology during the past decades enables the ability to implement multicore system-on-a-chip (MCSoC). While MCSoC designs achieve high performance requirement, a large amount of time and human resources are required to obtain the optimal design. To solve the mentioned problem, IP-based design methodology [1] is proposed. In the IP-based design methodology, SoCs are designed by reusing previously designed modules, usually called intellectual property (IP), and standard bus architecture. Generally, IP-based design methodology iterates the generation and evaluation of architecture candidates by simulation, which is a time consuming task.

On MCSoC, not only data processing on Intellectual Property (IP), but also data communication on buses between processing cores drive the system to operate productively. Therefore, communication architecture should be capable of transferring data between IPs effectively as the number of data processing cores grows. Communication architecture and its parameters must be chosen carefully, so that the system achieves high performance.

One of the most important parameters on communication is buffer architecture. Communication buffers are used for temporarily storing data transferred between IPs, and compensating the operation frequency difference between IPs and buses. On-chip communication buffer also exists in bus bridge and DMA Controller. In several IP designs, buffers are usually implemented with SRAM because they are fast, compact and do not need refreshing circuitry. For instance, in Altera's inverse discrete wavelet transform IP [2], line and tile buffers are implemented using SRAMs.

Generally, buffer size should be determined by amount of data transfer in each channel. Nevertheless, bits per word, the number of words, and the number of SRAMs comprising buffer architecture also affect system's design quality in implementation. Therefore, the buffer architecture suitable for each design must be chosen carefully according to the amount of data and data width of each data transfer.

This paper studies implementations of buffer architecture. The contributions of this paper are (1) optimizing parameters of SRAM comprising buffer architecture corresponding to each communication channel for implementation and (2) proposing an exploration method to offer a concrete implementation of communication buffer. This paper is not only an extension of architecture level design quality estimation method based on data-flow analysis [7,8], but also offers several buffer architecture candidates with performance-area trade-off for ease of implementation and broadens design space exploration by exploring parameter sets of communication buffer. Furthermore, execution time, area, and power consumption are also evaluated as design quality.

The rest of this paper is organized as follows. Section II describes related works. Section III describes the conventional architecture exploration method based on system-level profiling. Section IV describes the proposed buffer architecture optimization method. Section V describes the evaluation experiment and result. Finally, section VI describes the conclusion and future work.

II. Related Work

There are several remarkable studies regarding communication architecture optimization [3,4]. The design exploration method for optimizing on-chip communication architecture [3] optimizes system-level on-chip communication into communication architecture template and efficiently explores communication architecture by employing system-level approach to speed up the method. In fast exploration of bus-based communication architectures [4], proposed by Parischa et al., models a system in cycle count accurate at transaction boundaries abstraction level, and explores bus-based communication

978-0-7695-4800-5/12 $26.00 © 2012 IEEE
DOI 10.1109/MCSoC.2012.36

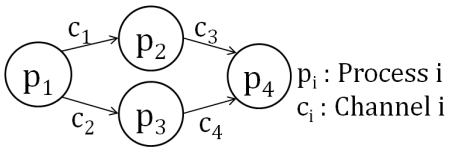

Fig. 1. Example of System-Level Model

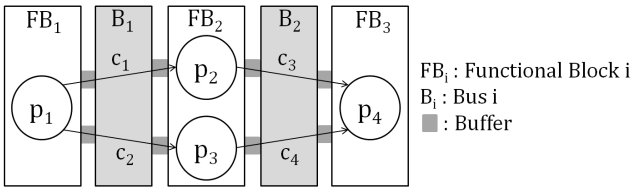

Fig. 2. Example of Architecture-Level Model

architecture and standard bus specification in system-level. However, their studies focus on architecture optimization and optimize buffer architecture only of its capacity.

Buffer optimization methods for network-on-chip (NoC) have been proposed as NoC has become an effective communication architecture on SoC. Jafari et al. [5] proposed a NoC's buffer optimization techniques through flow regulation, which analyzes communication flow on the chip and minimizes buffer size. Nevertheless, these methods focus on data flow in communication architecture analysis and optimizes buffer for only its capacity.

The work of Saastamoinen et al. has come closest to the purpose of this paper. Buffer size for Proteo network-on-chip (NoC) [6] is optimized considering traffic in the interconnect. The properties of buffer are defined by internal word width and the maximum size packets that the buffer can store. However, buffers in their research are implemented using register banks and performance evaluation is done by simulation.

This paper optimizes buffer architecture on communication architecture for not only its size, but also the physical dimensions of SRAMs for implementation, which are bits per word of SRAM, number of words of SRAM, and number of SRAMs comprising a buffer architecture. Performance evaluation is conducted by analyzing data flow and execution order, which is faster than simulation every architecture candidate.

III. Architecture Exploration Method Based On System-Level Profiling

This section describes the conventional architecture exploration method based on system-level profiling and architecture-level performance estimation based on data flow analysis proposed in [7,8].

A. Target Application System Model

1) System-Level Model (SLM): SLM is an architecture-independent model representing a target application system with process and channel corresponding to data processing and data transfer respectively. Figure 1 shows an example of the SLM, which composes of four processes and four channels. P_i represents process i and C_j represents channel j as well as the direction of the data transfer.

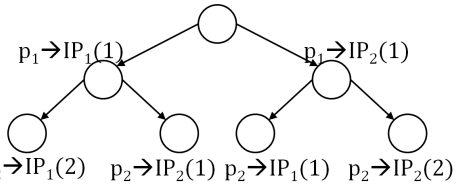

Fig. 3. Process Mapping Search Tree

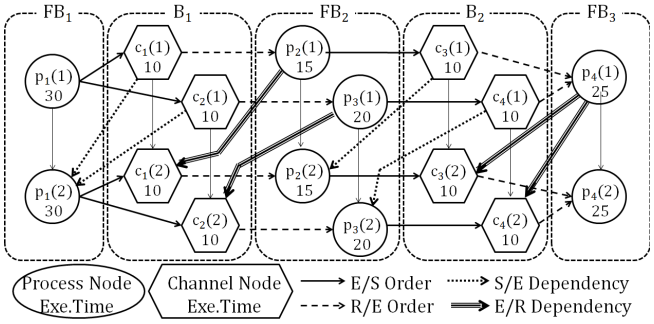

Fig. 4. Example of AL-EDG

2) Architecture-Level Model (ALM): ALM is a model representing an architecture of the system, consisting of functional blocks, which are registered in IP database, and buses. Each ALM contains information of an architecture, which are process to functional blocks mapping, channel to bus mapping, bus width, execution frequency, number of buffers, and its estimated design quality.

Figure 2 shows an example of the ALM and mapping of the SLM. Each process in SLM is mapped to a functional block from the IP database and each channel is mapped to a bus. A buffer exists in every channel between functional block and bus. FB_i represents functional block i and B_j represents bus j.

B. Architecture Exploration

The design space is explored by parameter set search tree traversal. Nodes of the parameter set search tree correspond to process mapping, channel mapping, execution frequency of functional blocks, execution frequency of buses, bus width and number of buffers. A path from the root to a leaf corresponds to one ALM. The search tree is traced in order.

First, each process is mapped to a functional block of IP registered in the IP database. Figure 3 shows an example of the process mapping search tree. $P_i \rightarrow IP_j(k)$ denotes process i is mapped to instance k of IP j. Then, each channel is mapped to a bus. Execution frequency of functional block is mapped to frequency candidates of the corresponding IP. Execution frequency of bus can be selected from bus frequency candidates inputted by designers. Bus width of each bus can be selected from bus width candidates inputted by designers. Lastly, the number of input and output buffers of each channel can be selected according to the maximum number of buffer constraint.

C. Architecture-Level Performance Estimation

Performance of the application is estimated by analyzing execution order of processes and the amount of transferred data between processes, obtained by system-level profiling using SystemC [9] to describe behavior of processes and data communication as SLM.

First, the execution order is used to construct system-level execution order graph (SL-EOG). Two types of execution order exist between data processing and data transfer. R/E order represents the condition where the execution of data processing starts after all data are received. E/S order represents the condition where data transmission starts after data processing is completed.

Architecture-level execution dependency graph (AL-EDG) is constructed from SL-EOG by adding two dependencies according to ALM. E/R dependency means that the next data can be received after the data in buffer are executed. S/E dependency means that the next data processing can be executed after the data in buffer are sent to other process.

Figure 4 shows an example of AL-EDG. Each vertex p_i represents either data processing or data transfer, and each edge c_i represents execution order in SLM and dependency between data processing and data transfer according to ALM. Execution time of data processing is execution cycle information from IP database. Execution time of data transfer can be obtained from bus execution frequency, bus bit width and the amount of data transferred.

Execution time is estimated by analyzing AL-EDG as in [7]. The process node that has no dependency is executable and the node that has the highest priority among executable processes on the same IP is executed. Likewise, the channel node that has no dependency is executable and the node that has the highest priority among executable processes on the same bus is executed. The executable nodes are searched until all nodes in AL-EDG is executed.

D. Buffer Architecture Realization

Ueda et al. [7,8] realized buffers as logical memory storage, where IP and bus can access buffers from their own perspective. For example, IP accesses a 512-byte buffer as 8-bit data storage, while 16-bit bus accesses the same buffer as 16-bit data storage. The conventional method considered buffers only for their size and lacked of consideration about SRAM's physical access behavior and implementation. Therefore, the conventional method can explore only one buffer architecture.

Communication buffer architecture optimization allows buffer architectures to be explored in parameter detail based on bus width and data size. Moreover, the optimization offers several potential combinations of buffer architecture for each architecture under performance-area trade-off.

IV. On-Chip Communication Buffer Architecture Optimization

This section discusses the effect of buffer architecture towards on-chip communication quality, specifies optimized target buffer architecture and describes buffer architecture exploration method.

A. SRAM towards Design Quality

In communication buffer implementation, designer must choose an appropriate SRAM that is suitable to characteristics of each data communication on the system. The characteristics include bus width and data width of each communication channel. At the same time, the SRAM should be economical in area and energy consumption. Among several buffer architectures for on-chip communication buffer, each architecture holds trade-off between execution time and area.

Assuming a communication channel of 8-bit data between two IPs on 16-bit bus. In one cycle, 16-bit bus can transfer two pieces of data. An 8-bit SRAM can store eight bits in one clock cycle, which means it requires two clock cycles to store two pieces of 8-bit data on 16-bit bus. In this case, the bus utilization is equal to that of transferring one piece of 8-bit data on 16-bit bus in each cycle and the bus cannot be efficiently utilized. There are two more approaches to implement the buffer. With the implementations using a 16-bit SRAM and using two 8-bit SRAMs of the equivalence capacity, the data transfer time can be a half because all bits of the bus are utilized and buffer can store 16 bits per clock cycle. However, the sizes of SRAM buffer of the later approaches are bigger than the size of one 8-bit SRAM buffer.

Communication throughput is a good measurement of bus utilization. In this paper, communication throughput is regarded as in Eq. (1).

$$CommunicationThroughput = \frac{Amount\ of\ Data}{Transfer\ Time} \quad (1)$$

This time, assume that a 64 pieces of 8-bit data transfer on 16-bit bus operating at 50 MHz with 3 aforementioned buffer architectures. Communication throughput of a one 8-bit SRAM buffer architecture is 400 Mbit/s, while communication throughput of a one 16-bit SRAM buffer architecture and a two 8-bit SRAM buffer architecture is 800 Mbit/s, which is twice as much as the first buffer architecture.

Moreover, number of row and column of SRAM also varies according to size of column multiplexer, which affects SRAM's physical width, height, speed and power consumption [10].

B. System Assumption

In this paper, all buffers are optimized using single port SRAM. In one communication channel, there are at least one receive buffer and one transmit buffer of the same architecture. The optimization is based on shared bus model, and the data transfer can utilize full bus width.

C. Target Buffer Architecture

Buffer architecture is optimized into two categories based on bus width and data size of each channel.

TABLE I
PARAMETER LIST OF BUFFER ARCHITECTURE

Parameter	Description
Number of bits per word	Number of bits per one word of SRAM associates with either bus width or data width.
Number of words per one SRAM	Number of words of each SRAM in buffer architecture necessary for storing data
Number of component SRAM	Number of SRAM comprising one buffer architecture
Size of multiplexer	Size of column multiplexer within the organization of SRAM determines number of physical row and column of SRAM

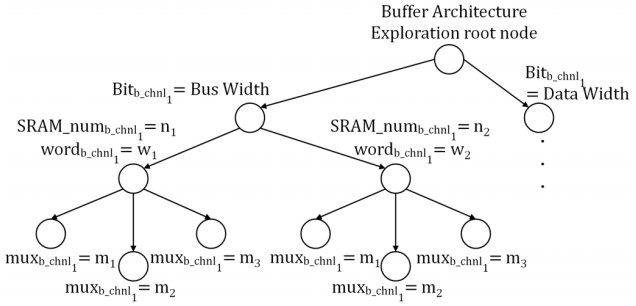

Fig. 5. Buffer Parameter Set Search Tree

1) Number of bits of SRAM associates with bus width: Buffer of this architecture can store all data bits on the bus within one clock cycle in one word. If bus is wider than data width, several pieces of data can be stored in the same word. In case of data width greater than bus width, it is assumed that data is transferred in two or more separate clock cycles. In this case, more than one SRAM would allow the connecting IP to read one piece of data within one clock cycle from two or more separate SRAMs.

2) Number of bits of SRAM associates with data size: Buffer of this architecture stores one data in one word, allowing at most one data can be stored in one clock cycle. In case that bus can transfer more than one data in one cycle, more than one SRAM would allow the system to benefit wide bus width.

D. On-Chip Communication Buffer Architecture Exploration

In this paper, buffer architecture is decided according to SRAM's necessary parameters as described in Table I. Note that number of bits per word and number of words per one SRAM are mapped to the smallest available SRAM value which can store data of the exploring channel. The buffer parameter search tree is shown in Fig. 5. Bit_{Ci} denotes number of bits per word of SRAM comprising buffer of channel i. $SRAM\_num_{Cj}$ denotes number of bits SRAM comprising buffer of channel j. $word_{Ck}$ denotes number of words per one SRAM of SRAM comprising buffer of channel k. mux_{Cl} denotes size of multiplexer of SRAM comprising buffer of channel l. The parameter of buffer architecture is searched and decided in order as follows.

1) Number of bits per word: Candidate for number of bits per word associates to either bus width or data width.

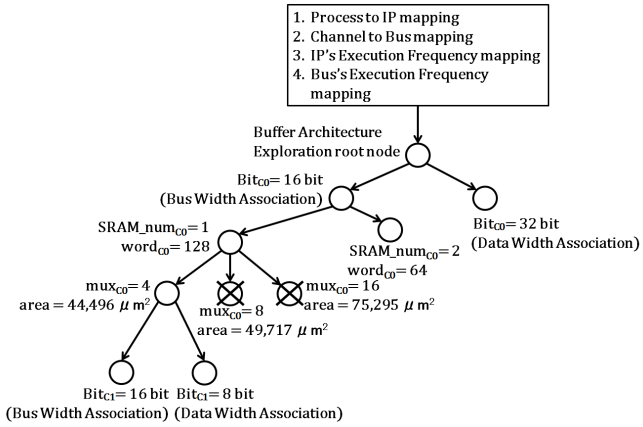

Fig. 6. Example of Buffer Parameter Set Search Tree

After number of bits per word association is decided for each buffer, Step 2 and 3 is the same for both association.

2) Number of component SRAM comprising a buffer: Maximum number of component SRAM candidates is shown in Eq. (2), where $\lceil\ \rceil$ denotes ceiling function. Then, number of words per one SRAM can be calculated according to number of bits per word association and number of component SRAM as in Eq. (3).

$$Max(SRAM\_num) = \begin{cases} \left\lceil \frac{DataWidth}{BusWidth} \right\rceil, DataWidth > BusWidth \\ \left\lceil \frac{BusWidth}{DataWidth} \right\rceil, DataWidth \leq BusWidth \end{cases} \quad (2)$$

$$Word\_num = \frac{RequiredWord}{SRAM\_num} \quad (3)$$

where $RequireWord$ equals to maximum number of data transferred in channel if number of bits per word associates with data width. If number of bits per word associates with bus width, $RequireWord$ equals to

$$\frac{\text{Data Width} \times \text{maximum number of data}}{\text{Bus Width}}$$

3) Size of multiplexer: In this paper, size of multiplexer is chosen towards size of multiplexer that yields the smallest SRAM to continue further exploration.

The design space exploration method is extended for buffer architecture exploration by appending additional buffer architecture parameter set search tree to the original search tree described in section III.C.

Exploring for architecture candidates could be time consuming while traveling along the parameter set search trees, especially when the traversal no longer produces optimal architecture. In order to shorten exploration time, parameter set search tree is pruned by considering execution time and

TABLE II
DATA TRANSFER TIME CALCULATION FORMULA

Relation of Bus Width and Data Width	SRAM's number of bits per word association	Transfer Time	
		between IP and send buffer/ between IP and receive buffer	between send buffer and receive buffer
Data Width $\leq$ Bus Width	Bus Width	Number of Data $\times \frac{1}{f_{IP}}$	$\left\lceil \frac{\text{Number Of Data} \times \text{Data Width}}{\text{SRAM Num} \times \text{Bus Width} \times f_B} \right\rceil$
	Data Width	Number of Data $\times \frac{1}{f_{IP}}$	Number of Data $\times \frac{1}{f_B}$
Data Width $>$ Bus Width	Bus Width	$\left\lceil \frac{\text{Number Of Data} \times \text{Data Width}}{\text{SRAM Num} \times \text{Bus Width} \times f_{IP}} \right\rceil$	$\left\lceil \frac{\text{Number Of Data} \times \text{Data Width}}{\text{SRAM Num} \times \text{Bus Width} \times f_B} \right\rceil$
	Data Width	Number of Data $\times \frac{1}{f_{IP}}$	$\left\lceil \frac{\text{Number Of Data} \times \text{Data Width}}{\text{SRAM Num} \times \text{Bus Width} \times f_B} \right\rceil$

hardware area. In case one or both of the conditions below is met, parameter set search tree pruning process occurs.

1) If one or both lower bound of execution time and hardware area of the current search node exceeds the design constraints, all descendants are pruned.
2) If both lower bound of execution time and hardware area of current search node exceeds the explored optimal architecture, all descendants are pruned.

This research prunes parameter set search tree when every parameters comprising an architecture of each buffer are mapped.

In the following, an example of buffer architecture exploration and search tree pruning for channel $C0$ and $C1$ of JPEG encoder system in section IV shown in Fig. 6 is described. Assuming that the architecture in this example contains one bus and bus width is 16 bits. SRAM in this example based on SRAM synthesized using 0.18 μm process technology. In the example, the node after process to IP mapping, channel to bus mapping, IP's execution frequency mapping and bus's execution frequency mapping becomes the root node for buffer architecture exploration.

From the root node's architecture, parameters of buffer architecture for each channel are explored and the search tree traversal is done in the depth first search manner. Firstly, number of bits per word (Bit_{C0}) is decided to associate with bus width. In this step, $Bit_{C0} = 16$. Secondly, number of component SRAM is decided to 1. $word_{C0}$ can be calculated using Eq. (3). Number of word required for the SRAM is 96 words. However, SRAM's available number of words is in the value of the power of 2. Therefore, $word_{C0} = 128$. Then, size of multiplexer is selected towards the smallest SRAM. In this example, $mux_{C0} = 4$ yields the smallest SRAM among available size of multiplexer. Therefore, descendants node upon $mux_{C0} = 8$ and $mux_{C0} = 4$ node are pruned. This time, the lower bound of the architecture is estimated. In case of one or both of the pruning condition is met, the descendants of the current node is also pruned. From $mux_{C0} = 4$ node, exploration for parameters of C1's buffer architecture continues.

TABLE III
JPEG ENCODER'S CHANNEL SPECIFICATION

Channel	Source	Destination	Data Size	Maximum Data
C0	block_ split	CT	24	64
C1	CT	DCT	8	64
C2	DCT	Q	12	64
C3	Q	Zigzag	12	64
C4	Zigzag	HUFF	12	64
C5	HUFF	writer	8	256

Finally, after the buffer architecture for every channel is explored, number of buffer in each channel is then explored.

E. Performance Estimation

Performance of the application system is estimated using AL-EDG analysis as explained in Section III.C. However, the transfer time of each channel node is calculated based on buffer architecture as shown in Table II, where f_{IP} denotes execution frequency of IP and f_B denotes execution frequency of Bus. Transfer time is the time spent for data communication between two IPs. In this paper, transfer time comprises of transfer time between IP and send buffer, transfer time between send buffer and receive buffer of each channel, and transfer time between receive buffer and IP.

F. Area Estimation

In this paper, area of each buffer is regarded as the product of area of SRAM and number of SRAM. Area of SRAM is obtained through logic synthesis using 0.18 μm process technology.

G. Energy Consumption Estimation

This paper estimates buffer's dynamic energy consumption when it is activated for reading and writing data in data communication [11]. Dynamic energy consumption of receive buffer in one transfer is estimated by the product of SRAM's write operation energy consumption and word access counts. Dynamic energy consumption of transmit buffer in one transfer

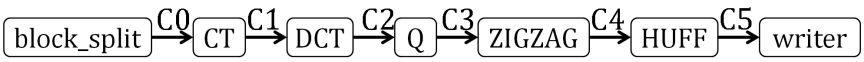

Fig. 7. System-Level Model of JPEG Encoder Application System

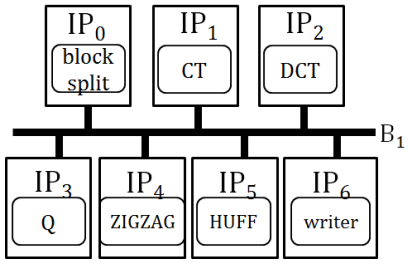

Fig. 8. Selected Architecture and Buffer Architecture Example a

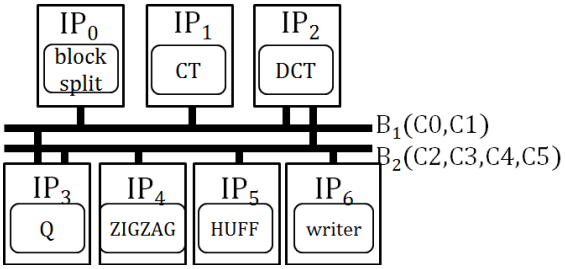

Fig. 9. Selected Architecture and Buffer Architecture Example b

is estimated by the product of SRAM's read operation energy consumption and word access counts. Word access count equals to transfer cycle if SRAM's number of bits is associated to bus width or else, it equals to number of transferred data if SRAM's number of bits is associated to data size. Write and read operation energy consumption are obtained using Synopsys's Power Compiler [12].

V. Experiment and Result

This section describes the experiment and its result. The experiment was conducted on 2.80 GHz Intel Core i7 CPU, 8 GB memory and 64 bit Fedora 14 operating system. The proposed method was applied to JPEG encoder application system, which consists of seven processes and six communication channels as shown in Fig. 7 to compress 512×512 pixels image. The processes are for block splitting (block_split), color transformation (CT), discrete cosine transformation (DCT), quantization (Q), zigzag ordering (ZIGZAG), huffman encoding (HUFF) and file writing (writer) respectively.

The specification for each channel is described in Table III.

First, the global exploration is conducted with the following parameters.

- Maximum number of bus : 2
- Bus width candidate : 16, 32 bit
- Bus frequency candidate : 50 MHz

Three most potential bus architectures shown in Fig. 8, 9, and 10 are selected based on the number of optimal solutions

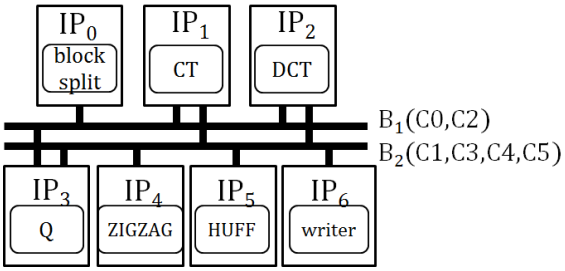

Fig. 10. Selected Architecture and Buffer Architecture Example c

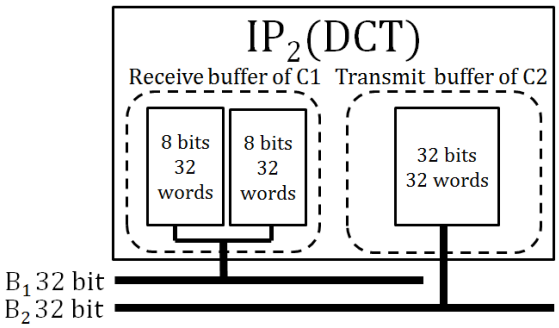

Fig. 11. Example of Explored Buffer Architecture on DCT IP

and performance to explore for parameters of buffer architectures. Architecture in Fig. 8 contains one bus and bus bit width is fixed to 16 bits and 32 bits in buffer architecture parameter explorations. Architecture in Fig. 9 shows an architecture with two buses, where channel $C0$ and $C1$ are mapped to bus B_1 and channel $C2$, $C3$, $C4$ and $C5$ are mapped to bus B_2. Both bit width of bus B_1 and B_2 are fixed to the same width of 16 bits and 32 bits in buffer architecture parameter exploration. Architecture in Fig. 10 shows an architecture with two buses, where channel $C0$ and $C2$ are mapped to bus B_1 and channel $C1$, $C3$, $C4$ and $C5$ are mapped to bus B_2. In buffer parameter exploration, bus B_1 is fixed to 16 bits and bus B_2 is fixed to 32 bits.

Figure 11 shows an example of an explored buffer architecture for DCT IP which is connected with two different 32-bit buses. In this architecture, buffer of $C1$ is selected as two 8-bit SRAMs, associates with data size, and buffer of $C2$ is selected as one 32-bit SRAM, associates with bus width.

Figure 12 shows the trade-off between transfer time estimated by formula in Table. II and communication architecture area, which includes bus and buffer area. Points on the graph represent the relationship between transfer time and communication architecture area of buffer architecture combinations of architectures in Fig. 8, 9 with 16-bit bus, 32-bit bus and Fig. 10 with one 16-bit bus and one 32-bit bus. Each line in the graph draws trade-off boundary of each architecture explored

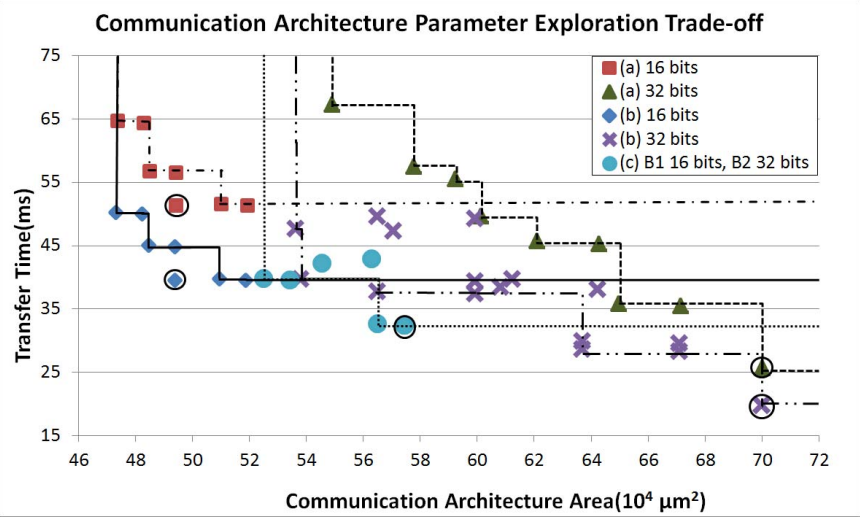

Fig. 12. Communication Architecture Parameter Exploration Transfer Time-Area Trade-off

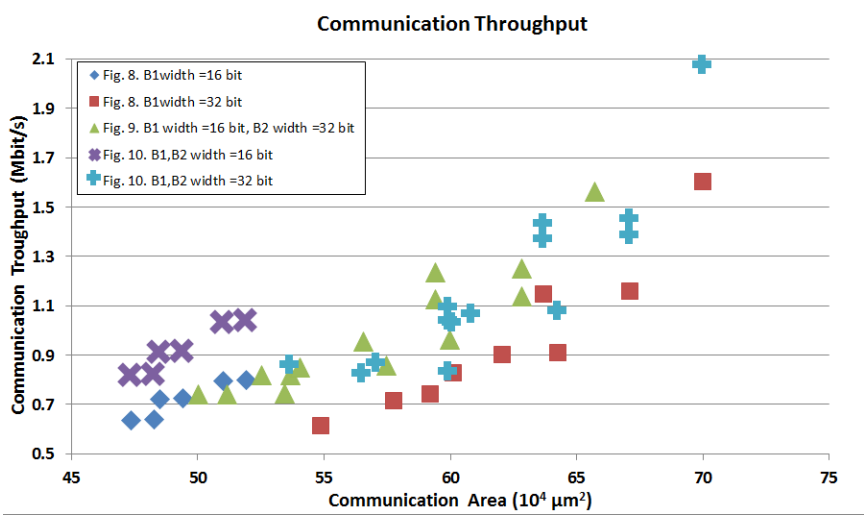

Fig. 13. Communication Throughput

by the proposed method. The architecture also explored by the conventional method corresponding to each bus architecture is surrounded by a circle. However, the architectures explored by conventional method for bus architecture in Fig. 8 B_1 width equals 16 bits, Fig. 9 and Fig. 10 B_1 and B_2 width equal to 16 bits are not used to draw trade-off boundary line because transfer time estimation in conventional method has inaccuracy regarding buffer architecture. Consequently, such architectures does not exist when considering buffer architecture for transfer time estimation in the proposed method.

To illustrate the effect of buffer architecture towards bus utilization, Fig. 13 shows the communication throughput corresponding to each architecture explored in Fig. 12. Various buffer architectures also have effect in determining the capacity of data transfer as shown in Fig. 13. Various buffer architecture can results in different communication throughput, despite of the fact that IP and bus architecture is the same.

Table IV shows the computation time and number of traversed node of the exploration with pruning and without pruning. The result shows that exploration with pruning can reduced computation time by 92-96% .

The proposed method can explore parameters of buffer architecture, which are SRAM's number of bit per word, number of word per one SRAM, number of component SRAM comprising a buffer architecture and size of SRAM's column multiplexer, constructing several buffer architecture combinations for one architecture, while conventional method can explore only one architecture. According to the designers' desired bus width, the designer can choose the buffer architecture based on their design constraint. For a strict time

TABLE IV
COMPUTATION TIME AND NUMBER OF EXPLORED NODE

Exploration		expl. with no pruning			expl. with pruning			Reduced
		No. Node	No. Est.	Comp. time (min)	No. Node	No. Est.	Comp. time (min)	Comp. time
Global expl.		56,360	20,517	114	3,514	636	6	96%
Parameter expl.	Fig.8 with 16-bit bus	8,394	2,924	25	712	141	2	92%
	Fig.8 with 32-bit bus	47,979	17,601	150	2,815	501	5	96%
	Fig.9 with 16-bit and 32-bit bus	28,876	10,582	91	3,720	684	6	92%
	Fig.10 with 16-bit buses	8,401	2,925	27	719	140	2	92%
	Fig.10 with 32-bit buses	47,986	17,602	150	3,872	683	6	96%

* $No.Node$ denotes number of node traversed during exploration, $Comp.time$ denotes computation time

constraint design, a larger buffer architecture can be chosen in trade-off with high performance. On the other hand, a smaller buffer architecture can be selected if area constraint is strict.

VI. CONCLUSION AND FUTURE WORKS

This paper is not only an extension of the proposed method in [7], but also offers buffer architecture candidates with performance-area trade-off and optimizes physical architecture of SRAM to be able to explore parameters of buffer architecture. The experimental result has verified that the optimization method can explore parameters for buffer architecture constructing smaller buffers, comparing to the conventional method, with execution time trade-off. However, this paper has not yet considered bus protocol in the communication buffer architecture optimization. Therefore, communication buffer architecture optimization considering bus protocol remains as future work. Moreover, since multi-layer bus increases performance of systems because it allows concurrent communications of 2 or more data transfer, bus matrix optimization for multi-layer standard bus protocol also remains as communication optimization future work.

ACKNOWLEDGMENT

This research was partly supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B), 2030017, 2011.

REFERENCES

[1] D. Gajski, "IP-Based Design Methodology," in *Proceedings of 36th Design Automation Conference*, 1999, On page(s): 43.

[2] Altera Corporation, "Inverse Discrete Wavelet Transform BA114iDWT," 2004. [Online]. Available: http://www.altera.co.jp/products/ip/dsp/image_video_processing/m-bar-ba114idwt.html

[3] K. Lahiri, A. Raghunathan and S. Dey, "Design Space Exploration for Optimizing On-Chip Communication Architectures," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, On page(s): 952-961 Volume: 23, Issue: 6, June 2004.

[4] S. Parischa and N. Dutt, "Fast Exploration of Bus-Based Communication Architectures at the CCATB Abstraction," in *ACM Transactions on Embedded Computing Systems*, On page(s): 22-32 Volume: 7, No: 2, February 2008.

[5] F. Jafari, Z. Lu, A. Jantsch and M. H. Yaghmaee, "Buffer Optimization in Network-on-Chip Through Flow Regulation," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, On page(s): 1973-1986 Volume: 29 No: 12, December 2010.

[6] I. Saastamoinen, M. Alho, J. Nurmi, "Buffer implementation for Proteo network-on-chip," in *Proceedings of the 2003 International Symposium on Circuits and Systems*, 2003, On page(s): II-113 - II-116 Volume 2.

[7] K. Ueda, K. Sakanushi, Y. Takeuchi, and M. Imai, "Architecture-level Performance Estimation Method based on System-level Profiling," in *IEEE Proceedings Computers & Digital Techniques*, On page(s): 12-19 Volume 152 No 1, January 2005.

[8] K. Ueda, K. Sakanushi, Y. Takeuchi, and M. Imai, "Architecture-level Performance Estimation for IP-based Embedded Systems," in *Proceedings of Design, Automoation and Test in Europe Conference and Exhibition 2004*, 2004, On page(s):1002-1007 Volume 2.

[9] IEEE Computer Society, "IEEE Standard for Standard SystemC® Language Reference Manual," New York, USA, IEEE Standard Association, January 2012.

[10] K. Roy and S. Prasad, "Low-Power CMOS VLSI Circuit Design," USA, Wiley, 2000.

[11] K. Itoh, "VLSI Memory Chip Design," Germany, Springer, 2001.

[12] Synopsys, Inc, "Synopsys Product: Power Compiler," 2007. [Online]. Available: http://www.synopsys.com/tools/implementation/rtlsynthesis/pages/powercompiler.aspx.