BusMesh NoC: A Novel NoC Architecture Comprised of Bus-based Connection and Global Mesh Routers

SeungJu Lee[#1], Masao Yanagisawa[#2], Tatsuo Ohtsuki[#3] and Nozomu Togawa[#4]

[#1, 3, 4] *Dept. of Computer Science and Engineering, Waseda University,*

[#2] *Dept. of Electronic and Photonic Systems, Waseda University,*

N601A, 55 Bldg, 3-4-1 Okubo, Shinjuku, Tokyo, Japan

[1]lee@togawa.cs.waseda.ac.jp

***Abstract*— Network-on-chip (NoC) architectures are emerged as a promising solution to the lack of scalability in multi-processor systems-on-chips (MPSoCs). In this paper, A busmesh network-on-chip (BMNoC) architecture is proposed, together with simulation results. It is comprised of bus-based connection and global mesh routers to enhance the performance of on-chip communication. Furthermore, MPEG-4, H.264 and a hybrid application mixed MPEG-4 and H.264 on our architecture illustrates the better performance than earlier studies and feasibility of BMNoC.**

***Keywords*— Network-on-Chip (NoC), BusMesh NoC (BMNoC), A novel NoC architecture.**

I. INTRODUCTION

Future SoCs designed for ambient intelligence will be based on high-speed digital signal processing including audio/video coding/decoding, wireless communication, multilingual conversation. With the growing complexity in consumer embedded products and increasing transistor density, the leading edge System-on-Chip (SoC) architectures will be composed of many complex heterogeneous cores called Multi-Processor Systems-on-Chip (MPSoC). However, traditional communication architectures based on shared communication resources (single shared buses or hierarchy buses) or dedicated interconnections are not sufficient in many aspects, such as scalability, flexibility, communication performance and power efficiency. Furthermore, with technology scaling, the global interconnects cause severe on-chip synchronization errors, unpredictable delays, and high power consumption.

To mitigate these effects, the network-on-chip (NoC) approach emerged recently as a promising alternative to classical bus-based and point-to-point (P2P) communication architectures [1]. A Cluster-based NoC (C-NoC) [3] is created for the better performance of Network-on-chip. However, its performance is still insufficient for heterogeneous IP cores and clusters.

In this paper, we propose a novel approach for network-on-chips, comprised of bus-based connection and global mesh router, busmesh NoC. The busmesh NoC architecture, which analyses the data traffic, improves the system throughput compared to cluster-based NoC [3]. The result is verified through the simulation using a network simulator Ns-2 [12] and very encouraging simulation results have been observed.

II. RELATED WORK

In order to make every application running on an NoC follow the same basic operation principle defined in the backbone, several issues should be encapsulated into its design methodology. First of all, backbone architecture design should be scalable to the growing amount of traffic in a graceful manner. Also, flexibility is important to deal with the increasing complexity of an application. Various approaches have been explored for improving NoC performance: such as packet routing technique, application mapping and scheduling, topology synthesis, and flow control [1], [8]–[11].

In the previous studies, the Hermes NoC (H-NoC) switch [2] uses only one local port, which is not suitable in group communication. A cluster-based NoC (C-NoC) [3] switch, based on H-NoC switch, is created for the better performance of network-on-chip [3].

C-NoC switches have a routing logic and an arbitration logic in a control logic and the eight bidirectional ports. Four local ports connect IP cores to the switch and other four ports are connected to neighbouring switches. The ports contain the input and the output channels. The channels have buffers to store the information temporarily. The routing control logic implements the arbitration logic and a packet-switching algorithm same as H-NoC switch [2].

A. Problems of C-NoC

According to [3], comparing with the conventional NoC (SoCIN) [5], the area is decreased by 19.7% and the average of delay time is decreased by 15.1% in C-NoC [3]. However, C-NoC [3] has severe problems:

- The increased performance is still insufficient for heterogeneous IP cores and clusters.
- The practical system is composed of various-sized blocks and the floor-plan does not match the regular.

Thus, we propose busmesh NoC, a novel approach for network-on-chip. It configures a NoC architecture with clusters connected by mesh network and in intra-cluster, IP cores connected by buses. It provides the better performance such as scalability, flexibility, energy-consumption and latency.

III. BUSMESH NOC

978-1-4244-7456-1/10/$26.00 ©2010 IEEE

The basic idea of BMNoC is to develop a NoC architecture with clusters connected by mesh network, borrowing the hierarchy model from Internet and adapting it to communication networks. BusMesh NoC is to develop a NoC architecture with clusters connected by mesh network and IP cores in intra-cluster connected by buses. It provides the better performance such as scalability, flexibility, energy-consumption and latency. A generic architecture of BMNoC is depicted in Fig. 1. The network is composed of mesh routers (MRs), edge switches (ESs) and cluster nodes (CNs). With this architecture, the communication can be allocated in local and backbone network.

BMNoC architecture constraints are that each IP core has two I/O ports, each bus can connect up to three IP cores, each ES can connect up to three CNs and each ES can connect up to only one MR, based on [4].

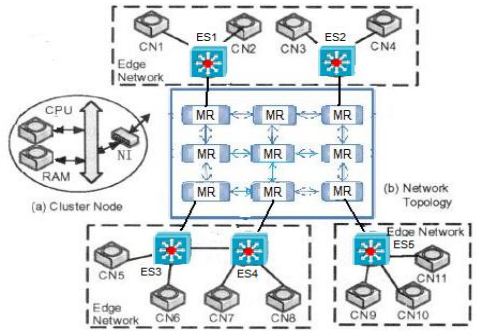

Fig. 1 A generic architecture of BMNoC.

A. Cluster node and network topology

The cluster is a group of hardware IP cores that couple tightly with each other and accomplish a specific task, together with corresponding firmware or software. A CPU connected to RAM and NI by bus, as shown in Fig. 1(a). Buses can efficiently cost connecting a few tens of components. Point-to-point connections between communication partners are practical for even fewer components. For intra-cluster communication, on-chip bus and point-to-point connection are more efficient than expensive on-chip router. The inter-cluster communication is handled by NoC components such as the NIs and routers. In general, a cluster node (CN) has several IP cores and a NI interconnected by bus, and communicates with other clusters through NI. The CN is organized by the computation complexity, communication requirement and functional relationship of IP cores. A single complex IP core wrapped with NI can form a CN, where the core and NI have point-to-point connection. With this organization, the functionality of each cluster in BMNoC is still described and synthesized along well established synchronous design flows, while the communication between clusters uses network components with asynchronous interface (router and link, mentioned below). Thus the tightly coupled clusters' interconnected by ESs within one MR will form an edge network (as shown in Fig. 1(b)) just like the tightly coupled IP cores are connected by the bus and form a cluster.

Through this classification, the traffic is limited to local cluster/network and the remote traffic will be as little as possible. Thus BMNoC could provide high performance network, and this shared network with independently clocked clusters (or IP cores) also facilitates modularity in large scale SoC designs.

B. Packet format

The packet has the source address (SrcAddr) and destination address (DstAddr) in its header, together with an optional field (Flag), as illustrated in Fig. 2, where M and N are determined for a specific BMNoC instance. The Flag field contains SPB (Service Priority Bit) indicating whether the current packet is guaranteed throughput (GT) or best-effort (BE). It also contains PktId (packet sequence number) which is needed by the partially adaptive routing that may disturb the order of packets. CRC field is also optional, and system can provide end-to-end error correction with it. Flit is the flow control unit, and the packet is divided into a number of flits (designate to k flits in Fig. 2) to be transmitted in the network. Each flit has N+2 bits: N for Data, and 2 for DataType field used to control the routing process. "01" ("10") indicates the first (last) flit of the packet, and "00" means the transmission is in process. The address (the SrcAddr and DstAddr filed) is the identification of each cluster node, and assigned according to its location in the network. Every single cluster has a unique address (ID) which contains the MRid, ESid and CNid.

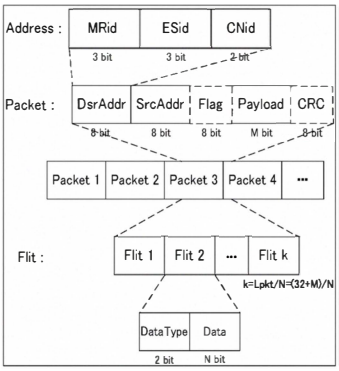

Fig. 2 Packet structure and data structure of BMNoC.

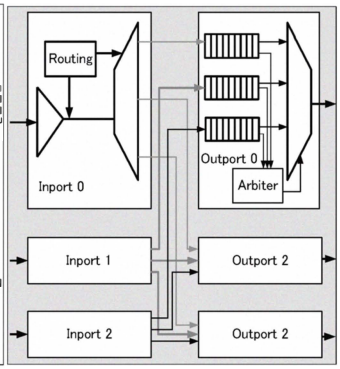

Fig. 3 MR and ES Architecture.

C. Mesh router (MR) and edge switch (ES)

Fig. 3 depicts the hardware architecture of MR and ES. A router uses output queuing (OQ) scheme to get better performance of bandwidth and latencies, and uses the wormhole switching scheme to optimize the buffer usage, because the router only needs to buffer several flits rather than the whole packet. For routing technique, we use a distributed and partially-adaptive routing. Both ES and MR can route for a packet only with its DstAddr field. The difference between MR and ES is the routing algorithm module in the inports.

MRs check a route table to route the packet. The route table contains possible routes between MRs, whereas ES compares its MRid and ESid fields with those of DstAddr to judge whether the packet is to forward to MR or local cluster [4].

IV. Evaluation Platform

A. Simulation setup

For the sake of illustration, we used a NoC based on a mesh topology and wormhole routing. In wormhole routed networks, each packet is divided into a sequence of flits which are transmitted over physical channels one by one in a pipeline fashion. A hop-to-hop credit mechanism guarantees that a flit is transmitted only when the receiving port has free space in its input buffer [7]. The cycle time in the simulator is defined as the transmission time of a single flit to cross from one PE to the next. Traffic is described with a triple of parameters: traffic intensity, number of IPs connected to the network and link bandwidth. The traffic intensity is the mean value of flit injection operated by IPs; this value is normalized to maximum value of one flit per clock cycle. The number of IPs connected to the network is depended upon an application and average packet latency will increase when the link bandwidth decreases. The reason is that a lower frequency will lead to a long flight time and receive time.

We assume each packet consists of 64 flits and each flit is 64 bits long and maximal link bandwidth of 200Mbits/s in the experiments. Moreover, packets are routed across the network using a deadlock free XY routing [7]. The Ns-2 [12] was used for carrying out simulation experiments.

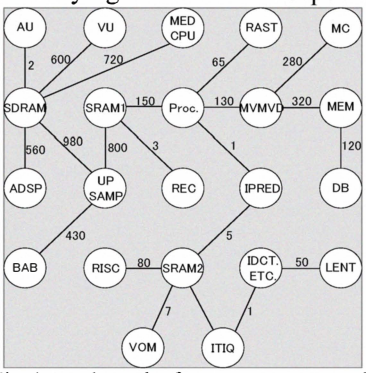

Fig. 4 A task graph of the hybrid application.

C-NoC			
Application	#Nodes	#Edges	#Routers and swithes
MPEG-4	12	15	3
H.264	10	13	3
A hybrid application mixed MPEG-4 and H.264	22	31	6
BMNoC			
Application	#Nodes	#Edges	#Routers and swithes
MPEG-4	12	20	4
H.264	10	17	4
A hybrid application mixed MPEG-4 and H.264	22	38	8

Tab 1.The comparison of C-NoC and BMNoC architectures.

We take a MEPG-4 [3], a H.264 [8] and a hybrid application mixed a MPEG-4 and a H.264 as part of the target application.

In Fig. 4, there is the core graph representation of a hybrid application mixed H.264 and MPEG-4 designs with annotated average communication requirements. Each edge represents the communication dependence with a value of communication volume divided by the bandwidth of a data channel. The total execution time of the hybrid application is 5324 cycles. We have applied the three applications to C-NoC and BMNoC and the result is shown in Tab 1. The BMNoC architectures applied a hybrid application mixed H.264 and MPEG-4 is shown in Fig. 5(a) and (b), respectively.

B. Packet latency

The packet latency of BMNoC equals to the time taken by a packet to go through a communication path from its source to its intended sink. It can be given by

$$G_{delay} = (m+1) \times (D_t + D_l) + \sum_{i=1}^{m} (D_q)_i$$

$$D_t = D_b + D_{es}$$

where m is the number of hops. As shown in Fig. 6, *transfer* (D_t), (D_b) and (D_{es}) refer to the delay to put a packet from a node to the link, the bus and the Edge Switch respectively. *On the Link* (D_l) refers to the delay on the link and *Enqueue-Dequeue* (D_q) refers to buffer delay in a switch. We assumed that D_l is so small as to be ignored.

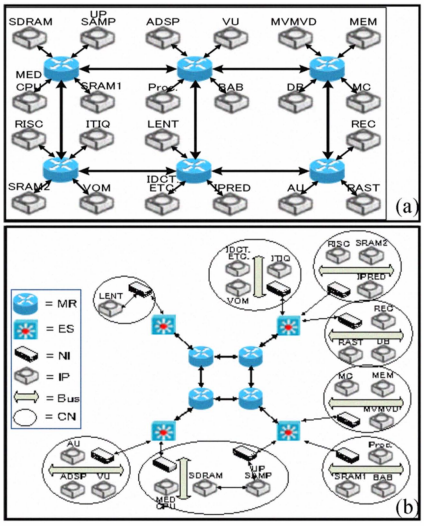

Fig. 5 C-NoC (a) and BMNoC (b) architectures applied a hybrid application mixed H.264 and MPEG-4.

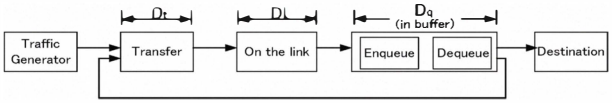

Fig. 6 General communication path in BMNoC.

V. Experimental Result

A. Buffer size analysis

To understand the effects caused by buffer size, we apply a hybrid application mixed MPEG-4 and H.264 to run simulation since it is more realistic compared to other two.

Fig. 7 shows the result of simulation on C-NoC and BMNoC under different buffer sizes and at a flit injection rate of 0.05 for comparison. As buffer size decreases, the latencies of BMNoC are not increased dramatically as the C-NoC did.

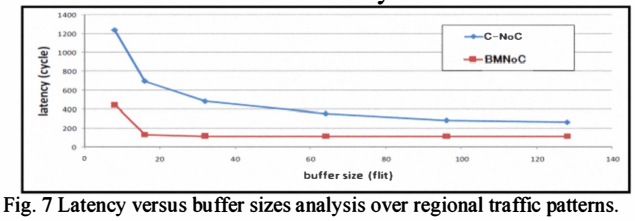

Fig. 7 Latency versus buffer sizes analysis over regional traffic patterns.

B. Experiments with real applications

According to the experimental results above, the channel configuring times have important impact on transmission latency. However, in real cases, data transmission is much regular than synthetic patterns. Therefore, we use a network

simulator Ns-2 to demonstrate the improvements of our BMNoC architecture compared to C-NoC in this section. The three tests used are MPEG-4, H.264, and a hybrid application mixed MPEG-4 and H.264 which were run on BMNoC and C-NoC, respectively. First, we input task graphs of the three applications and performed task mapping to map each task in the graph to a tile on C-NoC and BMNoC. Task mapping will put IPs with heavy communication on the same bus and connect other IPs with comparatively small communication by mesh topology via network interface to optimize communication cost on BMNoC. Then the resulting process graph was converted to a packetized traffic flow which was fed into a network simulator Ns-2. As shown in Fig. 8, BMNoC consistently provides the lowest average packet latencies among the three applications. This is because its connection with buses and routers has more flexibility for data transmission. The important result for BMNoC running the three applications is that BMNoC owns better latencies at all levels of packet injection rates.

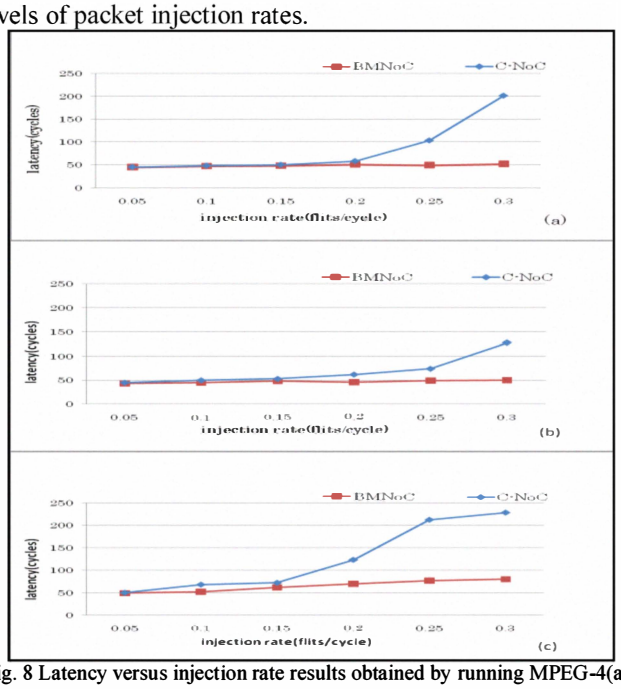

Fig. 8 Latency versus injection rate results obtained by running MPEG-4(a), H.264(b) and a hybrid application mixed MPEG-4 and H.264(c).

C. Area overhead

BMNoC and C-NoC architectures, applied the hybrid application mixed MPEG-4 and H.264, were implemented in TSMC 0.13 μm technology and synthesized using Synopsys Design Compiler under typical operating conditions. As shown in Fig. 9, the area numbers of these two router architectures are compared with C-NoC. We can find that our BMNoC has about 44% reduced area overhead C-NoC due to the total number of buffers dominates major part of area number.

VI. Conclusions

In this paper, we proposed a novel NoC backbone architecture BMNoC which comprised of Bus-based connection and global mesh router to enhance the performance of on-chip communication. Experimental results using both of the three real-applications and a network simulator Ns-2 verified that the proposed BMNoC architecture can significantly reduce the packet delivery latency at all levels of packet injection rates.

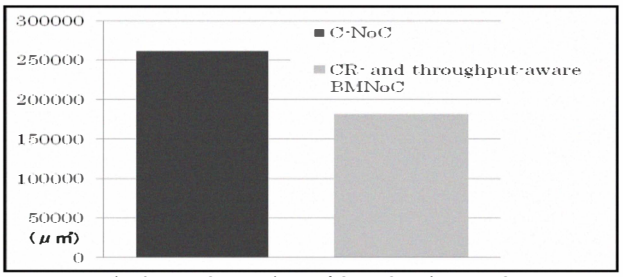

Fig. 9 Area Comparison of C-NoC and BMNoC applied a hybrid application mixed MPEG-4 and H.264.

VII. Acknowledgments

This research was supported by Waseda University Global COE program „International Research and Education Center for Ambient SoC' sponsored by MEXT, Japan.

References

[1] Radu Marculescu, Umit Y. Ogras, Li-Shiuan Peh, Natalie Enright Jerger, and Yatin Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design Integreted Circuits and System*, vol. 28, no. 1, pp. 3-21, Jan. 2009.

[2] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller and Luciano Ost, "HERMES: an infrastructure for low area overhead packet-switching networks-on-chip," *Integration The VLSI Journal,* vol. 38, pp. 69-93, Oct. 2004.

[3] Mohammad Reza Seifi and Mohammad Eshghi, "A clustered NoC in group communication," in *Proc. TENCON 2008*, pp. 1-5, Nov. 2008.

[4] Xianglun Leng, Ningyi Xu, Feng Dong and Zucheng Zhou, "Implementation and simulation of a cluster-based hierarchical NoC architecture for multi-processor SoC," in *Proc. ISCIT 2005,* vol. 2, pp. 1203-1206, Oct. 2005.

[5] Cesar Albenes Zeferino and Altamiro Amadeu Susin, "SoCIN: a parametric and scalable network-on-chip," in *Proc. SBCCI* 2003, pp. 169-174, Sep. 2003.

[6] J¨org Henkely, Wayne Wolfz and Srimat Chakradhary, "On-chip networks: A scalable, communication-centric embedded system design paradigm," in *Proc. ICVLSI Design*, pp.845-851, 2004.

[7] William J. Dally and Brian Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. Design Automation Conference,* pp. 684-689, 2001.

[8] D. Bertozzi, R. Tamhankar and L. Benini, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113-129, Feb. 2005.

[9] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computer Survey*, vol. 38, no. 1, pp. 1-51, Mar. 2006.

[10] U. Orgas, J. Hu, and R. Marculescu, "Key research problems in NoC design: a holistic perspective," in *Proc. CODES+ISSS*, pp. 69-74, Sep. 2005.

[11] H. G. Lee and N. H. Chang, "On-chip communication architecture exploration: a quantitative evaluation of point-to-point, bus and network-on-chip approaches," *ACM Transactions on DAES*, vol. 12, no. 3, pp. 1-20, Aug. 2007.

[12] Kevin Fall and Kannan Varadhan, "The *ns* manual", *The VINT Project*, Jun. 2001. *http://www.isi.edu/nsnam/ns/ns-documentation.html.*