A Hybrid and Hierarchical Network-on-Chip Architecture and its Configuration Algorithm
ハイブリッドかつ階層的なオンチップネットワークに関する研究

February 2014

Waseda University

Graduate School of Fundamental Science and Engineering

Department of Computer Science and Engineering,

Research on Imformation System Design

Seungju LEE

李　昇周

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Future SoCs designed for ambient intelligence will be based on high-speed digital signal processing including audio/video coding/decoding, wireless communication and 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. What is more, chip-wide synchronous operation is becoming extraordinarily difficult. 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 has recently emerged as a promising alternative to classical bus-based and point-to-point (P2P) communication architectures [19, 29]. According to the multi-core chip evolution, which forecasts tens and hundreds of cores in few years, logical and physical clusters of cores have been considered as an approach to support parallel processing. The next phase for this evolution can be called on-chip distributed computing, which

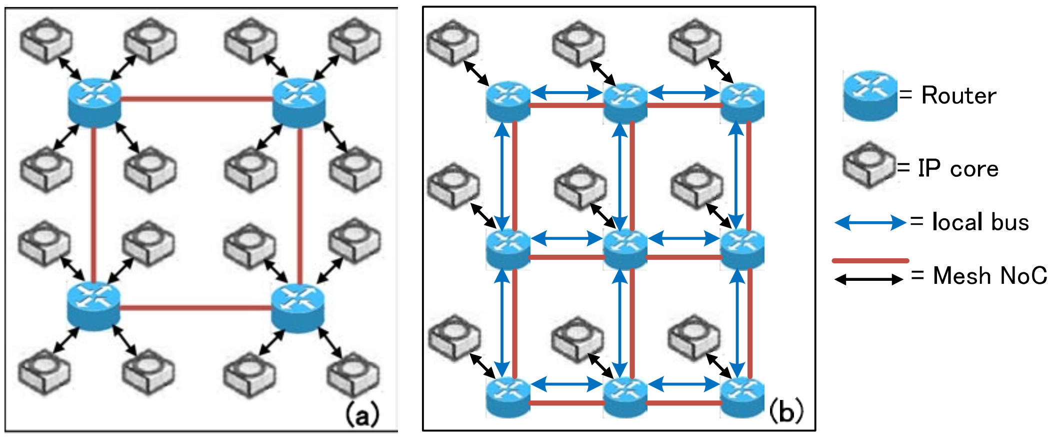


Figure 1.1: (a) C-NoC architecture and (b) HNoC architecture.

consists of distributed clusters of cores to process a large number of different workloads [11, 15, 16, 42]. A cluster-based NoC (C-NoC) a modified model of Hermes NoC (H-NoC) [31] is introduced by Seifi and Eshghi [42]. Each switch is attached to one local core in H-NoC. Every transmission must be performed via switches so the delay is increased. To solve this problem, the C-NoC switch has four local ports and configures a cluster as shown in Fig. 1.1(a). The latency of C-NoC [42] is decreased by 15.1% compared with the conventional NoC, SoCIN [50]. However, its performance is still insufficient for heterogeneous cores and clusters.

Among numerous NoC topologies, mesh is a popular one due to the simplicity and regularity. In mesh topology, each router is connected to several local cores and adjacent routers. Since the router only connects to its neighboring routers, the data packets transmitted from the source core to the destination core may travel long distance, and affects the performance of whole SoC [7]. Besides, with increasing size of NoC, the mesh topology has its disadvantage in the communication latency and the concentration of the traffic in the center of the mesh topology (named hot-spot). Although some studies have been made on effective core mapping methodologies to solve the hot-spot problem [18, 33, 34], what seems to be lacking is a better on-chip communication architecture. To solve the problems of the mesh topology for future

SoC design, many research papers have proposed a great number of approaches in recent years and a hybrid bus-NoC architecture is one of them [5, 26, 45, 49]. A hybrid bus-NoC architecture is a system platform which is based on a standard NoC architecture, and contains several clusters which are composed of local buses. In the hybrid bus-NoC architecture, cores with heavy traffic and communication volume are placed in the same cluster with a local bus to avoid hot-spots and reduce the transmission latency. Since the hybrid architecture is based on a standard mesh NoC concept, the router of the hybrid system not only connects with its neighboring routers but also connects to a cluster which is composed of several cores and a local bus. It is noteworthy that new interface is not needed for each core in subsystem, and it can further reduce the design cost.

It is very important for a configuration algorithm for hybrid bus-NoC architectures to decide which cores should be assigned to the same cluster and map the clusters onto the network that the transmission latency is minimized and the locality is contemplated. When partitioning cores, the cores with heavy traffic and communication volume should be assigned to the same cluster and placed as close as possible to reduce the transmission latency.

Unfortunately, the conventional NoC configuration algorithms [3, 17, 21, 22, 27, 33, 36, 44, 45] are not suitable for a hybrid bus-NoC architecture since they are only greedy of communication volume or bandwidth requirement. The basic idea of mapping algorithm in [45] and [25] are the same as [33]. The approach in [33] maps each core to NoC architecture one by one, in *breadth-first-search manner*, only in the order of its communication volume after picking up the two cores which are connected by the highest communication volume in an input task graph. The algorithm in [33] is used for a hybrid bus-NoC architecture as a makeshift now. It may cause a poor result such as a longer latency and a low throughput since the greedy approach does not contemplate the localities between cores. Thus, we propose a novel configuration algorithm for a hybrid bus-NoC architecture called BMNoC and decide to compare the performance of our proposed algorithm with

[33], which is used for a hybrid bus-NoC architecture as a makeshift now, to verify its advantage.

In this dissertation, first of all, we decribe and define our target architecture, which is called a busmesh NoC (BMNoC). BMNoC is a generalized version of hybrid bus-NoC architectures. After that, we propose a configuration algorithm for hybrid bus-NoC architecture (BMNoC configuration algorithm) which analyses the data traffic of the target application and determines which core is the right one to put into a certain cluster with its communication volume and locality. In many applications, we have several cores which communicate with each other more frequently than with other cores, such as a processor and cache memories. If there is a higher communication volume between some specific cores, our BMNoC configuration algorithm reduces network traffic and latency by mapping the cores into the same *local* cluster node and connecting them with a local bus. Our BMNoC configuration algorithm reduces the traffic load at the center of the network by a hierarchical communication network.

Performance of many-core chip multiprocessors (CMPs) is significantly influenced by performance of NoC which interconnects each core for on-chip communication. Specially, latency of packets in NoC remarkably effect performance of applications executing on a many-core CMP. Because it is projected that executing different applications concurrently on a many-core CMP becomes highly required in the future, it will be important that a priority control which guarantees or differentiates latency of packets. Thus, a new packet transmission priority control method for NoCs has been proposed in [43] which can improve the efficiency of the buffers.

We also propose a novel BMNoC utilizing packet transmission priority control method. Our proposed BMNoC is a generalized and simplified version of a hybrid NoC which is composed of local buses and global mesh routers. Our proposed architecture is composed of clusters which are connected by mesh network, borrowing the hierarchical model from the Internet and adapting it to communication networks. In intra-cluster node, several cores which have a heavy communication to each other are connected by a local bus. It can provide the better performance in terms of the latency since local buses transmit data directly to other cores in the same cluster node with a parallel fashion, which eliminates packetizing overhead. Furthermore, Our proposed BMNoC utilizing packet transmission priority control method can minimize the average packet latency by improving the efficiency of the buffers.

This dissertation is organized as follows:

Chapter 2 [**BusMesh NoC: An NoC Architecture Composed of Bus-based Connection and Global Mesh Routers**] describes our proposed busmesh NoC which is a novel NoC architecture composed of bus-based connection and global mesh routers. We explain a busmsh NoC (BMNoC) at first. BMNoC is a generalized version of a hybrid NoC with local buses. Recently, the mainstream of NoC architecture is a hybrid bus-NoC architecture, which is based on a hierarchical model from Internet. We introduce several conventional NoC architectures. We define a novel hybrid bus-NoC architecture, BMNoC, in this chapter. We evaluate BMNoC as compared with conventional architectures with Ns-2 simulator. **Chapter 3** [**A Locality-aware NoC Configuration Algorithm Utilizing the Communication Volume among IP Cores**] describes our proposed BMNoC configuration algorithm. We propose a locality-aware NoC configuration algorithm (BMNoC configuration algorithm). BMNoC configuration algorithm for hybrid bus-NoC architectures analyses the data traffic of the target application and configures communication networks with cluster nodes, edge switches and mesh routers to establish a hierarchical structure. Second, we introduce a breadth-first approach algorithm for comparison purpose. We evaluate algorithms which are applied realistic applications with Ns-2 simulator. **Chapter 4** [**BMNoC utilizing Packet Transmission Priority Control Method**] describes our proposed BMNoC configuration algorithm utilizing packet transmission priority control method. We explain packet transmission priority control methodat first. Packet transmission priority control method improves average delay by enhancing the efficiency of the buffers are proposed for NoCs. We propose a novel BMNoC which is utilizing packet transmission priority control method. We evaluate our proposed architectures and algorithms with Ns-2. **Chapter 5** [**Related Works**] introduces related works. We introduce conventional NoC architectures, algorithms at first. We explain why a novel NoC architecture and its configuration algorithm are needed. **Chapter 6** [**Conclusion**] summarizes our research and indicates future works.

Chapter 2

BusMesh NoC: An NoC Architecture Composed of Bus-based Connection and Global Mesh Routers

In this chapter, we define a hybrid bus-NoC architecture, called a busmesh NoC (BMNoC), which is a generalized version of a hybrid NoC with local buses. Recently, the mainstream of NoC architecture is a hybrid bus-NoC architecture, which is based on a hierarchical model from Internet [5, 25, 26, 27, 45, 47]. The hybrid bus-NoC architecture, called a busmesh NoC (BMNoC), is derived from the generalization of the previous topologies [5, 25, 26, 27, 45, 47].

In BMNoC, clusters which are composed of several cores are connected by a standard mesh topology, borrowing the hierarchical model from the Internet and adapting it to communication networks. BMNoC is composed of cluster nodes (CNs), edge switches (ESes) and mesh routers (MRs) to establish a hierarchical communication network. In intra-cluster node, several cores which have a heavy communication with each other are connected by a local bus. It can provide the better performance such as the latency and the throughput since local buses transmit data directly to other cores in the same cluster node in a parallel fashion, which eliminates packetizing overhead.

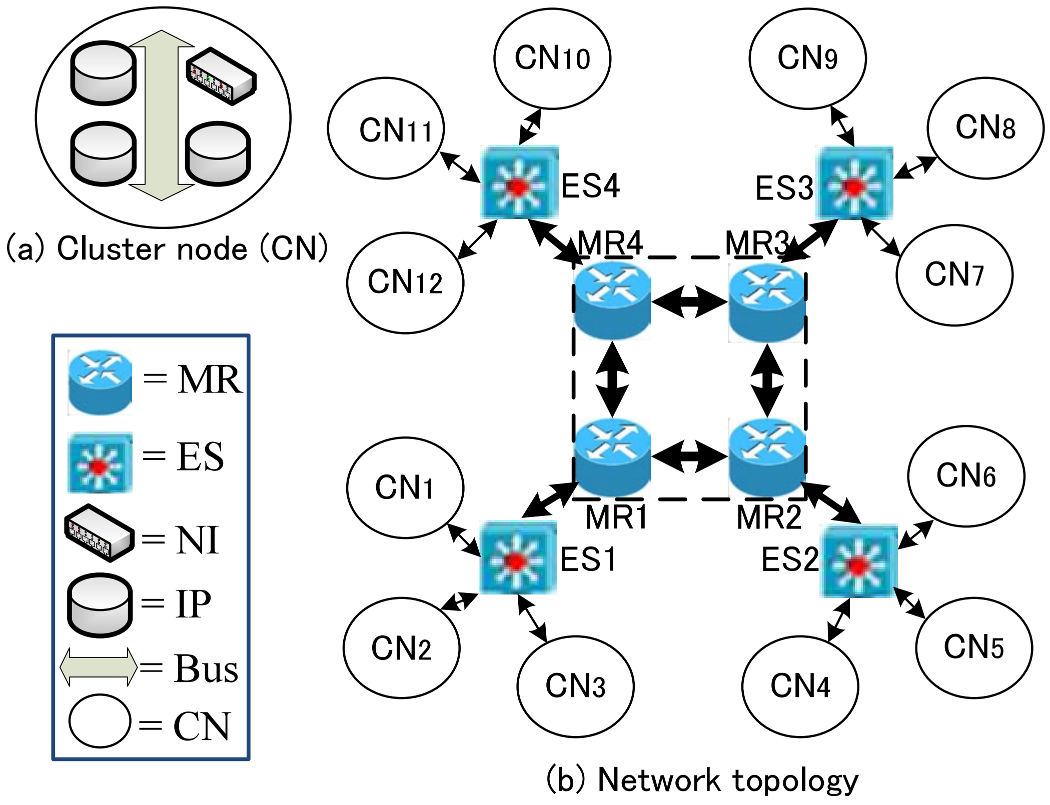


Figure 2.1: A generic architecture of BMNoC.

2.1 BMNoC architecture

A generic architecture of BMNoC is depicted in Fig. 2.1. The network is composed of mesh routers, edge switches and cluster nodes and in a cluster node, there exist several cores which have a heavy communication volume between them. Since the communication can be allocated in local and backbone network with the hierarchical structure, BMNoC can improve the performance in terms of the latency and the throughput as compared with the standard NoCs [5].

2.1.1 Cluster node (CN) and network topology

A cluster node is a group of hardware cores that couple tightly with each other and accomplish a specific task, together with corresponding firmware or software. For intra-cluster node communication, on-chip bus connection is more efficient than expensive on-chip router. Then several cores which have a heavy communication volume with each other are connected by a *local bus* with a network interface (NI), as shown in Fig. 2.1(a). The cluster node is organized by the computation complexity, communication requirement and functional relationship of cores. The inter-cluster node communication is handled by NoC components such as the NIs and routers; edge switches and mesh routers [39].

The tightly coupled cluster nodes interconnected by edge switches and mesh routers will form a hierarchical network as shown in Fig. 2.1(b), just like the tightly coupled cores are connected by local buses, and form a cluster node. Through this classification, the traffic is limited to local cluster/network and the remote traffic will be as little as possible. BMNoC could provide high performance network, and this shared and classified network with independently cluster nodes also facilitates modularity in large scale SoC designs.

2.1.2 Packet format

The packet has the source address (SrcAddr) and destination address (DstAddr) in its header, together with an optional field (Flag), for example as illustrated in Fig. 2.2, 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. 2.2) 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

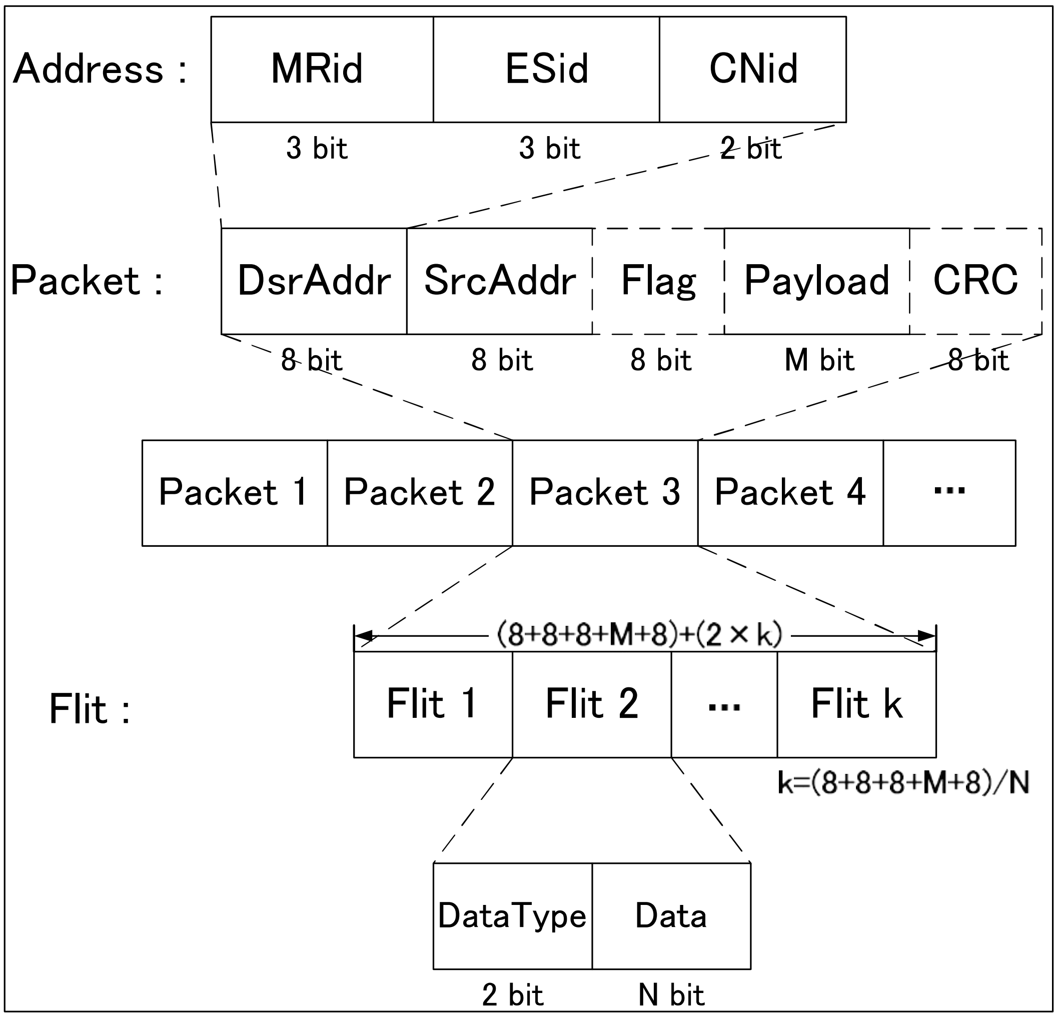


Figure 2.2: Packet structure and data structure of BMNoC.

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.

2.1.3 Mesh router (MR) and edge switch (ES)

Fig. 2.3 depicts the hardware architecture of MR and ES. MR and ES have the same structure. 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 [29]. 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.

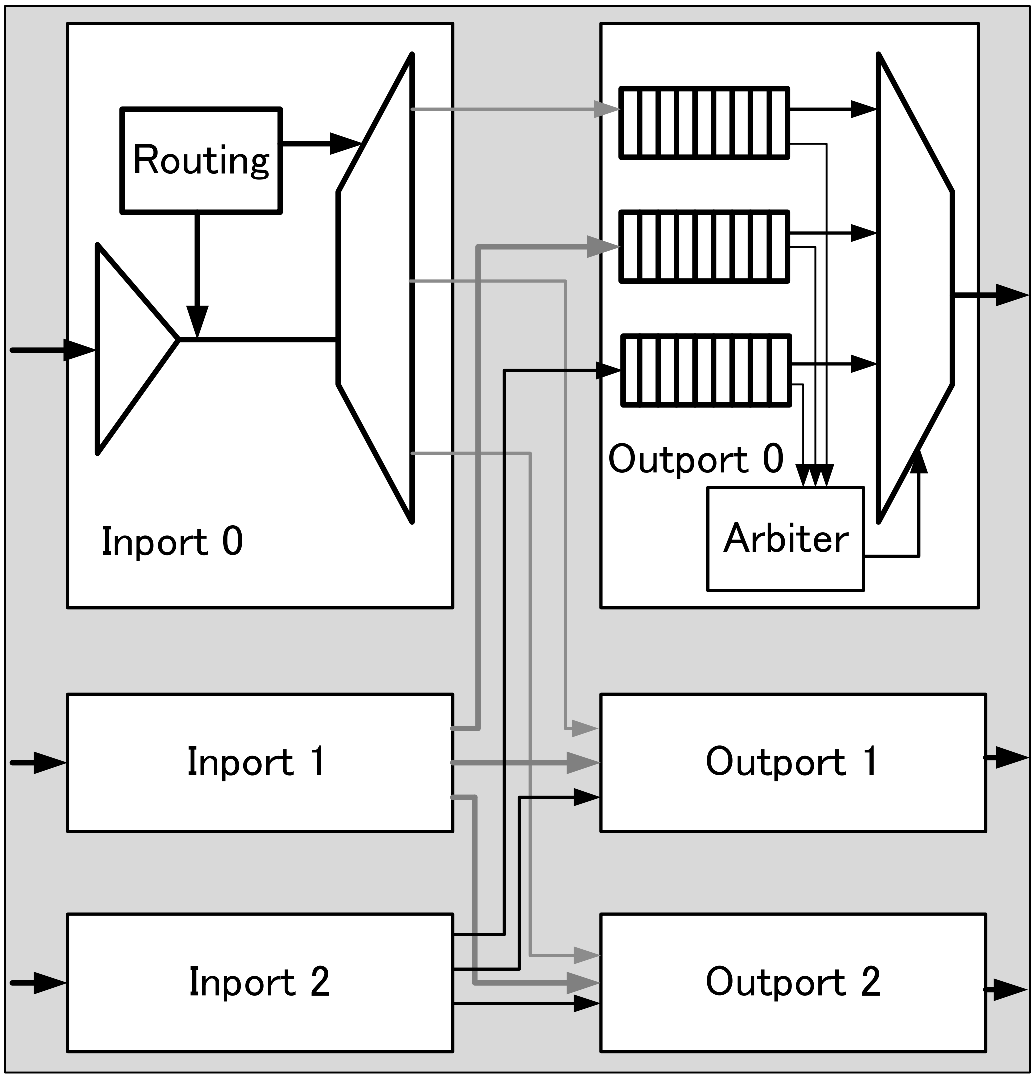


Figure 2.3: MR and ES architecture.

2.2 Experiments

2.2.1 Experiment setup

In these experiments, we used 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 [9]. We assume each packet consists of 32 flits, each flit is 16 bits long and maximal link bandwidth of 200Mbits/s at 100MHz operation. Our router requires five cycles to route a flit. The buffer size of edge switches and mesh routers is 32 flits. We set up the bus arbitrations consume two cycles from request to address transmission and then one cycle later, data transmission is executed, resulting in three cycles from request to data transmission as in [37, 41]. We set up the routers such as edge switches and mesh routers require four cycles to process a header flit and one cycle to route a packet across the wires of the appropriate port as in [37, 41].

Throughout the experiments, Ns-2 [12] was used for carrying out simulation experiments. Ns-2 is the most typical network simulator which has facilities to describe network topology, network protocols, routing algorithms and communication traffic generation. It provides many mechanisms for modelling traffic generation [1].

2.2.2 Experimental evaluations: C-NoC, HNoC, and BMNoC

Here, we evaluate the performance of BMNoC using two realistic applications, *auto industry* and *telecom* which are widely used benchmarks for NoC experiments. We compare them with conventional NoCs such as a cluster-based NoC (C-NoC) [42] and a hybrid NoC (HNoC) [49]. Both of these applications retrieved from

Table 2.1: The comparison among C-NoC, HNoC, and BMNoC.

auto industry				
Architectures	**# of routers**	**# of CNs**	**# of ESes**	**# of MRs**
C-NoC	10	–	–	–
HNoC	40	–	–	–
BMNoC	–	10	5	5

telecom				
Architectures	**# of routers**	**# of CNs**	**# of ESes**	**# of MRs**
C-NoC	8	–	–	–
HNoC	32	–	–	–
BMNoC	–	8	4	4

E3S benchmark suite [10] where *auto industry* has 40 vertices and *telecom* has 32 vertices. These applications, *auto industry* and *telecom* , are mapped onto $(4, 2, 1)$-BMNoC, using our proposed BMNoC configuration algorithm. Each cluster node includes four vertices as in the previous papers [25, 27, 47] i.e., $k = 4$. Then, the number m of edge switches connected to a single mesh router is set 1 i.e., $m = 1$ as in [27, 45] and the number l of cluster nodes connected to a single edge switch is set on 2 to implement a hierarchical structure i.e., $l = 2$. In this evaluation, four vertices are in the same cluster node, two cluster nodes are connected to the same edge switch and each edge switch is connected to the one mesh router.

We have applied the two applications to C-NoC, HNoC and our BMNoC. The result is shown in Table 2.1. As in [14, 20, 32], the variation of *average packet latency* as a function of traffic injection rates for *auto industry* is given in Fig. 2.4. Average packet latency equals the average cycles taken by a packet to go through a communication path from its source to its intended sink. For example, when we have an edge $e = (u, v)$ and its communication volume $c(e)$ in a given task graph,

$c(e)$ packets go through its commucation path from u to v and we can obtain its required cycles $t(e)$. Then we obatin its average cycle by calculating $t(e)/c(e)$. We calculate the average cycles for all the edges in a given task graph and then finally have an average packet latency by averaging them.

The average number of packets which are transferred among CNs, ESes and MRs are 152.5, 33 and 12, respectively for *auto industry*. The total time to complete all traffic pattern of *auto industry* is 141,400 cycles.

These plots show that our BMNoC shifts the critical traffic load from 0.27 to 0.38 (approximately 29% improvement) as compared to C-NoC and from 0.34 to 0.38 (approximately 11% improvement) as compared to HNoC. Similarly, the average packet latency for BMNoC is consistently smaller as compared to conventional NoCs such as C-NoC and HNoC. For instance, at 0.25 packet injection rate, the latency drops from 88 to 54 cycles, giving about 39.0% reduction as compared to C-NoC.

Regulating the communication volumes of cores hierarchically is very important not to congest the network with generated packets. At low traffic loads, the average packet latency exhibits a weak dependence on the traffic injection rate. However, when the traffic injection rate exceeds a critical traffic load, the packet delivery cycles rise abruptly and the network throughput starts collapsing.

Similar improvements have been observed for *telecom* as shown in Fig. 4.9. The average number of packets which are transferred among CNs, ESes and MRs are 104.2, 21.4 and 8.2, respectively for *telecom*. The total time to complete all traffic pattern of *telecom* is 95,900 cycles. Specifically, the critical traffic load is improved from 0.43 to 0.63 showing approximately 32% improvement as compared to C-NoC and from 0.57 to 0.63 showing approximately 10% improvement as compared to HNoC. Likewise, the latency at 0.35 traffic injection rate drops from 95 to 59 cycles as compared to C-NoC.

For reference, BMNoC, HNoC and C-NoC architectures, applied *auto industry* and *telecom*, were evaluated in TSMC 0.13 μm technology and synthesized using

Table 2.2: Area comparisons among C-NoC, HNoC and BMNoC

auto industry					
Architectures	**# of buffers**	**Area** (mm^2)			
		IP cores	Routers/ESes/MRs/NIs	Bus	Total
C-NoC	38	27.6403	3.2209	–	30.8612
HNoC	100		5.3212	–	32.9523
BMNoC	30		2.7929	0.192	30.6252
telecom					
Architectures	**# of buffers**	**Area** (mm^2)			
		IP cores	Routers/ESes/MRs/NIs	Bus	Total
C-NoC	32	21.4903	1.9859	–	23.4762
HNoC	82		3.6189	–	25.1092
BMNoC	18		1.4368	0.160	23.0871

Synopsys Design Compiler. We consider the area for the bus architecture for area evaluation as in [23]. As shown in Table 2.2, we can find that our BMNoC has more reduced area overhead as compared with C-NoC and HNoC due to the total number of buffers dominates major part of area.

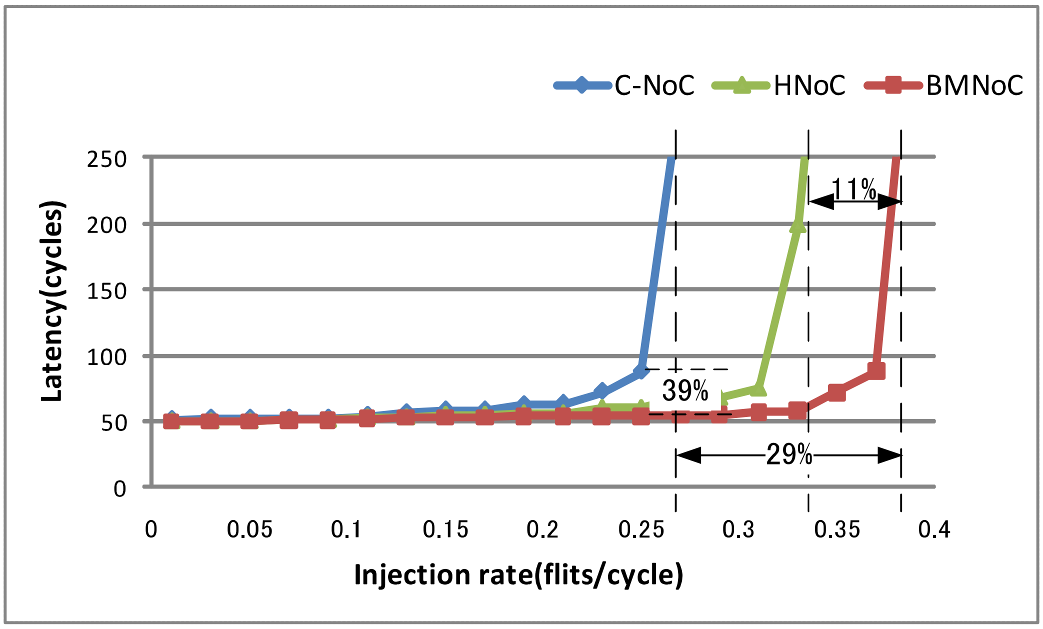


Figure 2.4: Traffic injection rate versus average packet latency for *auto industry* benchmark.

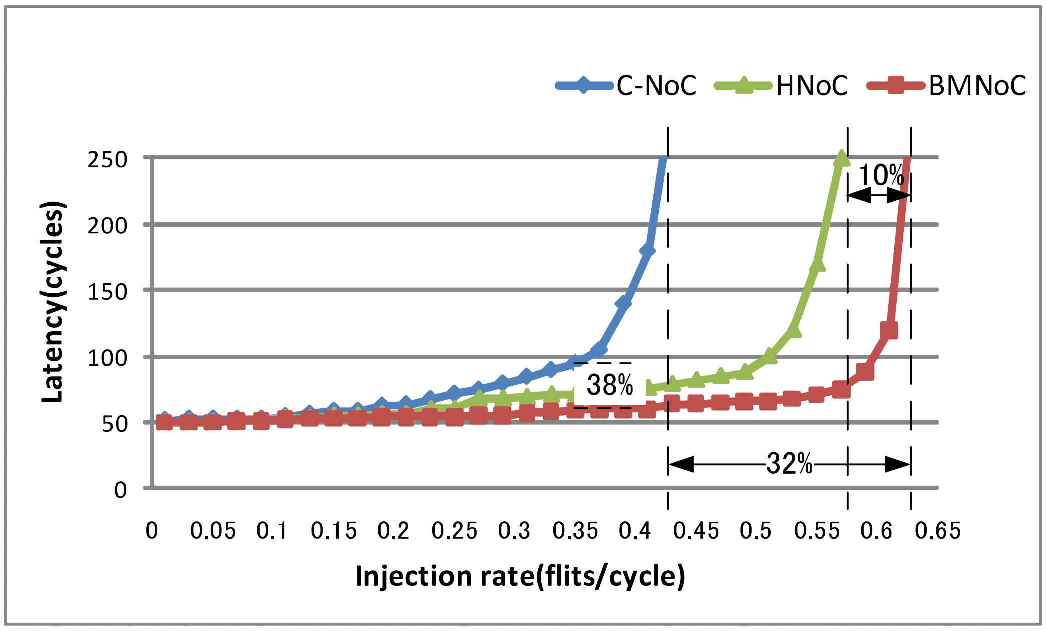


Figure 2.5: Traffic injection rate versus average packet latency for *telecom* benchmark.

2.3 Concluding remarks

In this chapter, we has defined a hybrid bus-NoC architecture, called a busmesh NoC (BMNoC), which is a generalized version of a hybrid NoC with local buses. Recently, the mainstream of NoC architecture is a hybrid bus-NoC architecture. 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. BMNoC 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 and latency. 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. Experimental results using the two realistic applications verified that the critical traffic load and the average packet latency of BMNoC are consistently smaller as compared to conventional NoCs such as C-NoC [42] and HNoC [49]. Moreover, our BMNoC has more reduced area overhead as compared with conventional NoCs.

Chapter 3

A Locality-aware NoC Configuration Algorithm Utilizing the Communication Volume among IP Cores

In this chapter, we propose a locality-aware NoC configuration algorithm utilizing the communication volume among IP cores. When we configure BMNoC architecture based on a given application task set, we need to consider traffic congestion in the network, the node connectivity and the diameter which directly affect the transmission of BMNoC in order to decide the best packet route. A long packet path in an NoC may increase the final transmission time between cores. Adapting the topology for the communication pattern of these applications onto BMNoC can be the best solution to reduce this long path.

As we desribed in Chapter 2, BMNoC architecture is based on a hierarchical model from the Internet and we adapt it to communication networks. It has high *locality* in each cluster and a mesh network composed of switches and routers. However, recently proposed NoC configuration algorithms [3, 17, 21, 22, 33, 36, 44] cannot be applied to BMNoC since they consider only communication volume among cores when they configure an architecture. In BMNoC, the latency across switches and routers and the localities of cores should be considered in determining which core is the right one to put in a certain cluster.

3.1 Problem formulation

Now we define our BMNoC configuration problem. A *task graph* $G = (V, E)$ is a graph, where each vertex $v \in V$ shows a task and, if there are some communications between two tasks v_1 and v_2, G has an edge $e = (u, v) \in E$. Each edge $e = (u, v)$ has a commucation volume denoted by $c(e)$ or $c((u, v))$ which shows communication volume between the two tasks u and v. In this paper, we assume that each task is executed by an identical hardware core, i.e., there is a one-to-one mapping between each task and each core. In other words, we can use a core and a task interchangably. An example of task graph is shown in Fig. 3.1

Users can specify several control parameters; the maximum number k of cores connected to a single bus, i.e., a cluster node is composed of up to k cores, the number l of cluster nodes connected to a single edge switch, and the number m of edge switches connected to a single mesh router. Then a *parameterized BMNoC* is denoted as (k, l, m)-BMNoC. If a mapping from a task graph to a particular (k, l, m)-BMNoC is given, we can obtain a *latency* which can be defined by average cycles required between any two tasks having communications. Then we can define our BMNoC configuration problem as follows:

Definition 1 *For given a (k, l, m)-BMNoC and a task graph, our BMNoC configuration problem is to map a vertex in the task graph to a (k, l, m)-BMNoC so as to minimize its latency.*

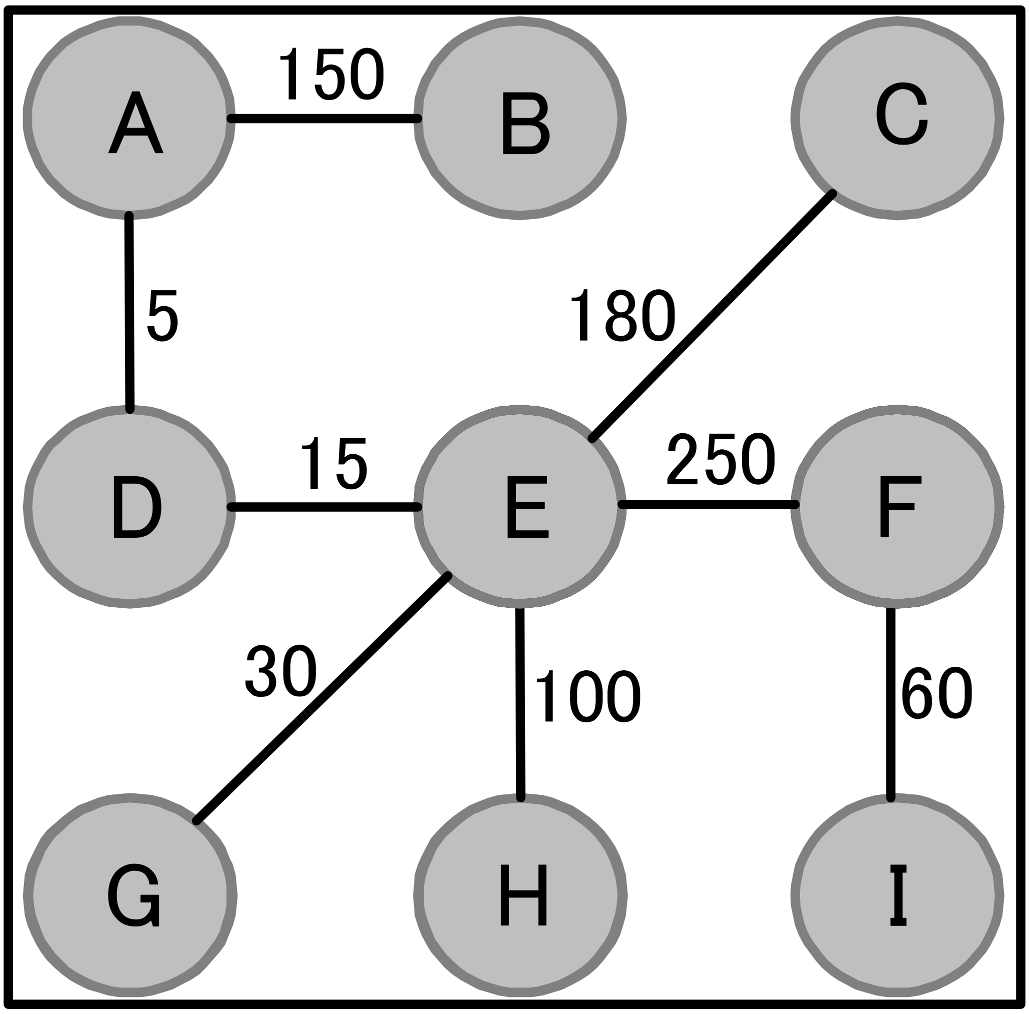


Figure 3.1: An example of task graph.

3.2 Strategy

In order to solve the BMNoC configuration problem above, we have roughly two algorithm candidates; one is *top-down partitioning*, such as min-cut partition, and the other is *bottom-up clustering*. A poor result early in top-down partitioning process imposes an unnatural circuit hierarchy and will likely lead to a suboptimal solution as in [49]. On the other hand, bottom-up clustering algorithms provide a solution to the problems encountered when partitioning very large circuits as in [49]. A bottom-up clustering algorithm can be naturally integrated into clustering local vertices and making a hierarchical structure. We can say that bottom-up clustering is superior to top-down patitioning in BMNoC.

In bottom-up clustering, there exist two basic approaches: One is *seed-based clustering* which is shown in Fig. 3.2(a) and the other is *growing-region clustering* which is shown in Fig. 3.2(b). While capable of achieving tight packings, seed-based clustering is greedy and may become trapped in local minima. Particularly, each independent seed and its surroundings are localized but it is very hard to configure a hierarchical structure that meets parameterized BMNoC. On the other hand, growing-region clustering can consider both communication volume and the *localities* of vertices. We can say that growing-region clustering is superior to seed-based clusterings in BMNoC [50].

In [33], a kind of growing-region-clustering-based approach has been proposed for NoCs. After picking up the two vertices in an input task graph connected by the highest communication volume, the approach in [33] maps each vertex to NoC architecture one by one in a *breadth-first-search manner* in the order of its communcation volume. It may cause a poor result such as a longer latency and a low throughput since a direct breadth-first approach does not contemplate the localities between vertices.

Based on the discussions above, we propose a BMNoC configuration algorithm in this section. In our proposed algorithm, we configure a hierarchical structure

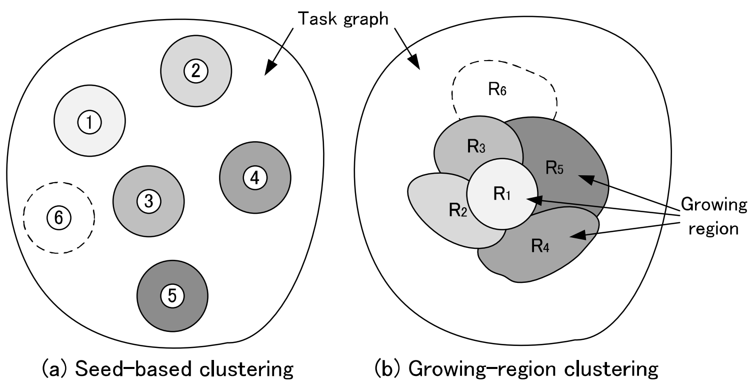


Figure 3.2: Examples of bottom-up clusterings. (a) Seed-based clustering. (b) Growing-region clustering. In (a), we first pick up a seed vertex and find its local surroundings as a cluster. Then we pick up another seed vertex and find its local surroundings as the next cluster. We repeat this procedure until all the vertices are included in any cluster. In (b), we first pick up a seed region R_1. After that, we find out the next region R_2, adjacent to R_1. By repeating this process several times, the regions $(R_1, R_2, \cdots, R_i)$ will make a single cluster. Similarly, the regions $(R_{i+1}, \cdots, R_{i+j})$ will make a next single cluster. We repeat this procedure until all the vertices are included in any region and any cluster. We can naturally consider the locality of vertices and adjust a cluster size in (b), whereas it is very difficult to do so in (a).

of BMNoC which is composed of both a global mesh network and local buses to improve the latency of BMNoC. Cores having a heavy communication volume between them are mapped into the same cluster node with a local bus. Cluster nodes can have communication with each other via edge switches and mesh routers. By this hybrid and hierarchical structure, BMNoC configured by our proposed algorithm can own the better performance than conventional NoCs.

3.3 Algorithm overview

Now we describe our proposed BMNoC configuration algorithm overview. As discussed in the previous subsection, our proposed algorithm adopts bottom-up growing-region clustering as in Fig. 3.2(b).

In our proposed algorithm, each growing region includes p or fewer vertices where $p \leq k$ and k shows the maximum number of cores connected to a single bus. We map them to the same cluster node. Firstly, we configure an empty cluster node CN_1 and find the first growing region R_1. Then we map all the vertices in R_1 to CN_1. Next we find the second growing region R_2 and map their vertices to CN_1. If CN_1 is full, we configure an empty cluster node CN_2 and connect it to CN_1 by using an edge switch. We find the next growing region and map their vertices to CN_2 and contitue this process until all the vertices in a given task graph are mapped to BMNoC. In this process, we configure another edge switch every time we configure l cluster nodes. We configure another mesh router every time we configure m edge swithces. Mesh routers are configured by a *reversed-snail array* in its order [13]. In this approach, as the regions in the task graph grow, its BMNoC architecture also grows. Thus adjacent regions are mapped onto adjacent BMNoC structure very naturally and hierarchical locality is preserved.

The problem here is how to find the next growing region. Assume that we already have growing regions R_1 to R_5 as shown in Fig. 3.2(b) and their vertices are already mapped onto BMNoC. Assume that we are now configuring the cluster node CN_i and we can map p or more vertices to CN_i. Assume that CN_i is connected to the edge switch ES_j and ES_j is connected to the mesh router MR_k. Now let us consider the next growing regtion R_6. R_6 has at most p vertices. Then the growth rate $GR(R_6)$ of R_6 is defined by:

$$GR(R_6) = \sum_{v \in R_6} cost(v), \tag{3.1}$$

where

$$cost(v) = \sum_{\text{each edge } e \in E \text{ connected to } v} cost(e). \tag{3.2}$$

Suppose that all the vertices in R_6 are mapped onto CN_i, i.e., $v \in R_6$ is mapped onto CN_i. Then the edge cost $cost(e)$ for $e = (u, v)$ is defined by:

$$cost(e) = \begin{cases} C_1 \times c((u,v)) & \text{if } u \text{ is already mapped to } CN_i. \\ C_2 \times c((u,v)) & \text{if } u \text{ is not mapped to } CN_i \\ & \text{but to the same edge switch } ES_j. \\ C_3 \times c((u,v)) & \text{if } u \text{ is neither mapped to } CN_i \\ & \text{nor to the same edge switch } ES_j, \\ & \text{but to the same mesh router } MR_k. \\ C_4 \times c((u,v)) & \text{if } u \text{ is already mapped to} \\ & \text{somewhere in BMNoC but} \\ & \text{does not fall into the above.} \\ 0 & \text{ohterwise.} \end{cases} \tag{3.3}$$

We design $C_1 \gg C_2 \gg C_3 \gg C_4$ in the cost function above.[1] When more than two combinations have the highest growth rate to make the next growing region, we grade the combinations according to their localities. [2]

[1]In our implementation result, we set $C_1 = 100$, $C_2 = 10$, $C_3 = 5$, and $C_4 = 1$.

[2]For example, two combinations R_x and R_y have the same highest growth rate, 500;

```
If $GR(R_x) = GR(R_y) = 500$;
Assume $C_1 = 100, C_2 = C_3 = C_4 = 0$ and re-calculate $GR(R_x)$ and $GR(R_y)$;
  if $GR(R_x) > GR(R_y)$; The next growing region is $GR(R_x)$.
  if $GR(R_x) < GR(R_y)$; The next growing region is $GR(R_y)$.
  if $GR(R_x) = GR(R_y)$;
  Assume $C1 = 100, C2 = 10, C3 = C4 = 0$ and re-
  calculate $GR(R_x)$ and $GR(R_y)$;
    if $GR(R_x) > GR(R_y)$; The next growing region is $GR(R_x)$.
    if $GR(R_x) < GR(R_y)$; The next growing region is $GR(R_y)$.
    if $GR(R_x) = GR(R_y)$;
    Assume $C1 = 100, C2 = 10, C3 = 5, C4 = 0$ and re-
```

Then a growth rate GR is heavily dependent on the locality of already mapped vertices. The more communications with local vertices we have, the higher the growth rate becomes. We try all the combination of p vertices in the unmapped vertices in the task graph and pick up the one with the highest growth rate as the next growing region R_6. Then we map the p vertices in R_6 onto the current cluster node CN_i.

An example of the growth rate $GR(R_6)$ of R_6 is shown in Fig. 3.3 in the case of $(3, 2, 1)$-BMNoC and $p = 2$.

How to determine the value of p is the next problem. This must be dependent on the size of a task graph. Roughly saying, p must be up to three in large task graphs since enumeration of all combination of three vertices requires $O(|V|^3)$ time. Based on this discussion, we set $p = 3$ as in our experiments.

3.3.1 The algorithm

In this subsection, we describe our proposed algorithm with a detailed example as shown in Figs. 3.4–3.7. We consider here the $(4, 2, 1)$-BMNoC and p is set to two ($p = 2$) for simplicity. We also assume that $C_1 = 100$, $C_2 = 10$, $C_3 = 5$, and $C_4 = 1$. The variable z means how many vertices can be mapped more to the same cluster node. For example, when $z = 4$, we can map four more vertices to the same cluster node.

In the initial step (*Step 1* in Fig. 3.4), we configure an empty cluster node CN_1 and map two vertices A and B which have the highest communication volume in the task graph. After that (*Step 2* and *Step 2.2* in Fig. 3.4), we compute

calculate $GR(R_x)$ and $GR(R_y)$;

if $GR(R_x) > GR(R_y)$; The next growing region is $GR(R_x)$.

if $GR(R_x) < GR(R_y)$; The next growing region is $GR(R_y)$.

if $GR(R_x) = GR(R_y)$;

The next growing region is GR which is made at first.

how many vertices can be mapped more to the same cluster node CN_1. Since we can map two more vertices to CN_1, i.e., $z = 2$, we will find a growing region which includes just two unmapped vertices. We compare the growth rate defined by Eqn. (3.1) for every two unmapped vertices and pick up the one with the highest growth rate. In our proposed algorithm, vertices C and D, which have a heavy communication, are mapped to the same cluster node. However, since the approach in [33] maps each core to NoC architecture one by one in the order of its communication volume, vertices C and H are mapped to the same cluster node. It causes a longer packet latency because the communication volume between vertices C and H is comparatively lower than the communication volume between vertices C and D. As such, our proposed algorithm determines which core is the right one to put into a certain cluster with both its communication volume and locality. Thus, our proposed algorithm is superior to [33] in BMNoC.

Then we have the cluster node CN_1 that includes four vertices as in Step 3 of Fig. 3.4. After that (*Step 3* in Fig. 3.4), we configure the next empty cluster node CN_2, since the first cluster node CN_1 is full of vertices. We connect CN_1 and CN_2 using ES_1.

Going back to *Step*1, we compute how many vertices can be mapped more to the cluster ndoe CN_2. Since CN_2 is empty, we can map four more vertices to CN_2 and then $z = 4$. Since $z > p = 2$, we now consider a growing region that includes just $p = 2$ unmapped vertices (*Step 2* and *Step 2.1* of Fig. 3.5). We try all the possible growing regions composed of two unmapped vertices and pick up the one with highest growth rate. Then we have the cluster node CN_2 that includes two vertices.

Again going back to *Step*1, we compute how many vertices can be mapped more to the cluster node CN_2, since CN_2 already has two vertices, we can map two more vertices to CN_2 and then $z = 2$. As in the first cluster CN_1, we can map two vertices onto CN_2 (*Step 2* and *Step 2.2* of Fig. 3.5).

In the same way, we have the next cluster node CN_3 as shown in Fig. 3.6 and

Fig. 3.7. Then, we finally have a complete BMNoC configuration.

We can summarize our BMNoC configuration algorithm as follows:

Step 1: Configure the first cluster node CN_1, the first edge switch ES_1 and the first mesh router MR_1 which are all empty. Find the two vertices which have the highest communication volume in the task graph $G = (V, E)$ and map them onto CN_1. Let $CN_i = CN_1$, $ES_j = ES1$ and $MR_k = MR_1$. Repeat Steps 1–3 below until all the vertices in G are mapped.

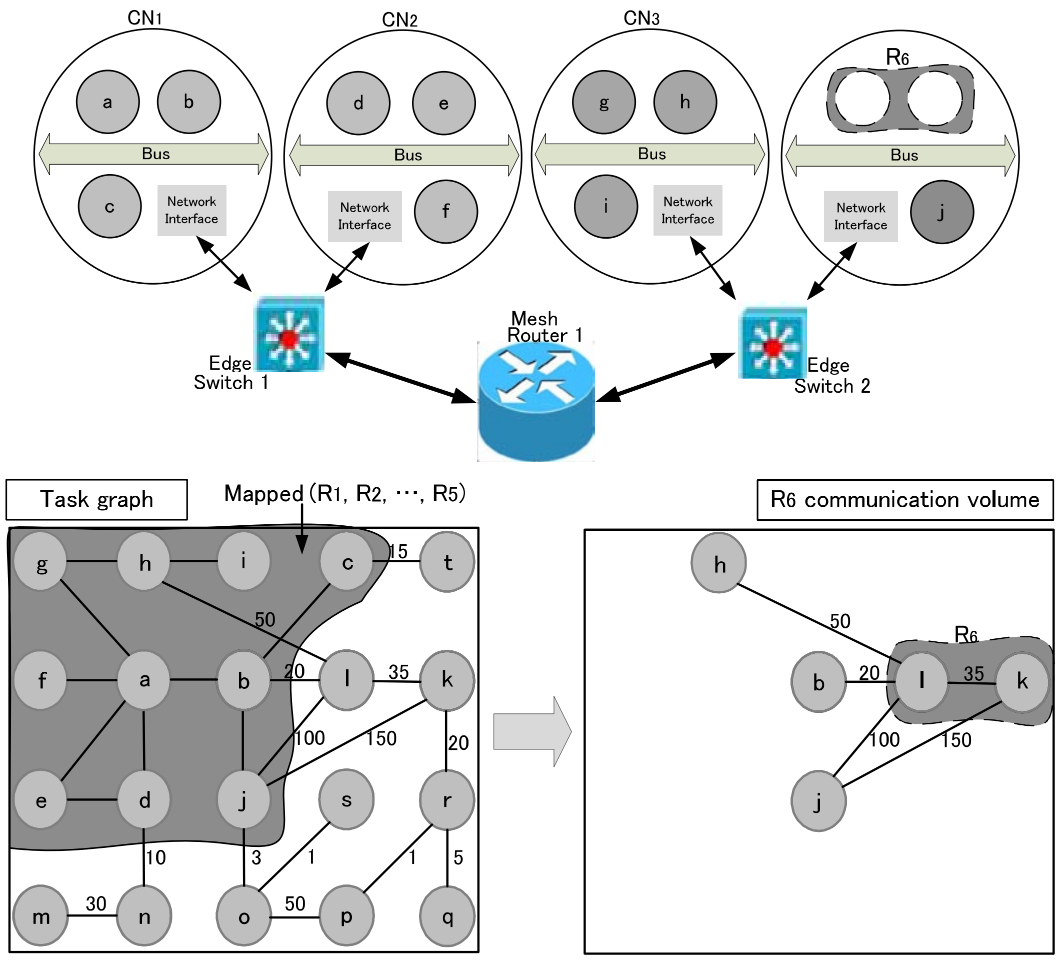
Step 2: Compute how many vertices can be mapped more to the current cluster node CN_i. Assume that z more vertices can be mapped to CN_i.

- **(2.1)** If $p < z$, compare the growth rates of all the possible growing regions composed of p unmapped vertices. Map the vertices in the growing region having the highest growth rates to CN_i.
- **(2.2)** If $p \geq z$, compare the growth rates of all the possible growing regions composed of z unmapped vertices. Map the vertices in the growing region having the highest growth rates to CN_i.

Step 3: Execute the following Steps:

- **(3.1)** If CN_i can have one or more vertices, go to Step 1.
- **(3.2)** Otherwise (CN_i is full), we configure the next empty cluster node CN_{i+1}. $i \leftarrow i + 1$.
- **(3.3)** If the current edge switch ES_j can have one more cluster node, we connect CN_i to ES_j. Go to Step 1.
- **(3.4)** Otherwise (ES_j is full), we configure the next empty edge switch ES_{j+1}. $j \leftarrow j + 1$.
- **(3.5)** If the current mesh router MR_k can have one more edge switch, we connect CN_i to ES_j and ES_j to MR_k. Go to Step 1.

(3.6) Otherwise (MR_k is full), we configure the next mesh router MR_{k+1}. $k \leftarrow k + 1$. MR_k is connected to MR_{k-1} in a reversed-snail order. We connect CN_i to ES_j and ES_j to MR_k. Go to Step 1.



$$GR(R_6) = \begin{array}{l} C1 \times 100\ (c(R_6, j)) + C1 \times 150\ (c(R_6, j)) + C1 \times 35\ (c(R_6, R_6)) + \\ C2 \times 50\ (c(R_6, h)) + \\ C3 \times 20\ (c(R_6, b)) \end{array}$$

Figure 3.3: An example of the growth rate $GR(R_6)$ of R_6.

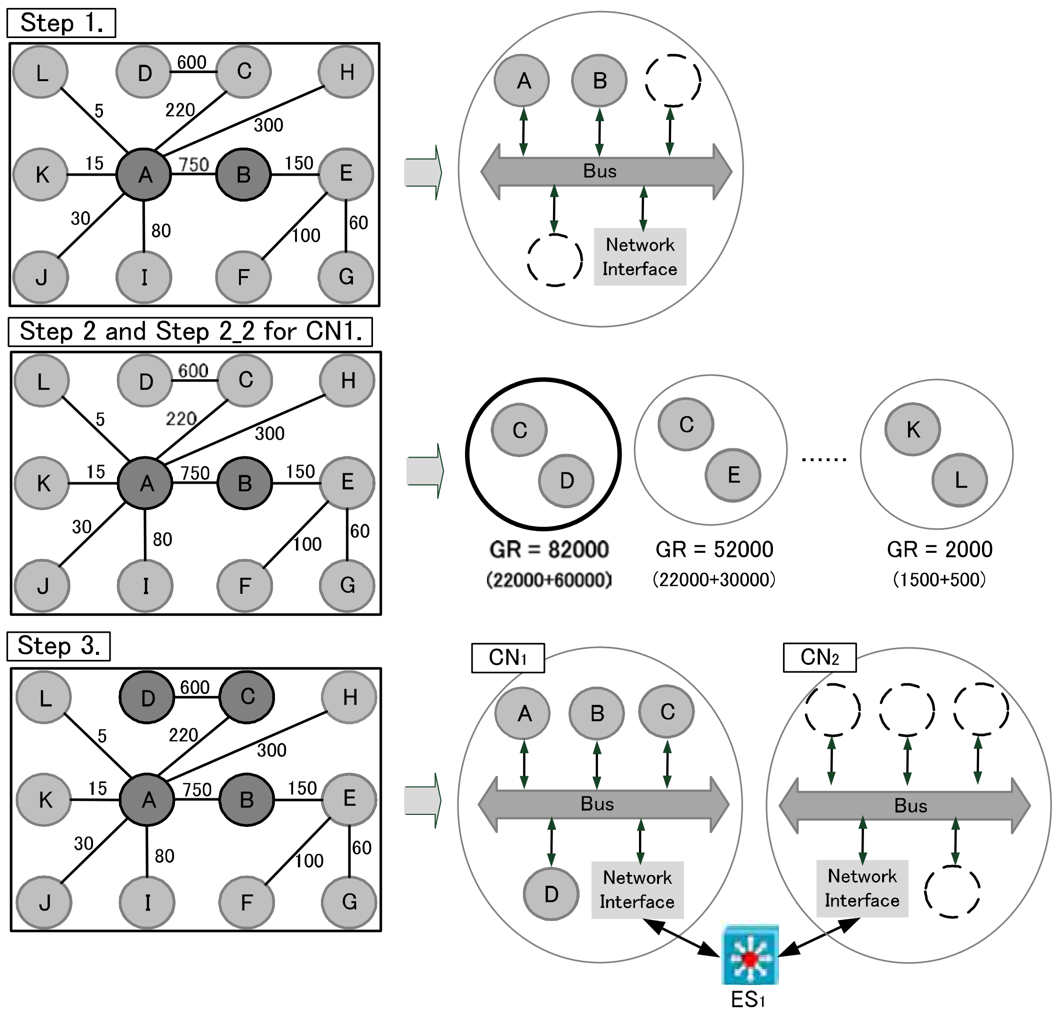


Figure 3.4: Steps 1, 2 and 3 for the cluster node CN_1 ((4, 2, 1)-BMNoC and $p = 2$).

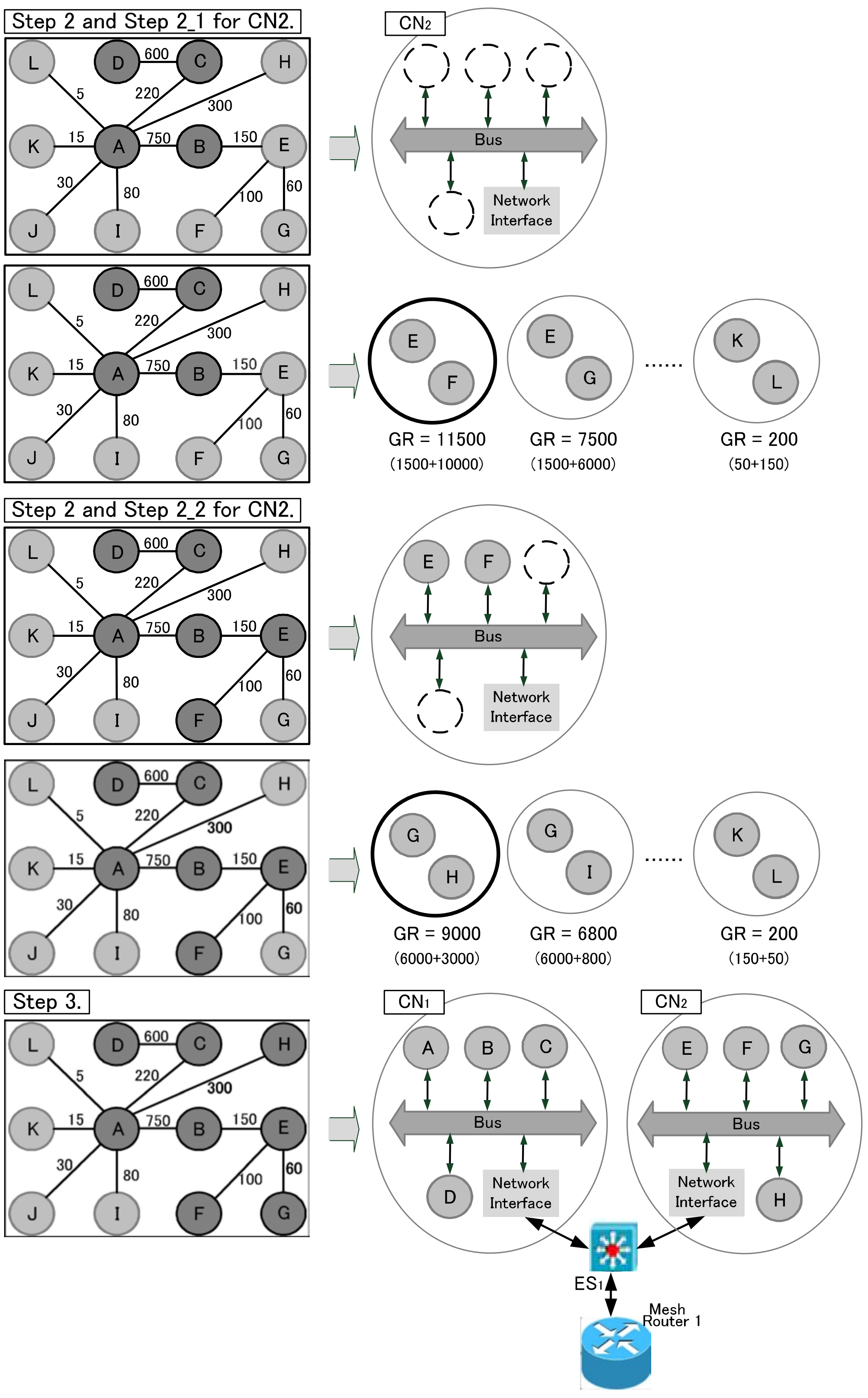


Figure 3.5: Continuing Step 2 for the cluster node CN_2 ((4, 2, 1)-BMNoC and $p = 2$).

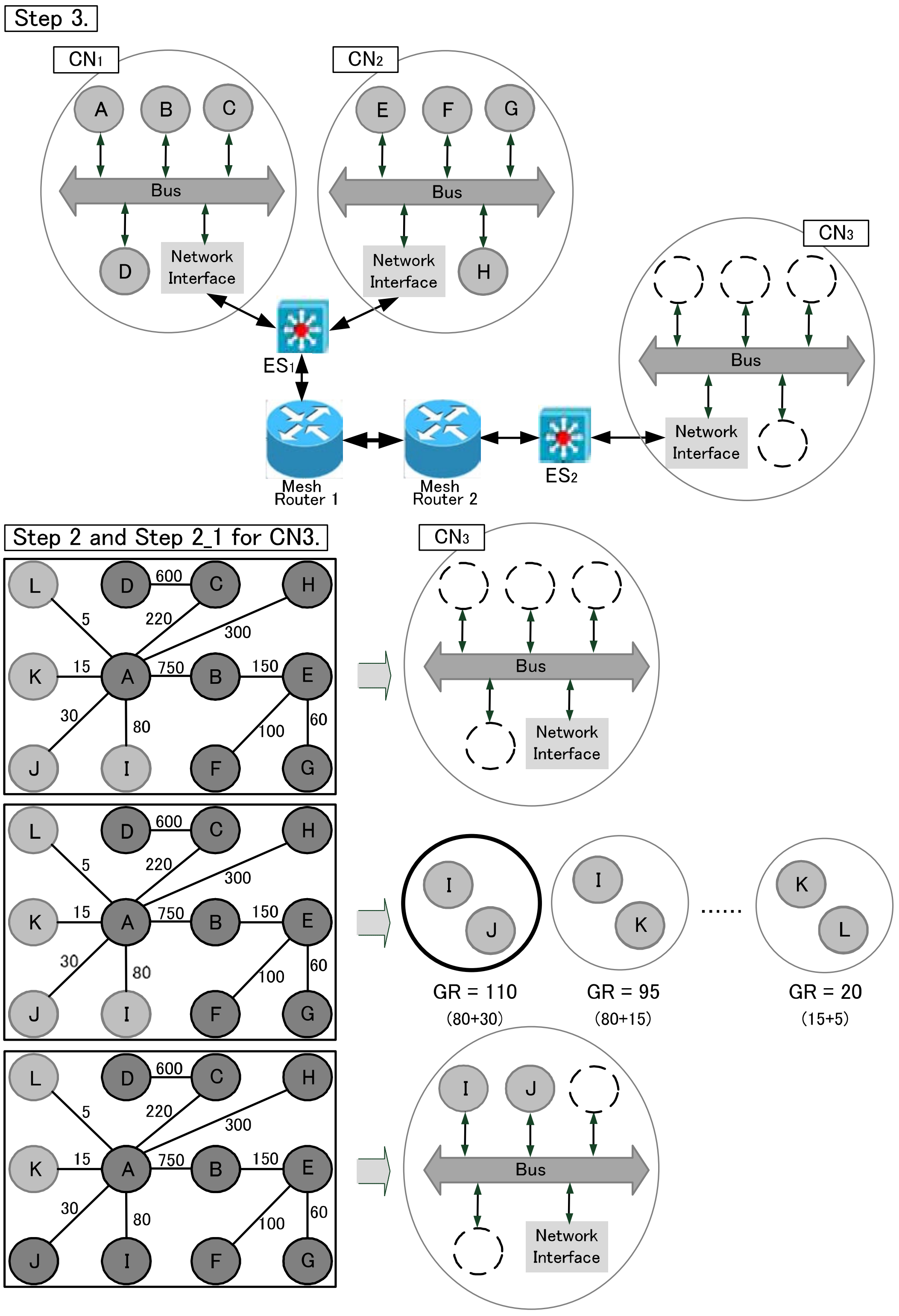


Figure 3.6: Continuing Step 3 for the cluster node CN_2 and Step 2 for the cluster node CN_3 ((4, 2, 1)-BMNoC and $p = 2$).

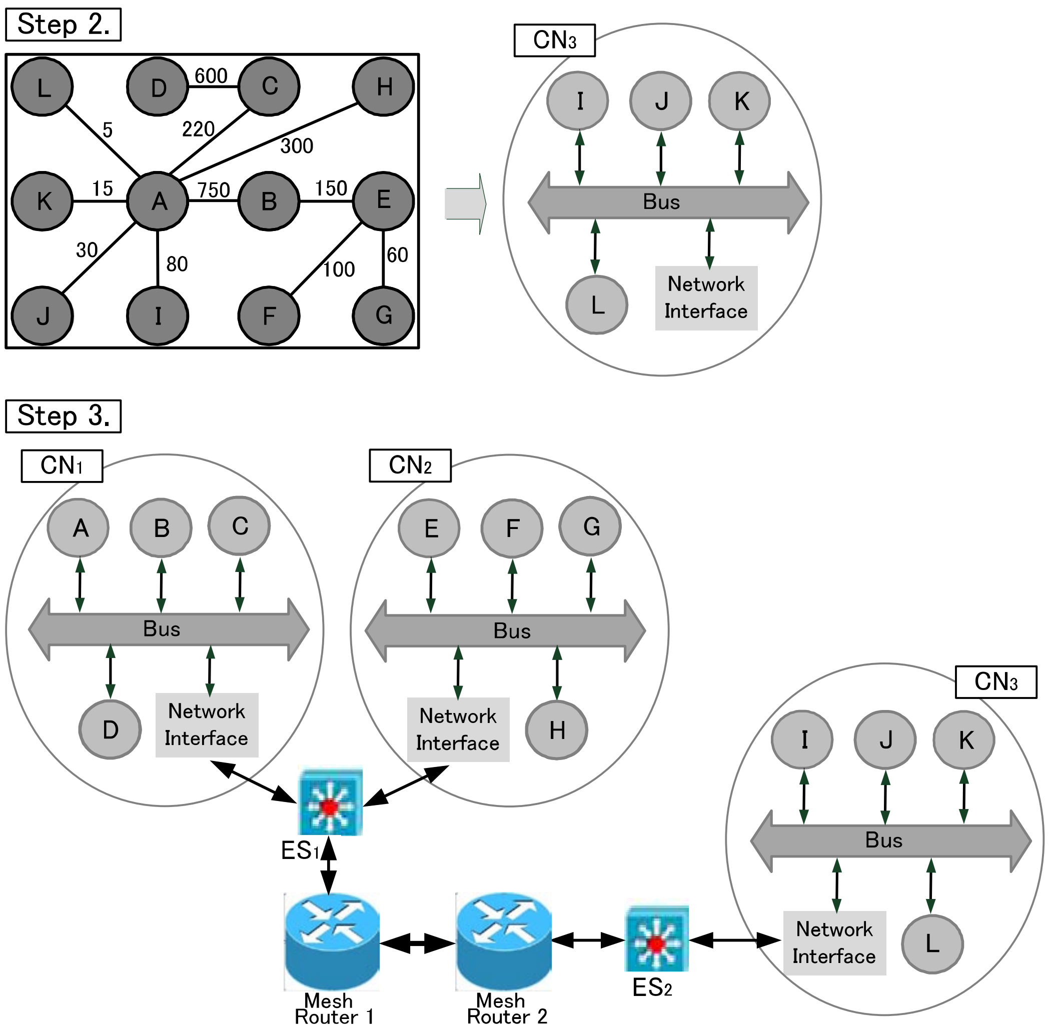


Figure 3.7: Continuing Step 2 for the cluster node CN_3 and final BMNoC configuration ((4, 2, 1)-BMNoC and $p = 2$).

3.4 Experiments

3.4.1 Experiment setup

We have developed a BMNoC configuration algorithm described in the previous section by using the C++ programming language and performed several experiments. In these experiments, we used 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 [9]. We assume each packet consists of 32 flits, each flit is 16 bits long and maximal link bandwidth of 200Mbits/s at 100MHz operation. Our router requires five cycles to route a flit. The buffer size of edge switches and mesh routers is 32 flits. We set up the bus arbitrations consume two cycles from request to address transmission and then one cycle later, data transmission is executed, resulting in three cycles from request to data transmission as in [37, 41]. We set up the routers such as edge switches and mesh routers require four cycles to process a header flit and one cycle to route a packet across the wires of the appropriate port as in [37, 41].

Throughout the experiments, Ns-2 [12] was used for carrying out simulation experiments. Ns-2 is the most typical network simulator which has facilities to describe network topology, network protocols, routing algorithms and communication traffic generation. It provides many mechanisms for modelling traffic generation [1].

3.4.2 Experimental evaluations: Ref. [33] and Our configuration algorithm

We have also applied the two realistic applications; *auto industry* and *telecom*, to the different two BMNoCs; One is configured by our proposed BMNoC configuration algorithm and the other is configured by a breadth-first approach algorithm [33] for comparison purpose. The synthesized BMNoC topology is the same for [33] and both our algorithm but the mappings from task graphs to cluster nodes are different. The CPU time of running our algorithm was 11534 milliseconds and that of running [33] was 9954 milliseconds for *auto industry*. 9934 milliseconds and 8526 milliseconds for *telecom*, respectively.

The variation of average packet latency as a function of traffic injection rates for *auto industry* is given in Fig. 3.8. These plots show that our proposed algorithm shifts the critical traffic load from 0.30 to 0.38 (approximately 22% improvement) as compared to the one configured by a breadth-first approach algorithm [33]. Similarly, the average packet latency for our BMNoC is consistently smaller as compared to [33]. For instance, at 0.29 packet injection rate, the latency drops from 100 to 54 cycles, giving about 46% reduction.

Similar improvements have been observed for *telecom* as shown in Fig. 3.9. Specifically, the critical traffic load is improved from 0.48 to 0.63 showing approximately 24% improvement as compared to the counterpart [33]. Likewise, the latency at 0.47 traffic injection rate drops from 110 to 65 cycles giving approximately 41% reduction.

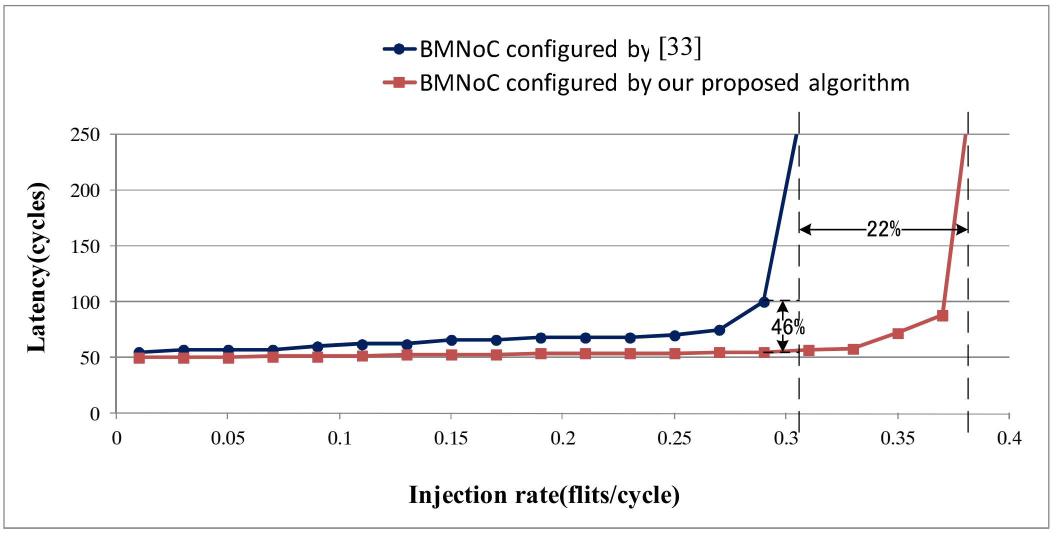


Figure 3.8: Traffic injection rate versus average packet latency for *auto industry* benchmark.

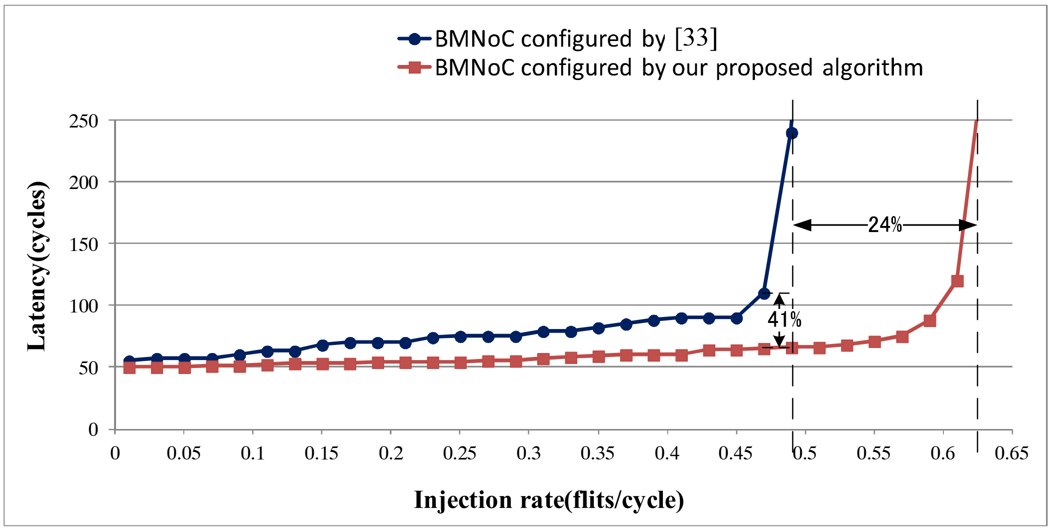


Figure 3.9: Traffic injection rate versus average packet latency for *telecom* benchmark.

3.5 Concluding remarks

In this Chapter, we proposed a novel BMNoC configuration algorithm with a target architecture, BMNoC. Our proposed BMNoC algorithm not only utilizes the communication volume between cores but also makes aware the localities of cores and BMNoC architectures. It reduces network traffic and latency by mapping the cores, which have the heavy communication with each other, into the same *local* cluster node and connecting them with a local bus. Therefore, our BMNoC configuration algorithm reduces the traffic load at the center of the network by a hierarchical communication network. Experimental results using the two realistic applications verified that BMNoC configured by the proposed algorithm can significantly improve the critical traffic load and the average packet latency.

Chapter 4

BMNoC utilizing Packet Transmission Priority Control Method

In this chapter, we propose a BMNoC configuration algorithm utilizing packet transmission priority control method. Performance of many-core chip multiprocessors (CMPs) is significantly influenced by performance of NoC which interconnects each core for on-chip communication. Specially, latency of packets in NoC remarkably effect performance of applications executing on a many-core CMP. Because it is projected that executing different applications concurrently on a many-core CMP becomes highly required in the future, it will be important that a priority control which guarantees or differentiates latency of packets. Our proposed BM-NoC configuration algorithm utilizing packet transmission priority control method minimizes the average packet latency by improving the efficiency of the buffers. Packet transmission priority control methods give transmission priority to a port, in cases of that a payload flit is blocked in the port during a header and its subsequent flits have passed through a crossbar. A new packet transmission priority control method does not need additional buffers since it just gives priority to the transmission permission issue of arbiters.

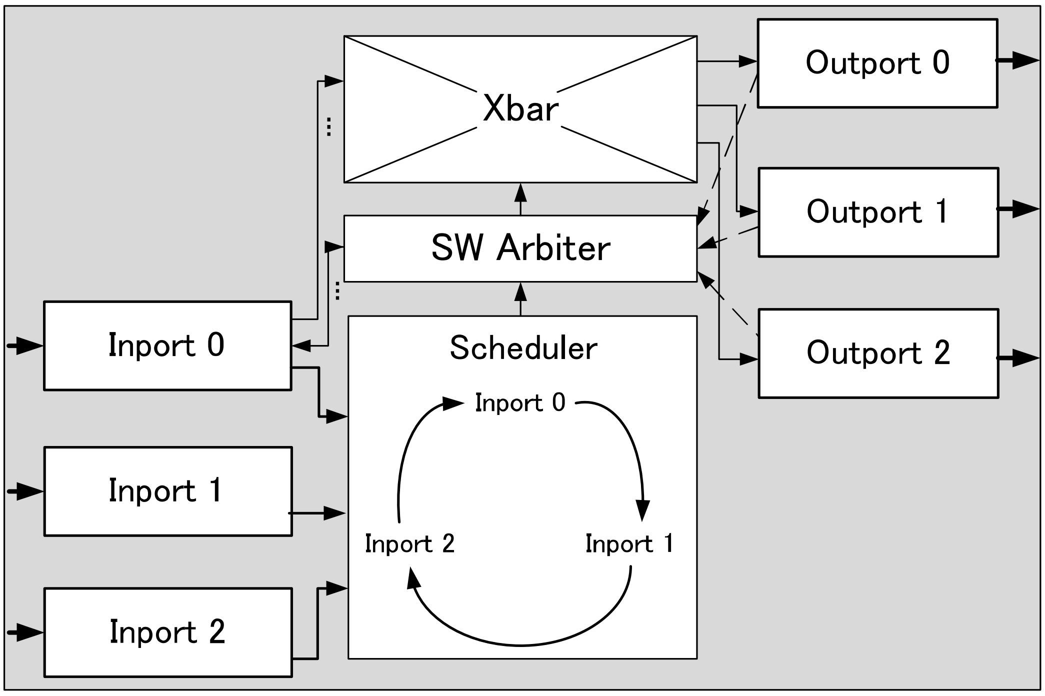


Figure 4.1: A round-robin router architecture.

4.1 Packet Transmission Priority Control Method

A round-robin arbiter is used to resolve conflicting requests generated from various sources for a shared resource in a directional and cyclic order for wormhole routings. Round-robin describes a method of choosing a resource for a task from a list of available ones, usually for the purposes of load balancing. As the basic algorithm, the scheduler selects a resource pointed to by a counter from a list, after which the counter is incremented and if the end is reached, returned to the beginning of the list as shown in Fig. 4.1. However, Round-Robin policy causes large delay for a wormhole switching on parallel computers in cases of stuck network pipelines as shown in Fig. 4.2. New packet transmission priority control methods that improve average delay by improving the efficiency of the buffers are proposed for NoCs.

The New packet transmission priority control method gives transmission priority to a port, in cases of that a payload flit is blocked in the port during a header

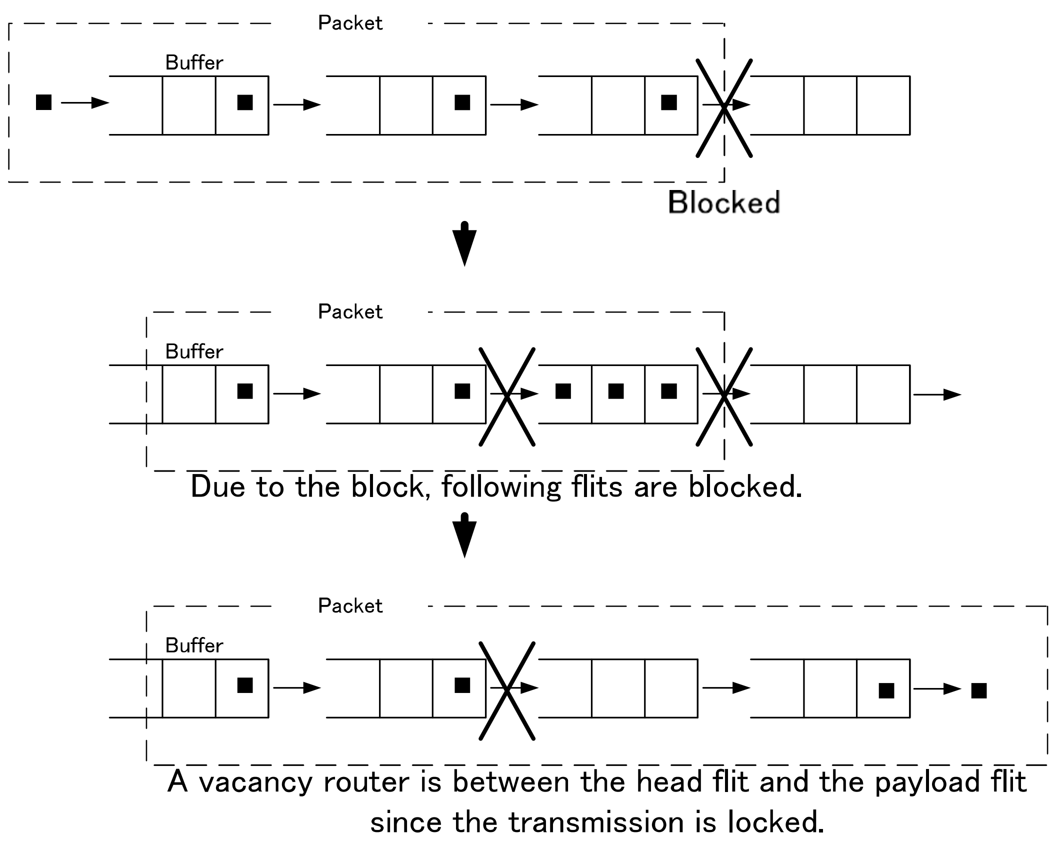

Packet
Buffer
Blocked
Packet
Buffer
Due to the block, following flits are blocked.
Packet
Buffer
A vacancy router is between the head flit and the payload flit since the transmission is locked.

Figure 4.2: In case of a blocked transmission.

and its subsequent flits have passed through a crossbar. The method does not need additional buffers since it just gives priority to the transmission permission issue of arbiters. We can summarize the new packet transmission priority control method as follows and shown in Fig. 4.3:

- If the transmission is blocked, give transmission priority to the blocked port which is revoked crossbar permission.
- If the next buffer is not full, the blocked port takes transmission priority and if the buffer is full, return to the round-robin method.
- When the end flit is reached to the destination, the transmission priority is revoked and return to round-robin method.

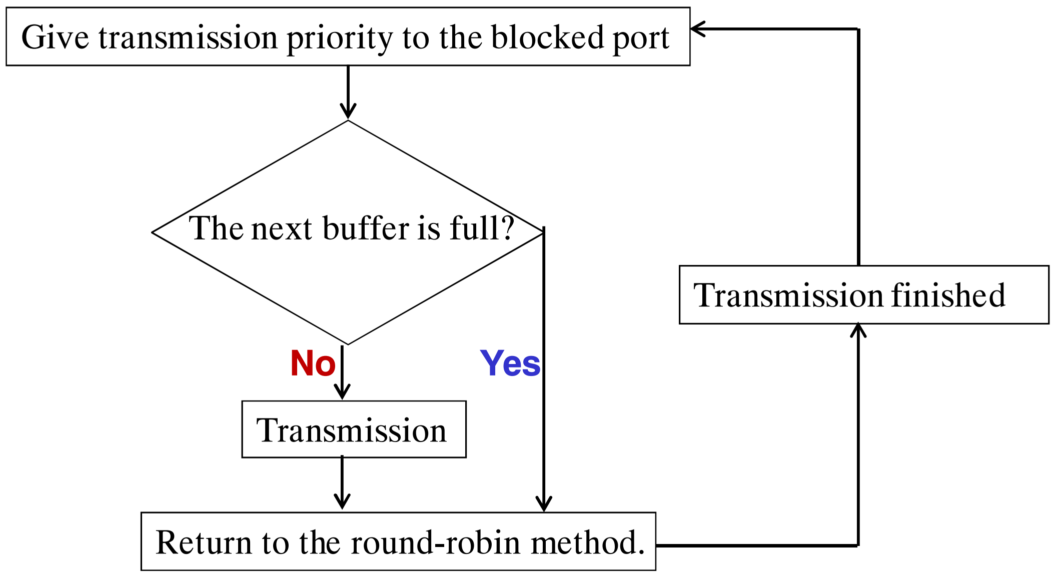

Give transmission priority to the blocked port
The next buffer is full?
No
Yes
Transmission
Return to the round-robin method.
Transmission finished

Figure 4.3: New packet transmission priority control method.

If 2 more ports have obtained transmission priority, it again gives them transmission priority by round-robin. Thus, the method can improve average delay by improving the efficiency of the buffers are proposed for NoCs. An example of transmission permission issue of arbiter is shown in Fig. 4.4.

Although port 2 starts transmission at some point, the transmission is blocked since the buffer of destination is full. In round-robin, ports obtain transmission permission in order of a list evenly, whereas in the new packet transmission priority control method, port 2 which has been blocked transmission has transmission permission first when the buffer of destination is available.

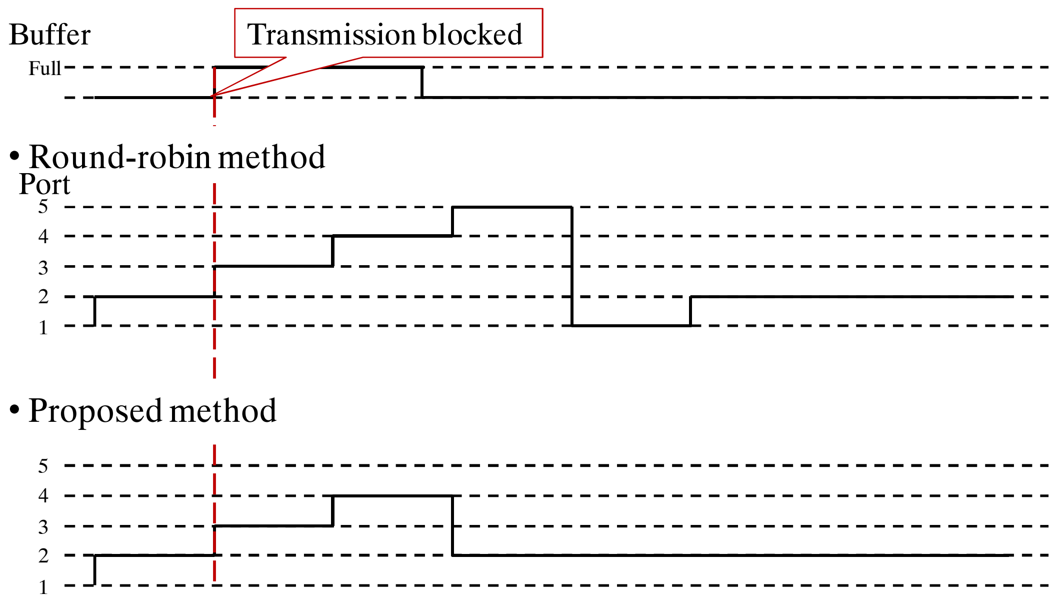


Figure 4.4: An example of transmission permission issue of arbiter.

4.2 BMNoC utilizing Packet Transmission Priority Control Method

In this section, we propose a novel busmesh NoC utilizing packet transmission priority control method. The basic idea of BMNoC is to develop an NoC architecture with clusters which are connected by mesh network, borrowing the hierarchical model from the Internet and adapting it to communication networks. Our proposed BMNoC is a generalized and simplified version of BMNoC as shown in Fig. 4.5. A novel BMNoC is composed of cluster nodes (CNs) and mesh routers (MRs). In intra-cluster node, several cores which have a heavy communication to each other are connected by a local bus. It can provide the better performance in terms of the latency and the throughput since local buses transmit data directly to other cores in the same cluster node with a parallel fashion, which eliminates packetizing overhead. Furthermore, our proposed novel BMNoC is applied packet transmission priority control method in [43] that minimizes the average packet latency by improving the efficiency of the buffers. Packet transmission priority control methods give transmission priority to a port, in cases of that a payload flit is blocked in the port during a header and its subsequent flits have passed through a crossbar. A new packet transmission priority control method does not need additional buffers since it just gives priority to the transmission permission issue of arbiters. Fig. 4.6 shows the architecture of router for our proposed BMNoC utilizing packet transmission priority control method.

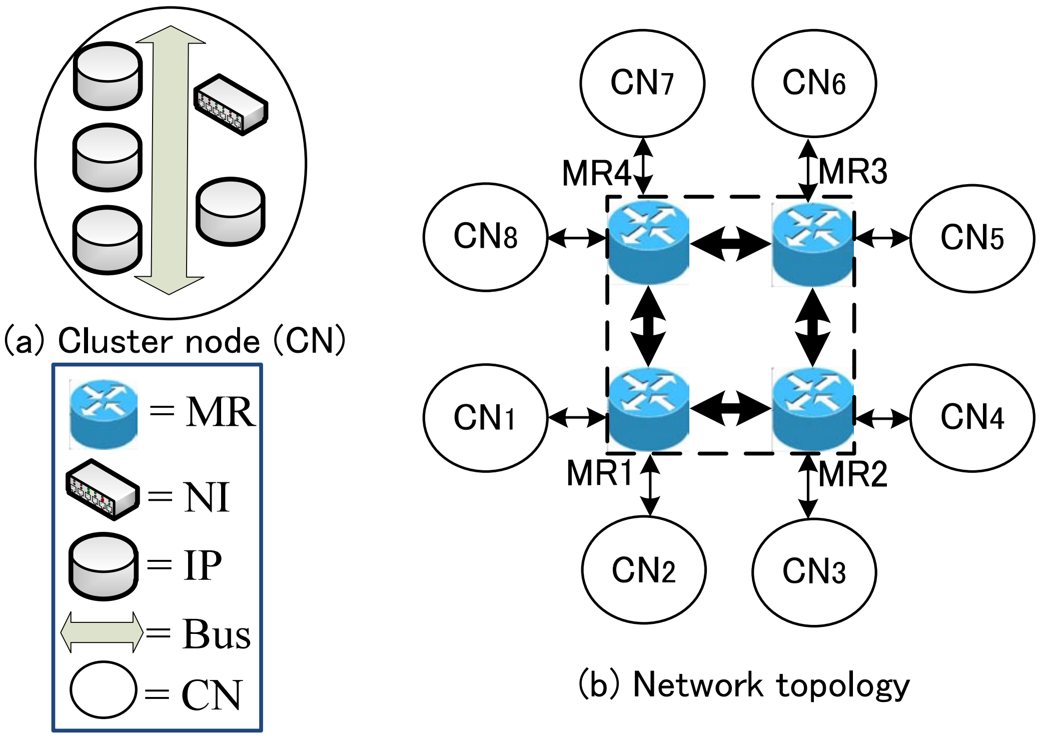


Figure 4.5: A general architecture of novel BMNoC.

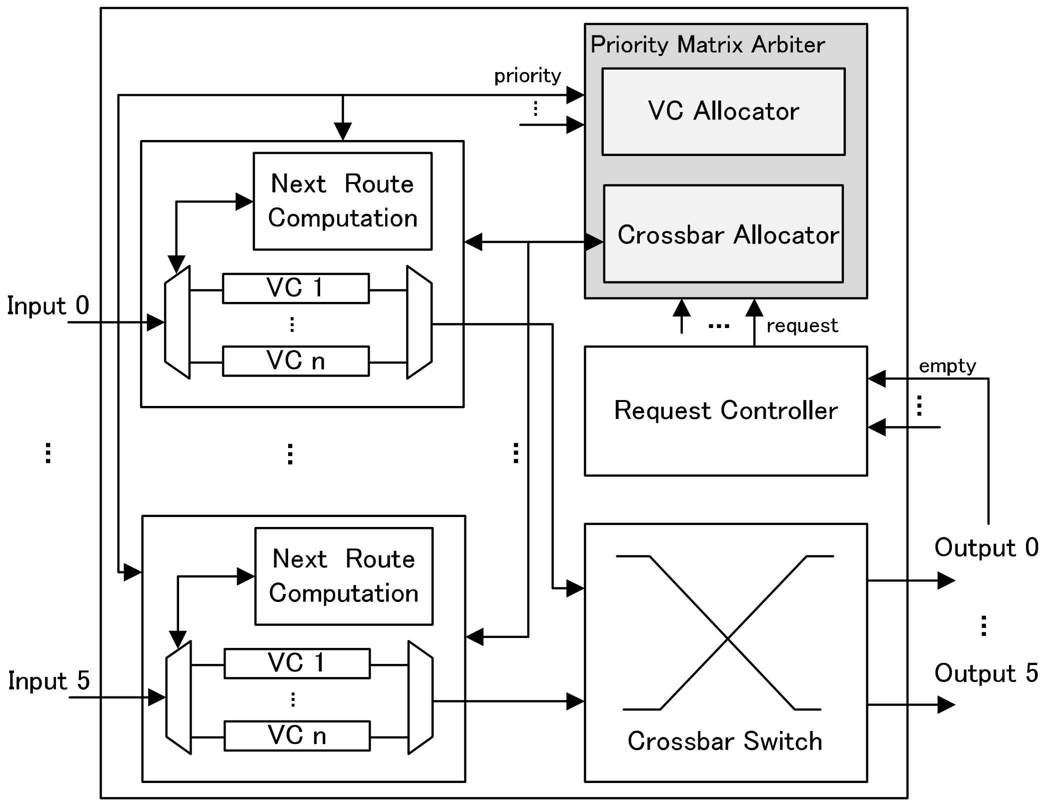


Figure 4.6: A router architecture of novel BMNoC.

4.3 Experiments

4.3.1 Experiment setup

In these experiments, we used 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. We assume each packet consists of 8 flits, each flit is 34bits long and maximal link bandwidth of 200Mbits/s at 100MHz operation. Our router requires five cycles to route a flit. The buffer size of edge switches and mesh routers is 8 flits. We set up the bus arbitrations consume two cycles from request to address transmission and then one cycle later, data transmission is executed, resulting in three cycles from request to data transmission as in [37, 41]. We set up the routers such as edge switches and mesh routers require four cycles to process a header flit and one cycle to route a packet across the wires of the appropriate port as in [37, 41].

Throughout the experiments, Ns-2 [12] was used for carrying out simulation experiments. Ns-2 is the most typical network simulator which has facilities to describe network topology, network protocols, routing algorithms and communication traffic generation. It provides many mechanisms for modelling traffic generation.

4.3.2 Object detection systems

In experiments, we take object dection systems as a target application. Object detection is a key capability for applications in robotics, surveillance, or automated personal assistance. The main challenge is the amount of variations in visual appearance owing to clothing, articulation, cluttering backgrounds and illumination conditions particularly in outdoor scenes. A number of different approaches for detecting object in images using some feature representations and learning methods have been proposed in the literatures. The use of orientation histograms has been extensively used in [8, 28, 30, 48]. The Histograms of Oriented Gradients *HOG* features have provided excellent performance contrast to other existing edge- and gradient-based features by Dalal and Triggs [8]. To overcome the affects of geometric and rotational variations, the system automatically assigns the dominant orientations of each block-based feature encoding by using the rectangular- and circular-type histograms of orientated gradients (HOG), which are insensitive to various lightings and noises at the outdoor environment.

The Adaboost approach has established itself as a powerful learning algorithm that can be used for feature selection [46]. The Adaboost approach selects a small set of discriminative HOG features, which well suited for human detection by constructing a cascade-of-rejecter system. The Adaboost algorithm [46] is used to select a small number of weighted *HOG* features, i.e. weak classifiers, to integrate into a strong classifier. Each weak classification is selected by evaluating training datasets of positive and negative, each classifier showing the lowest error is chosen. A powerful feature selection algorithm, Adaboost, is performed to automatically select a small set of discriminative HOG features with orientation information in order to achieve robust detection results.

The object detection system is shown in Fig. 4.7.

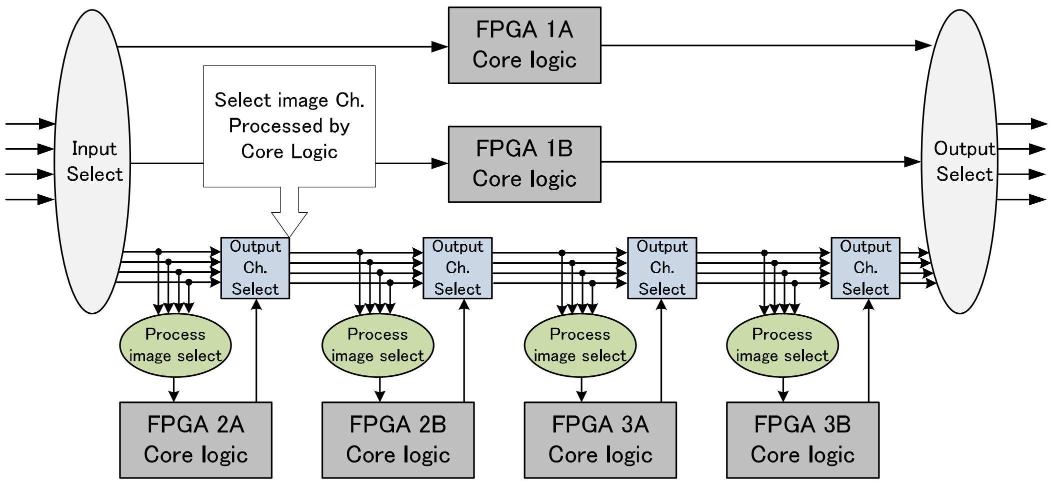


Figure 4.7: NTT object detection system [2].

Table 4.1: The comparison between HNoC and BMNoC.

object dection system			
Architectures	**# of routers**	**# of CNs**	**# of MRs**
HNoC	72	–	–
BMNoC	–	18	9

4.3.3 Experimental evaluations: HNoC and our BMNoC

Here, we evaluate the performance of BMNoC using two realistic applications, *telecom* and *object detection systems* , and compare it with a conventional NoC such as a hybrid NoC (HNoC) [49]. A *telecom* application is retrieved from E3S benchmark suite [10] where *telecom* has 32 vertices. A *telecom* , is mapped onto $(4, 2)$-BMNoC where 4 cores are put into a CN and 2 CNs are connected to a MR as in [25, 45]. The *object detection systems* application is retrieved from NTT Microsystem Integration Lab. The object detection system using 6 pairs of feature extraction and image recognition algorithms [2] which has 72 vertices. The task graph of *object detection systems* is shown in Fig. 4.8. A *object detection system* is also mapped onto $(4, 2)$-BMNoC using our proposed BMNoC configuration algorithm as in [25, 45].

We have applied the applications to HNoC, BMNoC, HNoC utilizing packet transmission priority control method (HNoC+PTPCM) and BMNoC utilizing packet transmission priority control method (BMNoC+PTPCM). The result is shown in Table 4.1

As in [14], the variation of *average packet latency* as a function of traffic injection rates for *telecom* is given in Fig. 4.9. Average packet latency equals the average cycles taken by a packet to go through a communication path from its source to its intended sink. For example, when we have an edge $e = (u, v)$ and its communication volume $c(e)$ in a given task graph, $c(e)$ packets go through its communication path from u to v and we can obtain its required cycles $t(e)$. Then we obatin its average cycle by calculating $t(e)/c(e)$. We calculate the average cycles for all the edges in a given task graph and then finally have an average packet latency by averaging them.

These plots show that our proposed BMNoC utilizing packet transmission priority control method improves the critical traffic load by approximately 20% as compared to HNoC and approximately 15% as compared to HNoC+PTPCM. Furthermore, BMNoC+PTPCM improves the critical traffic load as compared to conventional BMNoC (approximately 6% improvement). Similarly, the average packet latency for BMNoC+PTPCM is consistently smaller as compared to HNoC, HNoC+PTPCM and conventional BMNoC.

Regulating the communication volumes of cores hierarchically is very important not to congest the network with generated packets. At low traffic loads, the average packet latency exhibits a weak dependence on the traffic injection rate. However, when the traffic injection rate exceeds a critical traffic load, the packet delivery cycles rise abruptly and the network throughput starts collapsing.

Similar improvements have been observed for *object detection system* as shown in Fig. 4.10. Specifically, in BMNoC+PTPCM, the critical traffic load is improved by approximately 22% as compared to HNoC and approximately 13% improvement as compared to HNoC+PTPCM. Furthermore, BMNoC+PTPCM improves the critical traffic load as compared to conventional BMNoC (approximately 7% improvement). Likewise, the average packet latency for BMNoC+PTPCM is consistently smaller as compared to HNoC, HNoC+PTPCM and conventional BMNoC.

Also, we implemented BMNoC, BMNoC+PTPCM and HNoC architectures in order to evaluate the difference of areas including architectural components such as routers, network interfaces and buses (we do not consider the area of IP cores in this experiment). We consider the area for the bus architecture for area evaluation as in [23]. The architectures were synthesized in a Vertex-4 xc4vlx15-12sf363 by using Xilinx ISE Design Suite 11. Area results, presented in Fig. 4.11, show that BMNoC can save up to 70% of the number of LCs as compared to HNoC. Although BMNoC+PTPCM has the imperceptible larger areas as compared to BMNoC, it can also save up to 65% of the number of LCs as compared to HNoC.

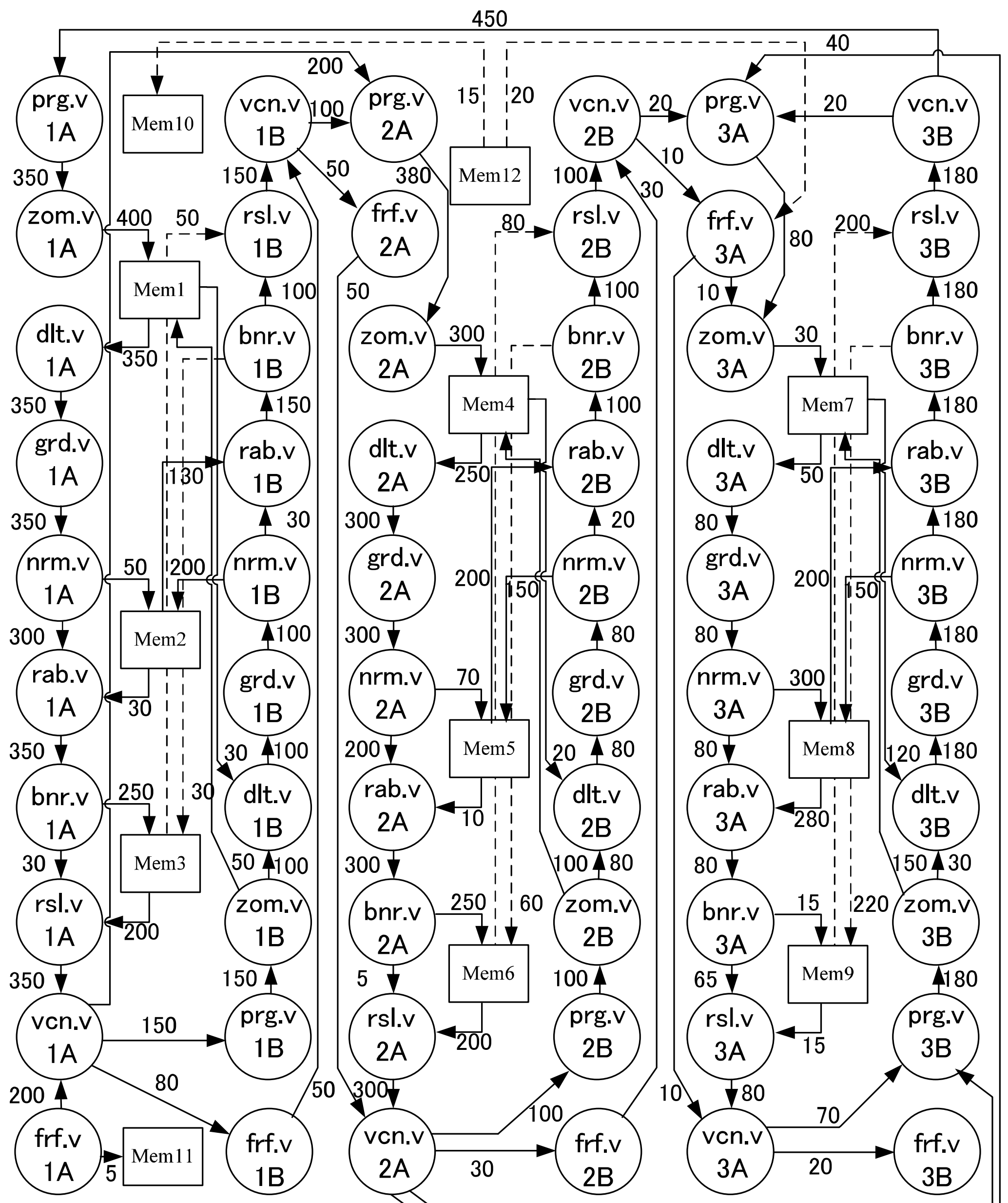


Figure 4.8: The task graph of *object detection systems*.

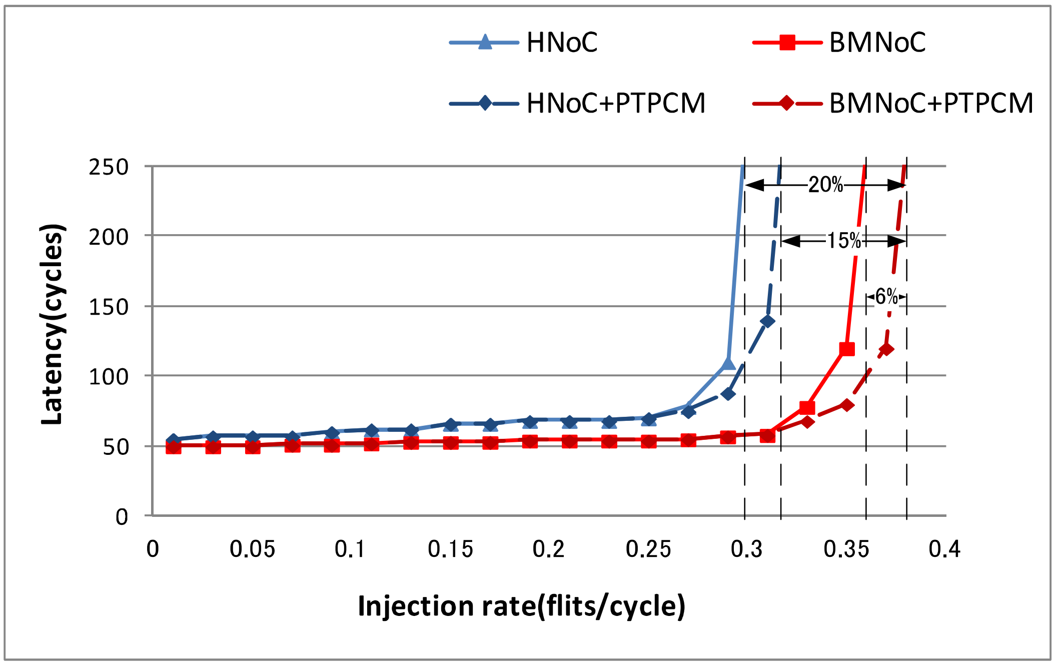


Figure 4.9: Traffic injection rate versus average packet latency for *telecom* benchmark.

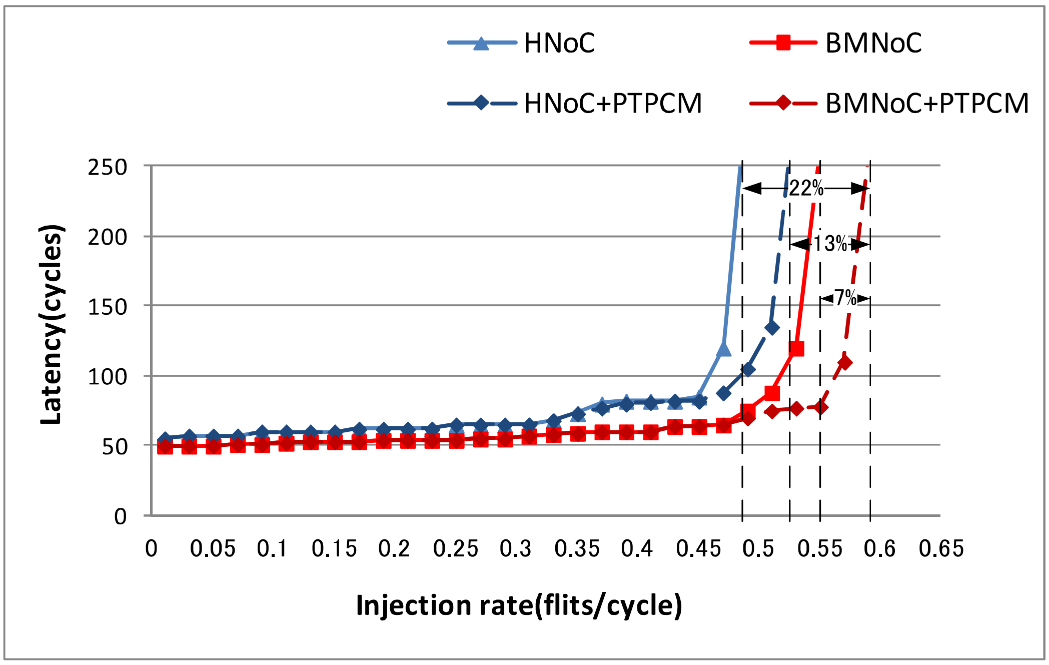


Figure 4.10: Traffic injection rate versus average packet latency for *object detection system* benchmark.

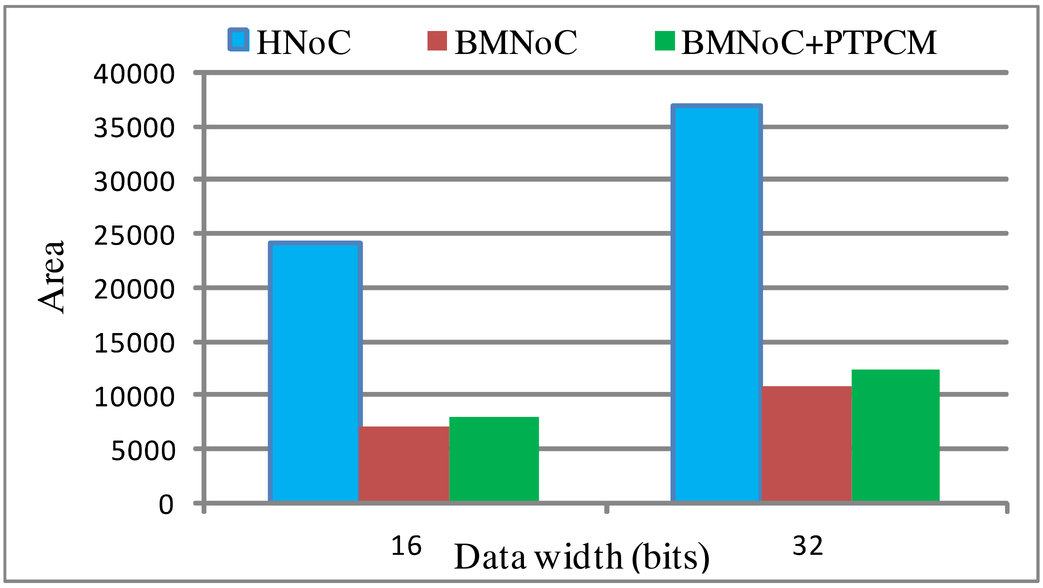


Figure 4.11: # LCs for BMNoC, BMNoC+PTPCM and HNoC architectures.

4.4 Concluding remarks

In this Chapter, we proposed a novel BMNoC utilizing packet transmission priority control method with synthesis results using Vertex-4 FPGA. It reduces network traffic and latency by mapping the cores, which have the heavy communication with each other, into the same local cluster node and connecting them with a local bus. Therefore, our proposed BMNoC reduces the average delay on the network by improving the efficiency of the buffers with a packet transmission priority control method. Experimental results verified that our BMNoC can significantly improve the critical traffic load and the average packet latency is consistently smaller as compared to conventional NoC such as HNoC [49]. Furthermore, BMNoC and BMNoC+PTPCM can save the number of LCs as compared to HNoC.

Chapter 5

Related Works

In this chapter, we introduce related works. Microprocessor performance has increased exponentially over the last four decades as advancing semiconductor technology has vastly increased the quantity and improved the speed of on-chip transistors available to circuit designers. Traditionally, computer designers took advantage of these resources to improve uniprocessor structures because of its simpler programming model compared to systems with distributed structures [29]. However, power consumption and wire delay have recently limited the continued scaling of uniprocessor systems making chip multiprocessor architectures more appealing [31]. In addition, network-on-chip (NoC) has become the emerging paradigm for communication within large chip multiprocessor systems to overcome scalability, power, delay, and other issues with global interconnects.

5.1 Architectures

According to the multi-core chip evolution, which forecasts tens and hundreds of cores in few years, logical and physical clusters of cores have been considered as an approach to support parallel processing. The next phase for this evolution can be called on-chip distributed computing, which consists of distributed clusters of cores to process a large number of different workloads [11, 15, 16, 42]. A cluster-based NoC (C-NoC) a modified model of Hermes NoC (H-NoC) [31] is introduced by Seifi and Eshghi [42]. Every H-NoC switch is attached to one core. Each core needs to communicate with other cores. Therefore, they must send packets to other switches and the delay is increased. To solve this problem, the C-NoC switch has four local ports and configures a cluster as shown in Fig. 1.1(a). The latency of C-NoC [42] is decreased by 15.1% compared with the conventional NoC, SoCIN [50]. However, its performance is still insufficient for heterogeneous cores and clusters.

Among numerous NoC topologies, mesh is a popular one due to the simplicity and regularity. In mesh topology, each router is connected to several local cores and adjacent routers. Since the router only connects to its neighboring routers, the data packets transmitted from the source core to the destination core may travel long distance, and affects the performance of whole SoC [7]. Besides, with increasing size of NoC, the mesh topology has its disadvantage in the communication latency and the concentration of the traffic in the center of the mesh topology (named hot-spot). Although some studies have been made on effective core mapping methodologies to solve the hot-spot problem [18, 33, 34], what seems to be lacking is a better on-chip communication architecture. To solve the problems of the mesh topology for future SoC design, many research papers have proposed a great number of approaches in recent years and a hybrid bus-NoC architecture is one of them [5, 26, 45, 49]. A hybrid bus-NoC architecture is a system platform which is based on a standard NoC architecture, and contains several clusters which are composed of local buses. In the hybrid bus-NoC architecture, cores with heavy traffic and communication volume are placed in the same cluster with a local bus to avoid hot-spots and reduce the transmission latency. Since the hybrid architecture is based on a standard mesh NoC concept, the router of the hybrid system not only connects with its neighboring routers but also connects to a cluster which is composed of several cores and a local bus. It is noteworthy that new interface is not needed for each core in subsystem, and it can further reduce the design cost.

5.2 Algorithms

The configuration problem for NoCs has been first addressed by Hu and Marculescu [21], where a branch and bound algorithm is proposed to map a given set of cores onto a regular NoC architecture such that the total communication energy is minimized. At the same time, the performance of the resulting communication system is guaranteed to satisfy the specified design constraints through bandwidth reservation. Murali and De Micheli [33] propose a configuration algorithm for NoC architectures which supports traffic splitting. Srinivasan and Chatha [44] present a configuration algorithm to minimize the communication energy subject to bandwidth and latency constraints. A multi-objective configuration algorithm for mesh-based NoC architectures is presented by Ascia, Catania, and Palesi [3]. This approach finds the Pareto mappings that optimize performance and power consumption using evolutionary computing techniques. Improving upon these studies, Hansson, Goossens and Radulescu [17] propose a more general unified approach for application mapping and routing-path selection which considers both of best effort and guaranteed service traffic.

It is very important for a configuration algorithm for hybrid bus-NoC architectures to decide which cores should be assigned to the same cluster and map the clusters onto the network that the transmission latency is minimized and the locality is contemplated. When partitioning cores, the cores with heavy traffic and communication volume should be assigned to the same cluster and placed as close as possible to reduce the transmission latency.

We note that many mapping algorithms use, directly or indirectly, the average packet hop count as a cost function by relating the average number of packet hops to the communication energy consumption [22] or communication cost [33]. However, these algorithms [3, 17, 21, 22, 27, 33, 36, 44] cannot be applied to a hybrid bus-NoC architectures since they consider only communication rate between cores and do not take into account the *locality* that hybrid bus-NoC architectures have.

In [27], several cores, which have heavy communication volume among them, are connected to the same router and the routers are connected to the upper level routers. This idea is rather similar to C-NoC [42] which has a specific switch having 8 bidirectional ports: 4 local ports and 4 links (North, South, East, and West). Although it is similar to our proposed algorithm in terms of using a hierarchical architecture, [27] does not use any clusters and subsystems whereas cluster nodes, edge switches and mesh routers are used for a hybrid and hierarchical structure in our proposed algorithm. In [45], a hybrid NoC architecture which is composed of routers and subsystems is proposed. Although it is similar to our proposed algorithm in terms of using a hybrid architecture, [45] does not partition cores into a hierarchical structure which is composed of clusters and routers differently from our proposed algorithm. The architecture and algorithm in [45] are not generalized so it is difficult to apply it to many applications.

Furthermore, the basic idea of mapping algorithm in [27, 45] are the same as [33]. The approach in [27, 45] and [33] maps each core to NoC architecture one by one, in breadth-first-search manner, only in the order of its communication volume after picking up the two cores which are connected by the highest communication volume in an input task graph. Although it may cause a poor result such as a longer latency and a low throughput since the greedy approach does not contemplate the localities between cores, the algorithm in [33] is used for a hybrid bus-NoC architecture as a makeshift now.

5.3 Concluding remarks

In this chapter, we introduce related works. First of all, we introduce conventional NoC architectures such as C-NoC and HNoC. After that, we decribe and define our target architecture, which is called a busmesh NoC (BMNoC). BMNoC is a generalized version of hybrid bus-NoC architectures. Furthermore, we introduce conventional NoC configuration algorithms and a breadth-first approach algorithm which is used for a hybrid bus-NoC architecture as a makeshift now. Unfortunately, the conventional NoC configuration algorithms are not suitable for a hybrid bus-NoC architecture since they are only greedy of communication volume or bandwidth requirement. Breadth-first approach algorithm maps each core to NoC architecture one by one, in *breadth-first-search manner*, only in the order of its communication volume after picking up the two cores which are connected by the highest communication volume in an input task graph. It may cause a poor result such as a longer latency and a low throughput since the greedy approach does not contemplate the localities between cores. Thus, we propose a novel configuration algorithm for a hybrid bus-NoC architecture.

Chapter 6

Conclusion

In this dissertation, we described and defined our target architecture, which is called a busmesh NoC (BMNoC). BMNoC is a generalized version of hybrid bus-NoC architectures. After that, we proposed a novel BMNoC configuration algorithm with a target architecture, BMNoC. Our proposed BMNoC algorithm not only utilizes the communication volume between cores but also makes aware the localities of cores and BMNoC architectures. It reduces network traffic and latency by mapping the cores, which have the heavy communication with each other, into the same *local* cluster node and connecting them with a local bus. Therefore, our BMNoC configuration algorithm reduces the traffic load at the center of the network by a hierarchical communication network. Our target architecture is called BMNoC which is based on a hierarchical model from the Internet and it adapts it to communication networks. It is composed of bus-based connection and global mesh routers to enhance the performance of on-chip communication. Experimental results using the two realistic applications verified that BMNoC configured by the proposed algorithm can significantly improve the critical traffic load and the average packet latency for BMNoC is consistently smaller as compared to conventional NoCs such as C-NoC [42] and HNoC [49]. Moreover, our BMNoC has more reduced area overhead as compared with conventional NoCs. We also proposed a novel BMNoC utilizing packet transmission priority control method with synthesis results using Vertex-4 FPGA. It reduces network traffic and latency by mapping the cores, which have the heavy communication with each other, into the same local cluster node and connecting them with a local bus. Therefore, our proposed BMNoC untilizing packet transmission priority control method reduces the average delay on the network by improving the efficiency of the buffers with a packet transmission priority control method. Experimental results verified that our BMNoC can significantly improve the critical traffic load and the average packet latency is consistently smaller as compared to conventional NoC such as HNoC [49]. Furthermore, BMNoC and BMNoC+PTPCM can save the number of LCs as compared to HNoC.

For future implementations of BMNoC, we contemplate a floorplan for reducing the latency caused by wire-length. Furthermore, we want to define a generic tile interface so that BMNoC can be embedded in a multi-tile SoC. It will support several types of communication and application that can be used by the designers.

Acknowledgement

I would like to express my profound gratitude and appreciation to my advisor, Prof. Nozomu Togawa, for his constant guidance, support and encouragement during my years at Waseda University. He models many of the high quality characteristics that I aspire to emulate during my professional and personal life. Working with him has been and will continue to be a source of honor and pride for me.

I also thank Prof. Tatsuo Ohtsuki and Prof. Masao Yanagisawa for being my associate advisors and being in my dissertation reading committee. Without them, I am not able to study at Waseda University. I am very grateful for their invaluable help during the preparation of this thesis and several papers. It will always be a source of honor for me to have had the names of these world-class professors on my dissertation.

I also thank Professor Satoshi Goto, Professor Shinji Kimura and Professor Keiji Kimura of Waseda University for giving me a lot of helpful advices and continuous encouragement during my research. I also thank Professor Shinji Kimura and Professor Keiji Kimura of Waseda University for being in my dissertation reading committee.

I also thank Professor Youhua Shi for giving me a lot of helpful advices and support my research.

I have greatly appreciated all of the students, my friends and colleagues in Professor Goto's laboratory, Professor Kimura's laboratory, Professor Yangisawa's laboratory and Professor Togawa's laboratory, for their cooperations.

Last but certainly not least, I want to thank my family who have been a main source of success in my life. Their prayer, love and support carried me through the most difficult moments in my life. I thank my wife, Naoko Tosa, who always stand by my side and never give up on me. There are no words that I can use to thank her and lovely my baby, Mirae.

References

[1] M. Ali, M. Welzl, A. Adnan and F. Nadeem: Using the Ns-2 network simulator for evaluating network on chips (NoC), *Proceedings of ICET 2006*, pp. 506-512 (2006).

[2] T. Aoki, E. Hosoya, T. Otsuka and A. Onozawa: A novel hardware algorithm for real-time image recognition based on real AdaBoost classification, Proceedings of ISCAS 2012 (2012),

[3] G. Ascia, V. Catania, and M. Palesi: Multi-objective mapping for meshbased NoC architectures, *Proceedings of Int. Conf. Hardware-Softw. CodesignSyst. Synthesis*, pp. 182-187 (2004).

[4] 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 (2005).

[5] L. Benini and G. D. Micheli: Networks on chips: a new SoC paradigm, *Computer*, vol. 35, no. 1, pp. 70-78 (2002).

[6] 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 (2006).

[7] S. Bourduas and Z. Zilic: A hybrid ring/mesh interconnect for networkon-chip using hierarchical rings for global routing, Proceedings of International Symposium on Network-on-Chip 2007, pp. 195-204 (2007).

[8] N. Dalal and B. Triggs: Histogram of Oriented Gradients for Human Detection, *In: CVPR. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 886-893 (2005).

[9] W. J. Dally and B. Towles: Route packets, not wires: On-chip interconnection networks, *Proceedings of Design Automation Conference*, pp. 684-689 (2001).

[10] R. Dick: Embedded System Synthesis Benchmarks Suites (E3S) [Online], Available: http://ziyang.eecs.umich.edu/ dickrp/e3s/.

[11] J. Dongarra, I. Foster, G. C. Fox, W. Gropp, K. Kennedy, L. Torczon and A. White: The Sourcebook of parallel computing, *Morgan Kaufmann* (2003).

[12] K. Fall and K. Varadhan: The Ns manual, *The VINT Project* (2001).

[13] A. G. Farrell: Introduction to maple programming, *The Maws, Kilcock Co. Kildare, Ireland* (2005).

[14] G. Fen, W. Ning and W. Qi: Simulation and performance evaluation for network on chip design using OPNET, *Proceedings of TENCON 2007*, pp. 1-4 (2007).

[15] A. Y. Grama, A. Gupta and V. Kumar: Isoefficiency: Measuring the scalability of parallel algotithms and architecures, *IEEE Parallel and Distributed Technology*, vol. 1, no. 3, pp. 12-21 (1993).

[16] J. L. Gustafson: Reevaluating Amdahl's Law, *Communications of the ACM*, vol. 31, no. 5, pp. 532-533 (1988).

[17] A. Hansson, K. Goossens and A. Radulescu: A unified approach to mapping and routing on a network-on-chip for both best-effort and guaranteed service traffic, *Hindawi VLSI Design*, vol. 2007, pp. 243–264 (2007).

[18] H. M. Harmanani and R. Farah: A method for efficient mapping and reliable routing for noc architectures with minimum bandwidth and area, Proceedings of IEEE Northeast Workshop on Circuits and Systems and TAISA Conference 2008, pp. 29-32 (2008).

[19] J. Henkely, W. Wolfz and S. Chakradhary: On-chip networks: A scalable and communication-centric embedded system design paradigm, *Proceedings of ICVLSI Design*, pp. 845-851 (2004).

[20] R. Holsmark, M. Palesi and S. Kumar: Deadlock free routing algorithms for mesh topology NoC systems with regions, *Proceedings of DSD 2006*, pp. 696-703 (2006).

[21] J. Hu and R. Marculescu: Energy-aware mapping for tile-based NOC architectures under performance constraints, *Proceedings of Asia South Pacific Des. Autom. Conf.*, pp. 233-239 (2003).

[22] J. Hu and R. Marculescu: Energy- and performance-aware mapping for regular NoC architectures, *IEEE Transactions on Compututer-Aided Design Integreted Circuits and System*, vol. 24, no. 4, pp. 551-562 (2005).

[23] B.J. LaMeres, K. Gulati and S. P. Khatri: Controlling inductive cross-talk and power in off-chip buses using CODECs, *Proceedings of ASP-DAC 2006*, pp. 6-12 (2006).

[24] 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 (2007).

[25] W. Lee and G. E. Sobelman: Mesh-Star Hybrid NoC Architecture with CDMA Switch, *Proceedings of ISCAS 2009*, pp. 1349-1352 (2009).

[26] X. Leng, N. Xu, F. Dong and Z. Zhou: Implementation and simulation of a cluster-based hierarchical NoC architecture for multi-processor SoC, *Proceedings of ISCIT 2005*, vol. 2, pp. 1203-1206 (2005).

[27] S. Lin, L. Su, H. Su, D. Jin and L. Zeng: Hierarchical cluster-based irregular topology customization for Networks-on-Chip, *Proceedings of EUC '08*, vol. 1, pp. 373-377 (2008).

[28] D. G. Lowe: Distinctive Image Features from Scale-Invariant Keypoints, *Proceedings of IJCV*, pp. 91-110 (2004).

[29] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. 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 (2009).

[30] K. Mikolajczyk, C. Schmid and A. Zisserman: Human Detection Based on a Probabilistic Assembly of Robust Part Detections, *Proceedings of ECCV 2004. LNCS*, pp. 69-81 (2004).

[31] F. Moraes, N. Calazans, A. Mello, L. Moller and L. Ost: HERMES: An infrastructure for low area overhead packet-switching networks-on-chip, *Integration The VLSI Journal*, vol. 38, pp. 69-93 (2004).

[32] S. Mostafavi, A. Khonsari, M. S. Talebi and M. O. Khaoua: Rate control for scalable multimedia applications in Network-on-Chips, *Proceedings of SCALCOM-EMBEDDEDCOM'09*, pp. 621-626 (2009).

[33] S. Murali and G. D. Micheli: Bandwidth-constrained mapping of cores onto NoC architectures, *Proceedings of Des., Autom. Test Eur. Conf.*, pp. 896-901 (2004).

[34] M. Modarressi and H. Sarbazi-Azad: Power-aware mapping for reconfigurable noc architectures, Proceedings of IEEE International Conference on Computer Design 2007, pp. 417-422 (2007).

[35] L. M. Ni and P. K. McKinley: A survey of wormhole routing techniques in direct networks, *Computer*, vol. 26, no 2, pp. 62-76 (1993).

[36] J. Niemann, M. Porrmann and U. Ruckert: A scalable parallel SoC architecture for network processors, *IEEE Annual Symposium on VLSI*, pp. 311-313 (2005).

[37] U. Y. Ogras, R. Marculescu, H. G. Lee and N. Chang: Communication architecture optimization: making the shortest path shorter in regular networks-on-chip, *Proceedings of DATE '06*, pp. 6-12 (2006).

[38] U. Orgas, J. Hu, and R. Marculescu: Key research problems in NoC design: a holistic perspective, *Proceedings of CODES+ISSS*, pp. 69-74 (2005).

[39] A. Radulescu, J. Dielissen, S. G. Pestana, O. P. Gangwal, E. Rijpkema, P. Wielage and K. Goossens: An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration, *IEEE Transactions on Computer-Aided Design Integreted Circuits and System*, vol. 24, no. 1, pp. 4-17 (2005).

[40] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage and E. Waterlander: Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip, *Proceedings of Computers and Digital Techniques*, vol. 150, no. 5, pp. 294-302 (2003)

[41] T. D. Richardson, C. Nicopoulos, D. Park,V. Narayanan, Y. Xie, C. Das and V. Degalahal: A hybrid SoC interconnect with dynamic TDMA-based transaction-less buses and on-chip networks, *Proceedings of ICVLSI Design 2006*, pp. 8-16 (2006).

[42] M. R. Seifi and M. Eshghi: A clustered NoC in group communication, *Proceedings of TENCON 2008*, pp. 1-5 (2008).

[43] Y. Sekihara, T. Aoki and A. Onozawa: Efficient packet transmission priority control method for network-on-chip, *Proceedings of SASIMI 2012*, pp. 503-507 (2012).

[44] K. Srinivasan and K. S. Chatha: A technique for low energy mapping and routing in network-on-chip architectures, *Proceedings of Int. Symp. Low Power Electron. Des.*, pp. 387-392 (2005).

[45] K. L. Tsai, F. Lai, C. Y. Pan, D. S. Xiao, H. J. Tan and H. C. Lee: Design of Low latency on-chip communication based on hybrid NoC Architecture, *Proceedings of NEWCAS 2010*, pp. 257-260 (2010).

[46] P. Viola and M. Jones: Rapid object detection using a boosted cascade of simple features, *IEEE Conf. on CVPR*, pp. 511-518 (2001).

[47] Y. Wei, P. Hongbing, P. Peng, L. Li, G. Minglun, H. Ning, D. Gaoming and Z. Duoli: Application-level pipelining on Hierarchical NoC, *Proceedings of ISCAS 2010*, pp. 3873-3876 (2010).

[48] B. Wu and R. Nevatia: Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors, *Proceedings of ICCV*, pp. 90-97 (2005).

[49] P. Zarkesh-Ha, G. B. P. Bezerra, S. Forrest, and M. Moses: Hybrid network on chip (HNoC): Local buses with a global mesh architecture, *Proceedings of SLIP 2010*, pp. 9-14 (2010).

[50] C. A. Zeferino and A. A. Susin: SoCIN: A parametric and scalable network-on-chip, *Proceedings of SBCCI 2003*, pp. 169-174 (2003).

List of Publications

Journal

- ◯ S. Lee, M. Yanagisawa, and N. Togawa, “A locality-aware hybrid NoC configuration algorithm utilizing the communication volume among IP cores,” IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E95-A, No. 9, pp. 1538–1549 (2012).

International Conference

- ◯ S. Lee, M. Yanagisawa and N. Togawa, “A Locality-aware Hybrid NoC Configuration Algorithm Utilizing the Communication Volume among IP Cores with a Router Soft-core,” in *Proceedings of AKC 2013* (2013).

- ◯ S. Lee, N. Togawa Y. Sekihara, T. Aoki, M. Nakanishi and A. Onozawa, “An FPGA Based BMNoC Architecture Consisting of Hybrid Connections and Hierarchical Structures with a Router Soft-core,” in *Proceedings of ITC-CSCC 2013* (2013).

- ◯ S. Lee, N. Togawa, Y. Sekihara, T. Aoki and A. Onozawa, “An FPGA based hybrid NoC architecture utilizing packet transmission priority control method,” in *Proceedings of ICEIC 2013* (2013).

- ◯ S. Lee, N. Togawa, Y. Sekihara, T. Aoki and A. Onozawa, “A hybrid NoC architecture utilizing packet transmission priority control method,” in *Proceedings of APCCAS 2012*, pp 404–407 (2012).

- ○ S. Lee, N. Togawa, T. Aoki and A. Onozawa, “A locality-aware hybrid NoC configuration algorithm and its application to object detection systems,” in *Proceedings of ITC-CSCC 2012* (2012).

- ○ S. Lee, N. Togawa, T. Aoki and A. Onozawa, “A novel BMNoC configuration algorithm utilizing communication volume and locality among cores,” in *Proceedings of ISCAS 2012*, pp 1668–1671 (2012).

- ○ S. Lee, M. Yanagisawa, T. Ohtsuki and N. Togawa, “A throughput- and bandwidth- aware novel NoC architecture comprised of bus-based connection and global mesh routers,” in *Proceedings of ITC-CSCC 2011* (2011).

- ○ S. Lee, M. Yanagisawa, T. Ohtsuki and N. Togawa, “BusMesh NoC: A novel NoC architecture comprised of bus-based connection and global mesh routers,” in *Proceedings of APPCAS 2010*, pp. 712–715 (2010).

- ○ S. Lee, M. Yanagisawa, T. Ohtsuki and N. Togawa, “A throughput-aware busmesh NoC configuration algorithm utilizing the communication rate between IP cores,” in *Proceedings of SASIMI 2010*, pp. 96–101 (2010).

Invited Talks

- 2013 年 2 月 S. Lee, N. Togawa, T. Aoki and A. Onozawa, “A locality-aware hybrid NoC configuration algorithm and its application to object detection systems,” The Korean Scientists and Engineerings Association in Japan, 情報システム分科会.

- 2013 年 1 月 S. Lee, M. Yanagisawa, and N. Togawa, “A locality-aware hybrid NoC configuration algorithm utilizing the communication volume among IP cores,” KOTRA Global Career Vision 2013, Ph.D Research Presentation Section.

- 2012 年 2 月 S. Lee, M. Yanagisawa, T. Ohtsuki and N. Togawa, “BusMesh NoC: A novel NoC architecture comprised of bus-based connection and global mesh routers,” The Korean Scientists and Engineerings Association in Japan, 情報システム分科会.

Domestic Conference

- 大塚卓哉、細谷英一、青木 孝、小野澤 晃、李昇周、戸川 望： 多数ビデオ入力に対する画像認識ハードウェアの制御方式の提案、電子情報通信学会総合大会, A-3-7 (2012).

- 片野弘規、李昇周、戸川望、青木孝、関原悠介、中西衛： 局所性を考慮したマルチ FPGA システム向けタスクマッピング手法, VLSI 設計技術研究会、vol. 113, no. 416, VLD2013-126, pp. 143–148 (2014).

Patent

- (発明者) 関原悠介, 青木孝, 戸川望, 李昇周, (出願人) 日本電信電話株式会社, 学校法人早稲田大学, 計算機システム, ルータ装置, パケット転送方法およびプログラム, 特願 2013-122601, 2013 年 6 月 11 日出願.

- (発明者) 小野澤晃, 青木孝, 戸川望, 李昇周, (出願人) 日本電信電話株式会社, 学校法人早稲田大学, 特願 2012-153450, 計算システム，処理装置，及び計算システムにおける内部負荷分散方式, 2012 年 7 月 9 日出願.

- (発明者) 小野澤晃, 青木孝, 戸川望, 李昇周, (出願人) 日本電信電話株式会社, 学校法人早稲田大学, 特願 2012-033894, 画像処理システムの構成装置および構成方法, 2012 年 2 月 20 日出願.