Exploring Data Transfer with ~~NoC~~Network-on-Chip based Many-Core Processors

Comment [Editor1]: Remark: Please check the journal guidelines for the preferred use of abbreviations in the title.

ABSTRACT

[TBD]

Keywords

Many-core; ~~Multi-core~~Multicore; Network-on-Chip

1. INTRODUCTION

The evolution of a next generation computing platform oriented toward multi/many cores is ~~becoming~~ unavoidable to satisfy ~~a~~ the demand for increasing ~~demand of~~ computation in conjunction with reasonable power consumption. ~~In recent years, computation amount~~Recently, the computational ability of single core ~~processor is constantly reaching~~processors has reached its limit, and ~~persistence~~thus the applicability of Moore's law [19] is unclear. Multi/many core architecture is ~~a major~~an important trend ~~of interest~~ in ~~the last~~recent years. ~~Integrating~~ as it integrates cores~~, they~~ to realize high-performance and general-purpose computing with low power consumption. ~~This~~Hence, high energy efficiency is ~~the~~a superior feature of multi/many core platforms.

Comment [Editor2]: Remark: Please check the journal guidelines for the preferred format of in-text citations. In general, they should appear in a ascending numerical order.

~~Above~~ As detailed in the preceding paragraph, the demand for increasing ~~demand of~~ computation in conjunction with reasonable power consumption drives the need for multi/many core architecture in ~~many~~several domains. Real-time embedded systems are ~~one~~an example ~~of adapting~~in which multi/many core platforms ~~because~~are adapted since they face increasing processing requirements. ~~In this domain, countless opportunities~~ Extant studies have examined numerous applications of multi/many cores platforms ~~are discussed~~ [4], [25], [22], [21], [5].

For example, automotive systems ~~contain~~involve various applications and ~~some of their~~ are sometimes characterized by demands for high-performance computing. Automotive applications are responsible for ~~many~~several control systems such as the powertrain, ~~the~~ chassis, ~~the~~ steering wheel, driver assistance, and user interface. Advanced driver assistance ~~system (ADAS) is becoming more and more complex~~systems are characterized by increasing complexity and computational requirements and ~~is required~~necessitate intelligence such as an autonomous driving system. ~~Their complexity and computing requirements are continuously growing. Automotive systems require more computing resources. Meanwhile, it needs to realize extraordinary~~ Moreover, they also require significant energy efficiency and ~~reduce the costs~~cost reduction. Although modern automotive systems are ~~constructed~~composed of ~~a lot of~~several Electronic Control Unit (ECU) managing subsystems, there is a

Comment [Editor3]:
Tip: American-British Style
In American English, a comma (called serial or Oxford comma) is inserted before "and" in a series of three or more items.

~~trend~~shift from numerous scattered ECUs to hierarchical multi/many core Domain Controllers (DC). ~~Obviously, the~~ Evidently, a hybrid scenario is also ~~available. Combing~~applicable. Specifically, DCs combine performance with low power consumption~~. DCs~~ to realize high parallel processing and efficient data transport. ~~In addition~~Furthermore, flexible scalability of a many-core platform ~~assists~~facilitates development efficiency.

~~For another~~A related example~~.~~ involves avionics systems ~~are~~that also ~~responsible for~~include various applications and ~~a need of~~require cost efficiency. ~~With highly integrated trend and increasing demand of processing, multi~~Multi/many core platforms ~~have opportunities~~can be applied in the avionics domain~~.~~ due to their highly integrated characteristics and increasing processing demands. Low power consumption of cores leads to lower cooling requirements, ~~which~~and this is ~~the~~a critical issue for avionics. Multi/many core platforms reduce ~~not only~~ energy consumption ~~but also~~as well as the costs and weight. ~~The~~Weight reduction ~~of weight~~ is the most important ~~result~~factor for avionics~~. Integration~~, and integration with ~~the~~a multi/many core platform reduces the amount of processing ~~board~~boards and cooling units. Thus, ~~utilizing~~the utilization of an integrated platform ~~saves~~results in space, weight, and ~~the costs~~cost reductions.

~~Considering above background~~Hence, multi/many core platforms are ~~drawn up~~fabricated and released as commercial off-the-shelf (COTS) ~~multi core~~multicore components. ~~They are delivered significant attention in recent years. Kalray's the~~ Increasing research attention has focused on multi/many core platforms. The Multi-Purpose Processing Array (MPPA) developed by Kalray 256 [12], [11], [15], ~~Intel's~~ Single-chip Cloud Computer (SCC) developed by Intel [1],[3], and ~~Tilera's~~ Tile64 developed by Tilera [2] ~~have clustered~~ include the clustering of many-core architectures~~, where~~ in which cores are mapped closely. ~~Their~~The clusters of cores are capable of performing separate independent applications with respect to the desired power envelope of embedded applications. Kalray's MPPA-256 packs 256 general-purpose cores~~, which is overwhelming.~~ This significantly exceeds the number of cores in other ~~COTS's cores. With high performance per watt,~~ COTS, and it targets embedded systems, ~~HPC,~~high-performance computing (HPC), image processing, and networking~~.~~ due to the high-performance per watt. Intel's Xeon Phi [7] [8] is ~~one of high performance computing (HPC) accelerators.~~ an HPC accelerator. Xeon Phi is based on x86 architecture and targets use-case of data centers and workstation.

Comment [Editor4]: Remark: Please consider revising this as "*…its targeted use corresponds to data centers and workstations*" if it retains your intended meaning.

~~Although~~ Despite the emergence of the need ~~of~~for multi/many core platforms~~ has emerged, there are~~, several difficulties ~~for~~persist in the adaptation of these platforms to real-time embedded systems [4], [25]. These difficulties are caused by ~~their~~the hardware architecture and strict requirements of embedded systems. ~~One~~A difficulty ~~is that cores share~~ involves the sharing of numerous resources (e.g.~~.~~., memory subsystems and I/O devices~~). Since parallelized~~) by cores. Parallelized complex processes share some resources, ~~contention to these is frequently occurred.~~and this leads to the frequent occurrence of conflicts. Cache coherency is also a critical problem ~~owing to their~~caused by the presence of numerous cores. These difficulties disturb predictable timing behavior and software analysis. ~~Ensuring~~Thus, the timing requirement of real-time embedded systems ~~is still an open issue. In multi/many core platforms, the~~continues to warrant solutions. The impact of integrating real

time applications in multi/many core platforms, is not completely understood ~~yet~~to date. This is a critical issue because embedded systems have requirements for reliable and predictable behavior.

Contributions: ~~In this paper, we clarify the~~This study focused on clarifying performance characteristics of currently-achievable data transfer methods for many-core computing based on Network-on-Chips (NoC), such as MPPA-256, while ~~unveiling~~examining several new data transfer methods.

[TBD]

To the best of ~~our~~the authors' knowledge, this is the first ~~evidence of~~study that examines data transfer matters for many-core computing beyond an intuitive expectation~~, which allows~~ to allow system designers to choose appropriate data transfer methods~~, depending~~ based on the requirement of their latency-sensitive many-core applications. ~~These~~It is expected that the findings of the study are ~~also~~potentially applicable for ~~same~~several types of many-core architectures ~~rather than~~ as opposed to solely for MPPA-256. ~~We believe that the~~ The contributions of this ~~paper are useful for~~ study can be used in low-latency many-core computing.

Organization: The remainder of this ~~paper~~study is organized as follows. First, ~~Section 2 summarizes our considering~~the system model ~~for~~ considered in this ~~paper. We present our~~study is discussed in Section 2 in which the hardware model,

~~1. e.,~~namely Kalray MPPA-256 Bostan, and ~~our~~the system model ~~there.~~are presented. Second, Section 4 illustrates ~~our~~ experimental evaluations. ~~Then~~Subsequently, Section 5 ~~presents~~examines related ~~work about~~studies that focus on multi/many core systems. Finally, Section 6 ~~concludes this paper~~presents the conclusions and ~~suggests~~ directions for future ~~work~~research.

2. SYSTEM MODEL

~~In this~~This section~~, we present our~~ presents the system model used throughout ~~this paper. We consider~~ the study. The many-core model of Kalray MPPA-256 Bostan~~.~~ is considered. First, ~~the~~a hardware model is introduced in Section 2.1, and this is followed by ~~the~~a software model in Section 2.2.

2.1— Hardware Model

The MPPA-256 processor is based on an array of compute clusters (CCs) and I/O subsystems (IOSs) that are connected to nodes of Network-on-Chip (NoC) with a toroidal 2D topology (~~see~~as shown in Figures 1, 2, and 3). The MPPA MANY-CORE chip integrates 16 computing~~e~~ clusters and 4 ~~I/O subsystems~~IOS on NoC. ~~We present the~~The architecture of Kalray MPPA-256 is presented in this section.

Comment [Editor5]: Remark: Please confirm that this change is accurate.

2.1.1 — I/O Subsystems~~IOSs~~ (IOS)

MPPA-256 has the following four I/O subsystems (IOSes): North, South, East, and West IOSes. The North and South IOSes are connected to a DDR interface and an eight-lane PCIe controller. The East and West IOSes are connected to a quad 10Gb/s Ethernet controller. Two pairs of IOSes organize two I/O clusters (IOCs) as shown in Figure 1.

Each ~~I/O subsystem (~~IOS~~)~~ comprises of quad core resource managers (RMs) and a network interface.

~~• Resource Managers (~~• **RMs)~~:** RM cores are connected to a 16-banked parallel shared memory ~~of 2MB~~with a total capacity (IO SMEM~~).~~) of 2 MB as shown in the left side of Figure 1~~.~~. The four RMs have their own instruction cache 8-way associative ~~of~~corresponding to 32 (8 *x* 4) KB and share a data cache 8-way ~~associative of~~ with 128 KB and external DDR access. ~~Sharing~~ The sharing of the data cache of 128 KB ~~makes it coherent~~allows coherency between the RMs. Additionally, RM cores operate controllers for the PCIe, Ethernet, Interlaken, and other I/O devices. They ~~are able to~~ operate the local peripherals~~.~~ including the network interfaces with DMA. ~~We can~~It is also ~~make~~possible to conduct an application run on ~~these~~the RMs.

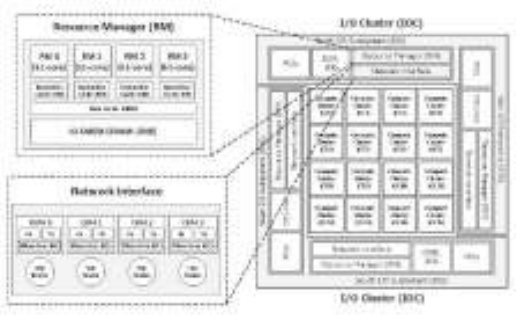

Figure 1: ~~Overview~~An overview of the architecture of the Kalray MPPA-256 Bostan ~~architecture~~.

~~• A~~ **Network Interface:** ~~A~~The network interface contains four DMA engines (DMA0-3) and four NoC routers as shown in the left side of Figure 1, and the IOS DMA engine manages

transfers between the IO SMEM, the IOS DDR, and

the IOS peripherals (e.g., PCIe interface and Ethernet controllers). ~~Through NoC Routers.~~The DMA engine transfers data between routers on NoC~~.~~ through NoC routers. The DMA engine has the following three NoC interfaces: a receive (Rx) interface, a transmit (Tx) interface, and a micro core (UC). ~~Micro~~A micro core is a network processor ~~programmable~~that can be programed to set threads sending data with a Tx interface. A UC ~~is able to~~can extract data from memory by using a ~~programmed~~programed pattern and ~~to~~ send ~~them~~the data on the NoC. ~~Once~~After it is initiated, this ~~is made~~continues in an autonomous fashion without using a Processing Elements (PE) and an RM.

2.1.2— Compute Clusters (CC)

In MPPA-256, the 16 inner nodes of the NoC correspond to the CCs. Figure 2 illustrates the architecture of each CC.

• ~~**Processing Elements (PEs)**~~ **and an RM:** In a CC, 16 processing elements (PEs) and ~~one~~a RM share ~~2MB~~2 MB cluster local memory (SMEM) that is composed of 16 independent memory banks ~~of~~with ~~16 independent memory banks of 16,~~ 384 *x* ~~64bit. Each bank of~~64-bit. The capacity of each SMEM ~~has a capacity of~~bank corresponds to 128 KB. ~~These~~The PEs are mainly used by users for parallel processing. Developers spawn computing threads on PEs. The PEs and an RM in CC ~~are~~correspond to the Kalray-1 cores, which implement a 32-bit 5-issue Very Long Instruction Word ~~(VLIW)~~ architecture with

~~16W 600MHz~~16 W–600 MHz (typical) or ~~24W 800MHz~~24 W–800 MHz. Each core is

fitted with its own instruction and data caches. Each cache is a 2-way associative with a capacity of 8 KB. Thus, 17 k1-cores (a PE or the RM) share a multi-banked

~~2MB~~2 MB SMEM.

• ~~A~~ **Debug Support Unit (DSU) and** ~~a~~ **Network Interface:**

In addition to PEs and an RM, bus masters on SMEM ~~are the debug support unit (~~correspond to a DSU~~)~~ and a DMA engine in a network interface. A DMA engine and a NoC router are laid out in a network interface. ~~Similar~~In a manner similar to IOS, the CC DMA engine ~~has~~ also has the following three interfaces: an Rx, a Tx, and a UC. It is instantiated in every cluster and connected to the SMEM.

Comment [Editor6]: Remark: Please confirm that this change is accurate.

Comment [Editor7]: Remark: Note that the intended meaning is not clear. Please check this term and revise accordingly.

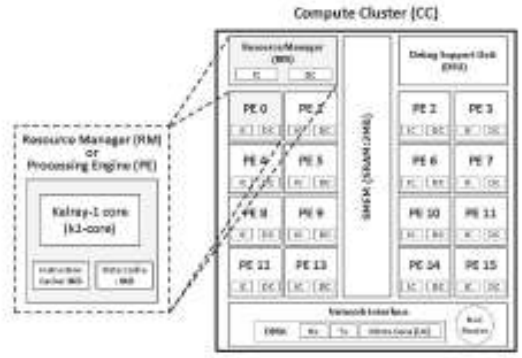

Figure 2: ~~Compute~~ Computation of Cluster ~~(CC)~~ architecture.

2.1.3 ~~Network on Chip (~~NoC~~)~~

The 16 CCs and the 4 IOSs are connected by a NoC as shown in Figure 3. Furthermore, a NoC is constructed on the bus network and has routers on each node.

• **Bus Network:** Bus network connects nodes (CCs and IOSs) with a torus topology [9~~]~~], which has ~~twice~~ dual bands and ~~takes~~involves a low average number of hops when compared to mesh topology [27], [26]. The network is actually composed of the following two parallel ~~NoC~~NoCs with bi-directional links (denoted by red lines in Figure 3): the data NoC (D-NoC~~), which~~) that is optimized for bulk data transfers~~.~~ and the control NoC (C-NoC~~), which~~) that is optimized for small messages at low latency. Functionally, NoC ~~is~~ corresponds to a packet switched network. Data are packaged in variable length packets ~~which~~that circulate between routers in a wormhole manner~~:~~ in which packets are broken into small pieces called flits (flow control digits). The NoC traffic is segmented into packets~~:~~, and each packet ~~has~~includes 1 to 4 header flits and 0 to 62 payload data flits.

Comment [Editor8]: Remark: Please confirm that this change is accurate.

• **NoC ~~Routers~~routers:** A node per compute cluster and four nodes per I/O subsystem ~~hold~~holds the following two own routers: a D-NoC router and a C-NoC router. Each RM on NoC node (CC or IOS) is associated with ~~these~~the fore-mentioned two NoC routers. Furthermore, DMA engines in a network interface on the CC/IOS send and receive flits through ~~these~~the D-NoC routers with the Rx interface, the Tx interface, and the UC. A mailbox component ~~which is~~corresponds to the virtual interface for the C-NoC and enables one-to-one, N-to-one, or one-to-N low-latency synchronizations. The NoC routers shown in Figures 1 and 2 illustrate nodes as R0-15, R128-131,

R160-163, R224-227, and R192-195 in Figure 3~~).~~. For

Comment [Editor9]: Remark: Please consider revising this as "routers of its own."

purposes of simplicity, ~~we illustrate~~ D-NoC/C-NoC routers are illustrated with ~~one~~a NoC router. In both D-NoC and C-NoC, each network node (CC or IOS) ~~has~~includes the following 5-link NoC routers: four duplexed links for north/east/west and south ~~neighbours, one~~neighbors and a duplexed link for local address space attached to the NoC ~~Router.~~router. The NoC routers ~~have~~include FIFOs queuing flits for each direction. The data links are four bytes wide in each direction and operate at the CPU clock rate of ~~600MHz~~600 MHz or ~~800MHz.~~800 MHz, and therefore~~,~~ each tile

Figure 3: ~~Network on Chip (~~NoC~~)~~ connections (both D-NoC and C-NoC).

can transmit/receive a total of 2.~~4GB~~4 GB/s or 3.~~2GB/.~~2 GB/, which is spread across the four directions (i.e., north, south, east, and west).

2.2 Software Model

~~A~~ Figure 5 shows the software stack used for Kalray MPPA-245 ~~used~~ in ~~this paper is shown in Figure 4. Kalray's MPPA~~the present study. The Kalray system is an extensible and scalable array of computing cores and memory. ~~On such a~~ With respect to this type of a system, ~~we can~~ it is possible to map several ~~programming~~programing models or runtimes~~, e.g.,~~ such as Linux, real-time ~~Operating System~~operating system, POSIX API, OpenCL, and OpenMP. ~~We describe each~~Each layer is described in detail.

In the hardware abstraction layer, an abstraction package abstracts hardware of a CC, IOS, and NoC. ~~This~~The abstraction package serves as a system that ~~doesn't~~does not provide any services. The hardware abstraction is responsible for partitioning ~~the~~ hardware resources and controlling access to ~~these~~the resources from the user-space operating system libraries. ~~Moreover~~Additionally, the abstraction package ~~is able to retrieve the~~retrieves resources allocated to a partition at any time. It ~~is able to set~~sets-up and ~~control~~controls inter-partition communications as a virtual machine abstraction layer. The hardware abstraction runs on the dedicated RM core. All the services are commonly provided by a rich operating system (for e.g., virtual memory, interrupts, ~~scheduler, etc...)~~and schedule) that must be provided by user-space libraries. Consequently, each runtime or operating system implements its own ~~required~~ services that are optimized to ~~its~~specific needs. This is because each ~~programming~~programing model or runtime ~~has~~involves different requirements. ~~Minimal~~A minimal kernel avoids ~~the waste~~wastage of resources and mismatched needs.

In a low-level ~~Library~~library layer, the Kalray system also provides libraries for handling NoC. Additionally, NoC features, such as routing~~.~~ and quality of service, are ~~to be~~ set by the programmer. The Libnoc allows direct access to memory mapped registers for their configurations and ~~use~~uses. It is designed to cause a minimum amount of CPU overhead. It also serves as a minimal abstraction for

Figure 4: ~~Kalray MPPA-256's~~ The software stack of Kalray MPPA-256.

resource allocation. ~~The~~ Librouting offers ~~the~~a minimal set of functions that can be used to ~~use for routing~~ route data between any clusters of the MPPA~~, both with~~ including unicast (one target) ~~mode~~modes or multicast (multiple targets) ~~mode. Routing~~ modes. As shown in Figure 3, routing on the torus network ~~shown in Figure 3~~ is statically conducted with its own policy. The Libpower enables spawning and waiting for the end of execution of a remote cluster.

~~In OS layer, various~~Various operating systems support the abstraction package~~. We introduce~~ in the OS layer. The following Real-Time

Operating System (RTOS~~).~~) is introduced:

• **RTEMS:** RTEMS~~, the~~ (Real-Time Executive for Multiprocessor Systems~~,~~) is a full featured RTOS prepared for embedded platforms. It supports several APIs and standards, and most notably supports the POSIX API. The system provides a rich set of features, and an RTEMS application ~~is most of the time just~~ mostly corresponds to a regular C or C~~+~~+H program ~~using~~that uses the POSIX API. ~~We are able to build~~Additionally,

RTEMS can be built on the IOC.

• **NodeOS:** On CC, ~~we can build~~ the ~~MPPAcluster~~MPPA cluster operating system utilizes a runtime called ~~NodeOS. This~~NodeOS. The OS addresses the need for a multicore OS ~~conforming as much as~~to conform to the maximum possible extent to the standard POSIX API. The NodeOS enables a user code by using POSIX API to run on PEs on CC. First, NodeOS runtime starts on PE0 ~~before~~prior to calling the user main function. ~~Then, we can call~~Subsequently, pthread is called on other PEs.

Comment [Editor10]: Remark: Please consider revising this to “C++.”

Comment [Editor11]: Remark: Please check whether the edits retain your intended meaning.

• **eMCOS:** On both CC and IOS, eMCOS provides minimal ~~programming~~programing interfaces and libraries. Specifically, eMCOS is a real-time embedded operating system developed by eSOL~~,~~ (a Japanese supplier for RTOS~~.~~), and eMCOS is the world's first commercially available many-core RTOS for use in embedded systems. This OS implements a distributed micro-kernel architecture. ~~This~~The compact micro-kernel is equipped with only minimal functions. It enables applications to operate priority based message passing, local thread scheduling, and thread management on ~~not only~~ IOS ~~but also~~as well as CC.

RTMES and NodeOS are provided by Kalray, and eMCOS

is released by eSOL.

3. DATA TRANSFER FRAMEWORK

In this section, ~~we explain~~ data transfer methods in MPPA-256~~:~~ are explained. For scalability purposes, MPPA-256 accepts clustered architectures ~~where~~in which each cluster contains its own memory. 16 cores are packed as a cluster and they share 2 MB memory (SMEM) as shown in Figure 2. ~~Avoiding~~This avoids frequent memory contention ~~by~~due to numerous cores~~, this~~ and helps ~~increment of~~in increasing the number of cores. However, ~~this architectures~~the architecture constraints memory ~~address which cores are able to access~~that can be directly~~. To~~ accessed by the cores. In order to communicate with cores outside the cluster, ~~we have~~it is necessary to transfer data between clusters through the D-NoC with network interfaces.

Comment [Editor12]: Remark: Please check whether the edits retain your intended meaning.

~~In~~ A Rx interface exists in the receiving side~~, there is Rx interface~~ to receive data with DMA. ~~We have~~It is necessary to allocate a D-NoC Rx resource and configure it to wait ~~receiving~~to receive the data. ~~One~~A DMA in a network interface contains 256 D-NoC Rx resources. ~~In sending side, we have two~~Two interfaces ~~for users to send data between clusters. One is~~, namely a Tx interface and ~~the other is~~a UC interface as explained in Sections 2.1.1 and 2.1.2~~.~~, are present with respect to the sending side for users to send data between clusters. The UC is a network processor ~~programmable~~ that is programmed to set threads ~~sending~~to send data in DMA. It executes ~~programmed~~a programed pattern and sends data through the D-NoC without a PE and an RM. The UC interface ~~provides~~results in faster data transfer ~~than~~when compared to that in the Tx interface. However, ~~One~~a DMA in a network interface contains only 8 D-NoC UC resources. Both interfaces use a DMA engine to access memory and copy data. ~~Regardless~~Irrespective of ~~using~~whether or not a UC interface ~~or not, we have~~is used, it is necessary to allocate a D-NoC Tx resource and configure it to send data. ~~If we use UC interface, we additionally~~Additionally, it is necessary to allocate and configure ~~a~~ D-NoC UC resources if a UC interface is used.

4. EVALUATIONS

~~In this~~This section~~, we have conducted~~ involves examining two ~~kinds~~types of evaluations~~. One is~~, namely a D-NoC data transfer evaluation~~. We explore~~ in which latency characteristics of interfaces and memory type~~. The other is~~ are explored and a matrix calculation evaluation~~. This test shows MPPA-256's~~ that demonstrates the parallelization potential ~~and characteristics~~ of the MPPA-256 and its memory access ~~when~~characteristics while dealing with large data.

4.1 D-NoC Data Transfer

In thisThis evaluation, we clarify involves clarifying end-to-end latency by considering the relation among interfaces (Tx or UC) ,), routing on NoC, and memory type (DDR or SMEM). To achieve above purpose, we prepareThis is achieved by preparing four routes as shown in Figure 5. TheseThe routes on D-NoC map (Figure 3) contains various connections between routers:, namely a direct link, a cross link, and a flying link. In case ofWith respect to routes from the IOS routers to the CC routers, transmitted data is allocated in DDR or IO SMEM. The CC has only includes SMEM as shown in Figure 2. We directly use A low-level library is directly used to transfer data with D-NoC. TransferredThe transferred data arecorrespond to 100 B, 1 KB, 10 KB, 100 KB, and 1 MB. TheseThe buffers are

sequentially allocated in DDR or SRAM (IO SMEM or CC SMEM). Since theThe capacity of CC SMEM is corresponds to 2 MB, we assumeand thus it is assumed that the appropriate communication buffer size is corresponds to 1 MB. On our assumtion,Given the assumption, the other memory area is own bycorresponds to the application, library, and operating system. In many situations, we have measured endEnd-to-end latencies are measured 1,000 times and drow in numerous situations as shown in Figures 6, 7, and 8, and boxplots are obtained as depicted in Figures 6, 7, 8, and 9, 10 and 11. Following, The following evaluations isare conducted on eMCOS.

Comment [Editor13]: Remark: Please check whether the edits retain your intended meaning.

Data transfer latencies between IOS and CC isare not influenced by routing. We prepareThis involved preparing two interfaces (Tx and UC), three routes (direct link, cross link, and detour route), and two memory location wherelocations in which the transferred data is allocated. As shown in Figures 6, 7, 8, and 9, end-to-end latencieslatency scales

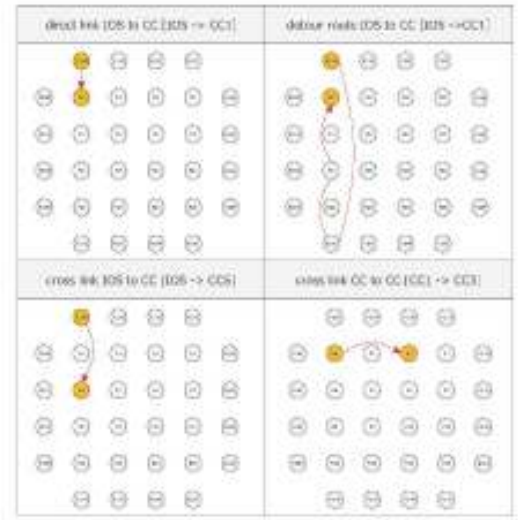

Figure 5: Four D-NoC routes used ~~for~~in the evaluation.

~~linearly~~exhibit a linear relation with data size ~~and~~ , and there are no significant differences between the three routes ~~do not produce differences in~~ with respect to data transfer latency. This result is important in torus topology NoC because the number of ~~its~~ minimum ~~step is larger than~~ steps exceeds those in the mesh ~~topology's one. We can observe~~ topology. It is observed that queuing in the NoC ~~Routers~~routers and hardware distance on the NoC ~~is~~are not dominant ~~factor~~factors for latency. ~~Transmitting~~The time taken by transmitting and receiving ~~transaction take much longer time than~~ transactions exceeds those of other transaction. ~~In addition, we~~ Additionally, it is briefly ~~recognize~~recognized that the speed of UC ~~is much faster than~~ exceeds that of Tx. ~~For~~The data is arranged as shown in Figures 10 and 11 to facilitate a precise analysis ~~for~~ with respect to the interface and memory location~~, we arrage data in Figures 10 and 11. In these~~ . In the figures, ~~we accept~~ only ~~cross link~~the crosslink from IOS to CC5 is accepted because routes do not influence latency. ~~For~~In order to facilitate intuitive recognition, ~~we arrange~~ two kinds of figures~~:~~ are arranged, namely a logarithmic axis ~~one~~ and a linear axis ~~one~~.

In the Tx interface, DDR causes a large increase in latency. The time taken by the DDR ~~takes~~is twice ~~time as long as~~ that of the IO SMEM as shown in Figure 11. This is ~~caused by~~due to the memory access speed characteristics of DRAM and SRAM. In the case of the Tx interface, it is necessary for an RM (k1-core) on IOS ~~has~~ to operate the DMA in the IOS network interface ~~of IOS. It~~. This is ~~thought that~~attributed to the fact that the core is involved in processing ~~is causing this result. Data~~. The speed of the data transfer latency between CCs ~~is a little faster than~~exceeds that between IOS and CC. This result indicates that the MPPA-256 is optimized for communication between the CCs.

In With respect to the UC interface, whetherthe latency is not significantly affected by the location at which the transferred buffer is allocated on(i.e., the DDR or SMEM has not much effect on latency. Same). Similar latency characteristics isare observed in Figures 10 and 11. In the case of the UC interface, an RM (k1-core) on the IOS does not involve a DMA transaction. A micro core in the network interface executes a programmedprograms thread sending data. This evaluation result tells that there is no need to be conscious of suggests that the slow access speed of the DDR is not significant in the case of the UC. As well asIn a manner similar to the Tx interface, Data the speed of the data transfer latency between CCs is faster thanexceeds that between IOS and CC.

4.2 —Matrix Calculation

In thisthe evaluation, we clarify matrix calculation time and

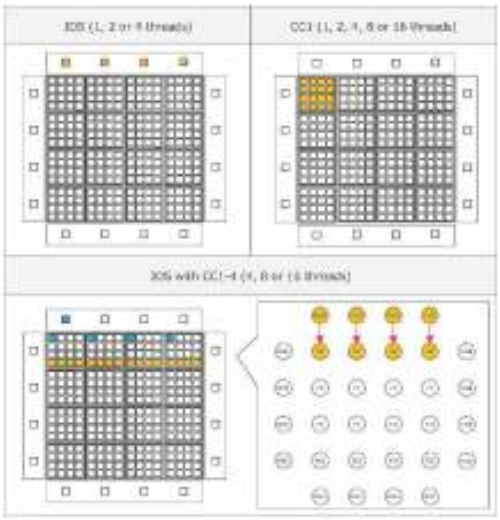

Figure 12: Situations of matrixMatrix calculation situations.

parallelization potential of MPPA-256. We have are clarified. Matrix calculations are conducted matrix calculation in IOS and CC. As shown in Figure 12, three Three computing situations isare considered. One is as shown in Figure 12. The first situation involves computing in IOS where four cores are available. ToIn order to analyze memory access characteristics, we allocatea matrix buffer is allocated in IO DDR and SMEM. Another isThe second situation involves computing in CC wherein which 16 cores are available. The other isthird situation involves offload-computing by using an IOS and four CCs. Parallelized processing is executed with four CCs cores and there SMEM. Somethree SMEMs. A

few cores in IOS and CC manage the parallelized transaction. ~~This~~The method ~~is able to~~can handle large data in which one cluster is not sufficient ~~cannot deal~~ because buffer capacity is not limited to ~~2MB of~~2 MB in SMEM. Parallelized processing and the total capacity of SMEM ~~is~~are superior to ~~computation in~~ IOS or CC~~. In~~ computations. With respect to the IOS, the application can handle large capacity data only in the DDR. However, ~~with~~in this method, ~~we can~~distributed memories are used to deal with large capacity data in SMEM ~~by distributed memories. For~~, In order to facilitate faster data transfer, a part of the matrix buffer is ~~parallelly~~ transmitted in parallel as shown in Figure 12. ~~To avoid cache coherency trouble,~~Thus, it is necessary for IOS and CC cores ~~must~~to access matrix buffers without cache to avoid cache coherency trouble.

~~We analyze matrix~~Matrix calculation time is analyzed with parallelization and memory allocation. ~~In addition, we analyze~~Additionally, the influence of cache is analyzed because cache coherency is an important issue in a many-core system. There are ~~many~~several cases ~~that~~in which applications must access specific memory space without a cache. ~~Since~~With respect to the given assumptions, maximum buffer size ~~is~~corresponds to 1 MB ~~in our assumption, we prepare~~, and thus three matrices buffers are prepared, and each size ~~is~~corresponds to 314 KB. ~~The matrix~~Matrix A and ~~the~~ matrix B are multiplied, and the result is stored in ~~the~~ matrix C. ~~We set the~~The total of the three matrices ~~to be~~is set as approximately 1 MB.

First, matrix calculation time with the cache in IOS and CC is ~~shown~~depicted in Figure 13. ~~Thanks to cache, there is~~There are almost no ~~difference~~differences between IO DDR, IO SMEM, and CC SMEM~~.~~ due to the cache. Furthermore, 128 KB data cache in the IOS works well and compensates for the DDR delay ~~of DDR~~. Additionally, ~~we can observe~~it is observed that calculation time scales ~~linearly~~exhibit a linear relation with the number of threads. This ~~is~~corresponds to ideal

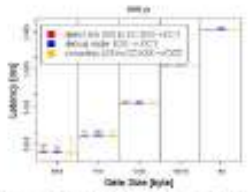

Figure 6: Data transfer with Tx from IO DDR to CC.

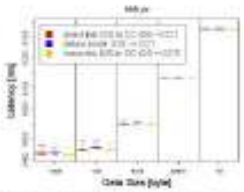

Figure 7: Data transfer with UC from IO DDR to CC.

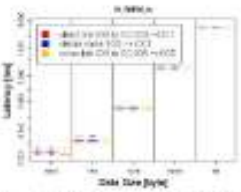

Figure 8: Data transfer with Tx from IO SMEM to CC.

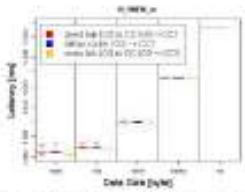

Figure 9: Data transfer with UC from IO SMEM to CC.

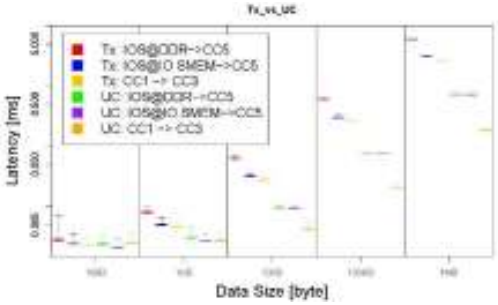

Figure 10: Data transfer with Tx/UC (logarithmic axis).

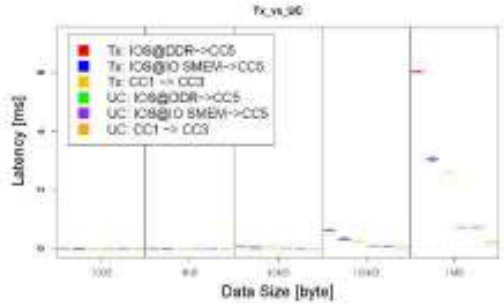

Figure 11: Data transfer with Tx/UC (linear axis).

behavior with respect to parallelization.

Second, matrix calculation time without cache in the IOS and CC is shown in Figure 14. ~~Due to the lack~~The absence of a cache~~, DDR has increased about~~ results in a fourfold increase in the DDR and a ~~big~~high difference arises with respect to the SMEM ~~has arisen.~~ . Another notable result is that calculation speed in

CC SMEM ~~is faster than~~exceeds that ~~in~~of the IO SMEM. This characteristic ~~hides~~is hidden in the calculation with the cache. ~~Since~~The computing cores ~~are~~involve the same k1-cores in IOS and CC, and thus it is ~~thought~~considered that the characteristics and physical arrangement of SMEM ~~are affecting~~exert a significant effect. This is ~~a~~an interesting result ~~that there is~~since a ~~big~~high difference that ~~can not~~cannot be ignored~~. We can~~ exists. It is also ~~observe~~observed that calculation time ~~scales linearly~~exhibits a linear relation with the number of threads. ~~The other notable result is that~~Furthermore, the calculation speed without the cache in CC SMEM ~~is faster than~~exceeds that with cache. ~~Although this~~This result is contrary to intuition, and the two factors are conceivable. A small data cache (8 KB) in CC does not function adequately ~~work~~, and direct assembly instruction for uncached access optimizes memory access.

Finally, matrix calculation with offload-computing in IOS and CCs is ~~is~~ shown in Figure 15. In this case, ~~we suppose that~~ it is assumed with respect to the calculation of large matrices ~~calculation~~ that the total capacity ~~is over~~exceeds 1 MB. ~~Because of this assumption, the~~The offloading result is compared with IO DDR (~~cashed). To~~cached) due to the fore-mentioned assumption. An overhead transaction (inverse matB and input matC) is performed to offload a part of calculation ~~onto~~on the four CCs and aggregate calculation result~~, overhead transaction (inverse matB and input matC) occurred. They produce~~. This produces a constant overhead ~~regardless~~irrespective of the number of threads as shown in Figure 15. However, the speed involved in offloading result ~~is faster than~~exceeds that of IO DDR (~~cashed). This~~ cached). The result ~~proves~~indicates several important facts. ~~One is that~~First, D-NoC data transfer ~~produce~~produces little overhead latency. ~~The other is that~~Second, the speed of DMA memory access to DDR ~~is much faster than an~~exceeds that of RM (k1-core)'s memory access. In the offloading case, a DMA accesses matrix buffers on DDR and ~~transfer them~~transfers the buffers from IO

DDR to each CC SMEM. ~~Then~~Subsequently, PEs in the CC access matrix

~~buffers for~~buffer the calculation without cache. ~~Since the~~The overhead of data transfer and DMA memory access is small, and thus parallel data transmission and distributed memory are practical in the case of MPPA-256. The impact of offloading ~~becomes larger~~increases when the matrix size is large as shown in Figure 16. ~~Since only~~Only a part of matrix is allocated in CC, ~~we can~~ and thus it is possible to handle larger matrices buffers. ~~We prepare~~Additionally, 640 KB matrices are prepared, and ~~evaluate~~ matrix ~~calculation~~calculations are evaluated with offload-computing. ~~Compared to result of 314 KB matrices in Figure 15.~~The speed of the offloading result ~~is much faster than~~exceeds that of IO DDR result with respect to the 314 KB matrices in Figure 15.

5. RELATED WORK

Comment [Editor14]: Remark: Note that this section forms the latter part of the introduction section. Hence, please consider moving this to the last part of the introduction section.

~~In this~~

This section~~, we compare~~ compares many-core platforms ~~to additional platforms~~ and ~~discuss~~discusses previous ~~work of~~ studies related to multi/many cores.

~~In recent years,~~Recently, studies indicate that the single core processors are characterized by limited computation performance ~~of~~. Pollack stated that a single core ~~processor~~ is ~~constantly coming to its limit. Pollack has said inefficiency of a single core in~~ inefficient [23] and that Moore's law [19] ~~has become unstable. To~~ is no longer applicable. Therefore, extant CPUs are not sufficient to satisfy increasing ~~a demand of~~ computation~~, CPU is not sufficient.~~ demands. Many other platforms including many-core are developed and researched ~~nowadays~~by current studies.

Table 1 summarizes the features of many-core platforms with that of other platforms. For instance, the GPU is a powerful device to enhance computing performance~~. In~~, and it has great potential in specific ~~area (~~areas (for e.g., image processing~~.~~ and learning~~), it has great potential.~~). However, it is mainly used for a specific purpose, and its reliability is not suitable for real-time systems. It is difficult to use a GPU for a global purpose and ~~a~~to guarantee ~~of~~its reliability due to the GPU architecture. ~~For~~Many-core is significantly superior to GPU with respect to a global purpose and multiple instructions~~, many-core is much superior to GPU. In addition~~, Additionally, it is commonly known that many-core ~~has~~involves a reasonable power consumption. In contrast, the GPU consumes a ~~lot~~significant amount of power and generates ~~much~~considerable heat. This is a critical problem for embedded systems. ~~DSP~~Furthermore, DSPs and ~~FPGA~~FPGAs are also high-performance devices when compared to ~~CPU~~CPUs. They are efficient in ~~a point~~terms of power consumption. A DSP is often used for real-time ~~system~~systems, and ~~FPGA guarantees~~FPGAs guarantee reliability and efficient processing. They

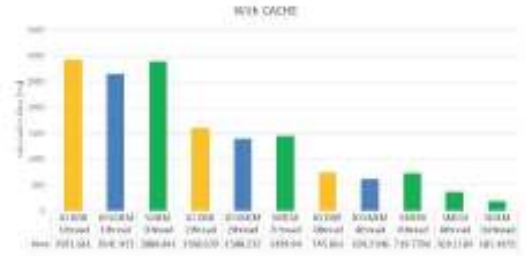

Figure 13: Matrix ~~calculation~~calculations in IOS and CC with cache.

IOS with CC1-4

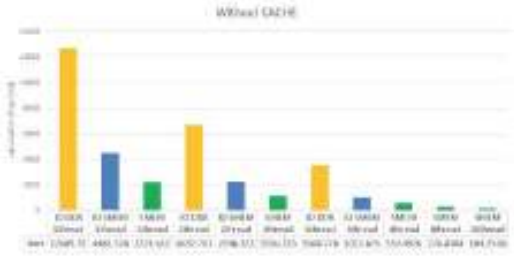

Figure 14: Matrix ~~calculation~~calculations in IOS and CC without cache.

IOS with CC1-4

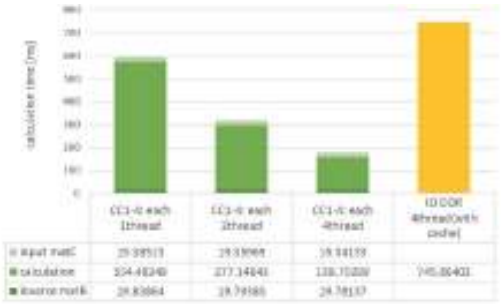

Figure 15: Matrix ~~calculation~~calculations with offload-computing (314 KB matrix x 3).

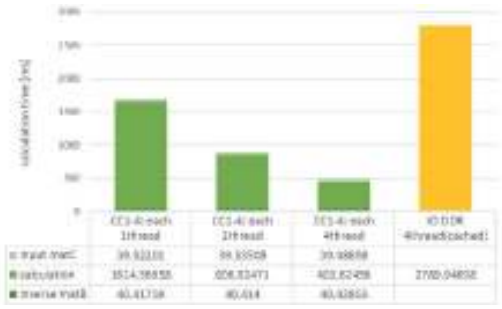

Figure 16: Matrix ~~calculation~~calculations with offload-computing (640 KB matrix x 3).

are suitable for time-critical computing [10]. However, ~~DSP~~DSPs cannot be used for global purpose and ~~programming FPGA is~~programing. Furthermore, FPGAs are difficult for software developers~~. Their~~ since their software model is ~~much~~significantly different from that of CPU and is not a substitute for CPU. Many-core platforms ~~have a potential to take the place of~~ can potentially replace single/multi-core CPU~~. Ease~~ as they possess ease of ~~programming~~programing and scalability with high acceleration ~~is highlighted~~.

Based on the ~~above~~fore-mentioned background, several extant studies examined real-time applications on many-core platforms ~~has received significant attention in recent years.~~. Many ~~commercial off-the-shelf (~~COTS~~) multi-core~~ multicore components are developed and released by several vendors~~.—~~ (e.g., Kalray's the Multi-Purpose ~~Pro-cessing~~Processing Array

(MPPA) 256 [12], [11], [15], Intel's Single-chip Cloud Computer (SCC) [1],[3], Tilera's Tile64 [2], and Intel's Xeon Phi

[7] [8~~]~~]). The present study focuses on the Kalray MPPA-256 that is designed for real-time embedded applications ~~and the target of this paper.~~. Kalray

5. A [12], [11], [15] ~~has~~ presented clustered many-core architectures on the NoC. ~~(Precise~~The precise hardware model is described in Section 2.1~~.)~~ ~~It~~. This is ~~often~~typically accepted ~~for a target of~~with respect to many-core platforms, and ~~various~~the model is used in several previous ~~work has considered this~~

~~model~~studies [22], [4], [6], [21]. [TBD]

6. ~~CONCLUSION~~CONCLUSIONS

[TBD]

Table 1: ~~Comparison~~A comparison of Many-core to CPU, GPU, DSP, and FPGA

	performance	**power/heat**	**reliability**	**real-time**	**software development**	**costs**	**multiple instruction**
CPU		*A*	✓	✓	✓	✓	*A*
GPU	✓		*A*		*A*	✓	
DSP	*A*	*A*	✓	✓	*A*	✓	
FPGA	✓	*A*	✓	*A*		*A*	
Many-core	✓	✓	✓	✓	✓	*A*	✓