81.6 GOPS Object Recognition Processor Based on a Memory-Centric NoC

Donghyun Kim, *Student Member, IEEE*, Kwanho Kim, *Student Member, IEEE*, Joo-Young Kim, *Student Member, IEEE*, Seungjin Lee, *Student Member, IEEE*, Se-Joong Lee, *Member, IEEE*, and Hoi-Jun Yoo, *Fellow, IEEE*

***Abstract*—For mobile intelligent robot applications, an 81.6 GOPS object recognition processor is implemented. Based on an analysis of the target application, the chip architecture and hardware features are decided. The proposed processor aims to support both task-level and data-level parallelism. Ten processing elements are integrated for the task-level parallelism and single instruction multiple data (SIMD) instruction is added to exploit the data-level parallelism. The Memory-Centric network-on-chip7 (NoC) is proposed to support efficient pipelined task execution using the ten processing elements. It also provides coherence and consistency schemes tailored for 1-to-N and M-to-1 data transactions in a task-level pipeline. For further performance gain, the visual image processing memory is also implemented. The chip is fabricated in a 0.18-μm CMOS technology and computes the key-point localization stage of the SIFT object recognition twice faster than the 2.3 GHz Core 2 Duo processor.**

***Index Terms*—Multiprocessing, network-on-chip (NoC), object recognition, VLSI.**

I. Introduction

OBJECT recognition plays an essential role in a wide range of applications including autonomous vehicles, robotic sentries and mobile intelligent robots. In vehicular applications, traffic lane markings and nearby cars are recognized to improve the driver's safety support [1]–[3] or to assist in navigation of autonomous vehicles [4]. In the case of robotic sentries, real time pattern recognition is used to distinguish human enemies from the background scenery [5]. Recently, object recognition has been the heart of vision-based localization and mapping technology in mobile intelligent robots [6]–[8].

Object recognition involves image processing stages including numerous iterations of complex calculations over the entire pixels of the input image and results in vast computing power requirements. In contrast to the vehicular and robotic sentry applications, object recognition in mobile intelligent robots is especially challenging because the robots are equipped with a limited power supply due to their small physical dimensions. Previous implementations of intelligent robots [6]–[10] realized their intelligence processing using power-hungry general purpose processors. For example, Pioneer 3DX [9], which uses a state-of-the-art laptop at the time of publication, consumes more power in its embedded computer than its mechanical movement. Due to the significant power consumption of the computing system [6], [8], [9], the operable time of the robots is limited to tens of minutes. Even worse, the performance achieved by the general purpose processor is insufficient, and it is a limiting factor in improving robot motion speed. Such high-performance requirements with low-power constraints led us to design a single chip, application-specific processor with power efficient features.

The most widely used object recognition algorithm in mobile intelligent robots is scale invariant feature transform (SIFT) [6], [7], [15], [17], [18]. As the first step to design an object recognition processor, we analyzed the SIFT [15] computation to determine the optimal hardware architecture, and we concluded that the SIFT computation can be efficiently accelerated by exploiting task-level parallelism as well as data-level parallelism [12]. To efficiently support task-level parallelism, we adopted a multi-processor architecture instead of a heavy super scalar processor design supporting simultaneous multi-threading (SMT) [19], [20], because the former approach is more advantageous for low-power designs. For a given performance requirement, the operating frequency and power supply voltage of the chip can be lowered using a larger number of processors [21] and the independent power gating of each processor is also possible. We implemented an 81.6 GOPS object recognition processor [11]–[14] incorporating ten processing elements (PEs) for task-level parallelism. To support data parallelism as well, each PE is equipped with a single instruction multiple data (SIMD) instruction that accelerates the image filtering operations. In the proposed processor, an ARM-based RISC and eight visual image processing (VIP) memories are also integrated for PE management and data communication buffers, respectively. In addition, the VIP memories further accelerate the SIFT computation by replacing the local maximum pixel location search operation with a single read operation.

As will be shown in Section II, it is noticeable that each SIFT task requires data from its former stage only. Therefore, efficient computation of the SIFT is achieved by organizing the task-level pipeline with proper mapping of the SIFT tasks into the ten PEs. Supporting the pipelined execution of the SIFT tasks requires a sophisticated communication scheme between the ten PEs and eight VIP memories because the SIFT tasks exchange a large amount of data simultaneously. We proposed the memory-centric network-on-chip (NoC) [11]–[14] to facilitate the pipelined task execution in the proposed object recognition processor.

Manuscript received March 28, 2007; revised December 03, 2007. First published February 03, 2009; current version published February 19, 2009.

The authors are with the School of Electrical Engineering and Computer Science, Korea Advance Institute of Science and Technology (KAIST), Daejeon 305-701, Korea (e-mail: donghyun53@eeinfo.kaist.ac.kr).

Digital Object Identifier 10.1109/TVLSI.2008.2011226

Communication buffers for the producer and consumer tasks are managed by the memory-centric NoC, which also provides memory transaction control. As a comprehensive extension to our previous works [11]–[14], this paper introduces a memory-centric NoC managed coherence and consistency protocol for efficient pipelined task execution, which was not covered in our previous works. Also, elaboration of the processor architecture, programming model, and operation of the memory-centric NoC are covered in more detail.

The remainder of the paper is organized as follows. In Section II, the SIFT-based object recognition is described as a target application. The desirable hardware features for efficient SIFT calculation are also discussed. Next, the overall architecture of the proposed processor is presented in Section III. The advantages of adopting the SIMD PE and the VIP memory are also briefly explained. Section IV introduces the coherence and consistency scheme of the memory-centric NoC devised for efficient pipelined task execution on a multi-core processor. The programming model, operation of the memory centric NoC, and resulting advantages are also explained. The performance evaluations in Section V quantify the contributions of the SIMD instruction, VIP memory, and the memory-centric NoC in achieving high-performance and power-efficient object recognition. After the implementation results are reported in Section VI, conclusions are made in Section VII.

II. Target Application Analysis

In this section, SIFT [15], the object recognition algorithm used in this work, is described as a target application. The decisions regarding the hardware architecture and features are made based on the characteristics of the data processing and transactions in the SIFT algorithm.

A. Overall Processing Flow

Fig. 1 shows the overall flow of the SIFT computation divided into the: 1) key-point localization and 2) descriptor vector generation stages. For the key-point localization, Gaussian filtering with varying coefficients is performed repeatedly on the input image. Then, subtractions among the filtered images are executed to yield the difference of Gaussian (DoG) images. By performing the DoG operation, the edges of different scales are detected from the input image. After that, the locations of the key-points are decided by finding the local maximum pixels traversing a 3×3 search window over all DoG images. The pixels with a local maximum value greater than a given threshold become the key-points. Once the key-point locations are selected, the DoG value of each key-point location is compared with the DoG values at the same location of the upper and lower DoG images, as shown in the third step of Fig. 1(a). The scale of each key-point is decided by identifying the one with the maximum DoG value among three adjacent DoG images.

The next stage of the key-point localization is the descriptor vector generation. For each key-point location, $N \times N$ pixels of input image are sampled first, and then the gradient of the sampled image is calculated. The sample size N is decided according to the scale of each key-point. Finally, a descriptor vector is generated by computing the orientation and magnitude histograms over $M \times M$ sub regions of the sampled input image.

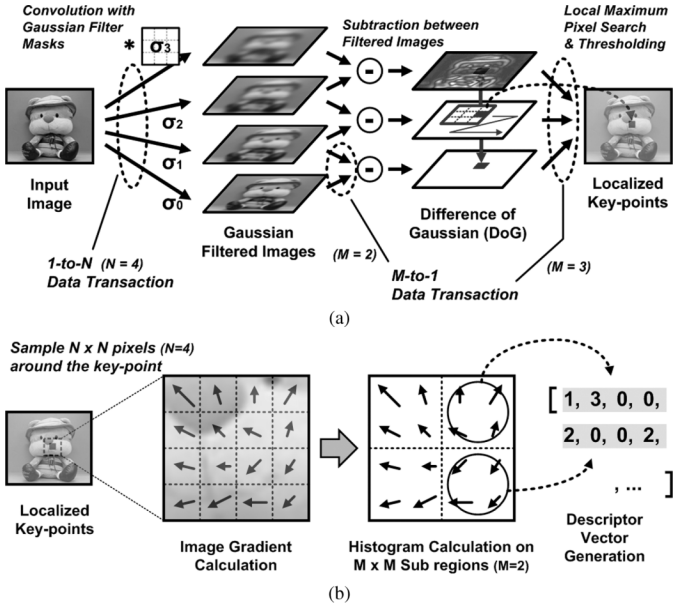


Fig. 1. Overall flow of the SIFT computation. (a) Key-point localization. (b) Descriptor vector generation.

The dimension of the descriptor vector is decided in accordance with M. In this paper, the typical value of N is in the range of 8 $\sim$ 32 in proportion to the scale of each key-point, and we used $M = 2$ to generate the descriptor vectors. The M and N values are decided by iterative MATLAB-based simulations regarding both the computation complexity and correct object recognition ratio. In addition, the number of key-points detected in each object is about a few hundred.

B. Exploiting Parallelism

The massive computation performed in the SIFT algorithm motivates the exploitation of parallel processing. For example, the Gaussian filtering task is repeated for every filter coefficient, and each Gaussian filtering repeats the convolution operation over all pixels in the image. Furthermore, some types of parallelism are revealed in these repeated tasks and operations.

In more specific terms, the Gaussian filtering tasks for different filter coefficients can be performed independently in multiple PEs. In other words, each concurrent task can be assigned to a dedicated PE, thus, multiple tasks are performed in parallel. This type of parallelism is called task-level parallelism. Meanwhile, a single instruction is applied repeatedly to the all pixels in each Gaussian filtering task, which can be accelerated by an SIMD structure exploiting data-level parallelism.

The advantages of exploiting task-level parallelism are more apparent in the descriptor vector generation stage. The processing of the descriptor vector generation stage starts with fetching $N \times N$ pixels from the input image, and the amount of subsequent computation depends on the number of pixels. Because N varies according to the key-points, every computation workload also varies for each key-point. In this situation, utilizing the multiple PEs in a SIMD fashion, which broadcasts a single program to all PEs, is not a practical solution because some PEs with small workloads will waste processing cycles to be synchronized with other heavily loaded PEs. Therefore,

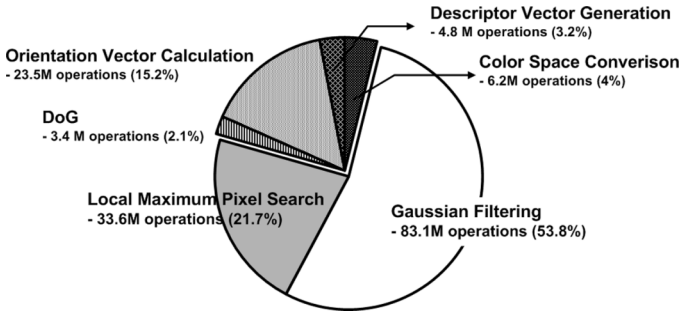


Fig. 2. Breakdown of required computation for each SIFT tasks.

independent task executions for each PE in the multiple instruction streams multiple data (MIMD) mode is adequate for the descriptor vector generation stage. However, the SIMD data path is also applicable for each PE to accelerate single tasks running on each PE.

The architecture of the object recognition processor in this paper is designed to exploit both the task-level and data-level parallelism. The multi-processor architecture with NoC and the number of PEs in the system are determined considering the characteristics of the tasks and the number of concurrent tasks found in the SIFT algorithm. In addition, each PE is designed to support four-way SIMD instructions facilitating data-level parallel processing in each task execution.

C. Hardware Acceleration

Fig. 2 shows the amount of computation required for each SIFT task when it is performed for a 320×240 pixel image using ARM instructions, excluding the overheads of instruction fetching, managing data structures, and loop iterations. It is clear that the Gaussian filtering and local maximum pixel search operation are the two most demanding tasks. Gaussian filtering is accelerated by task-level and data-level parallelism as described in Section II-B. For the local maximum pixel search operation, we decided to implement a special hardware accelerator.

The local maximum pixel search task spends vast amounts of cycles to load the nine pixels of the 3×3 search window and the following nine comparisons of the pixels. We implemented a special memory, a VIP memory, to read out the address of the local maximum pixel in response to the center pixel address input from the search window. The detailed architecture and operations of the VIP memory are described in Section III.

D. Data Transaction

Each task of the key-point localization consumes and produces a large amount of intermediate image data, and the data should be transferred between the tasks. The data transaction between tasks is performed through an on-chip interconnection. Therefore, the NoC design should account for characteristics of the data transaction. Below, we note two important characteristics of the data transaction in the key-point localization stage.

The first point is regarding the data dependency between the tasks. As illustrated in Fig. 1(a), the processing flow is completely pipelined, thus, data transactions only occur between two adjacent tasks. This fact impacts the design of the coherence and consistency scheme in the NoC as described in Section IV.

The second point is concerning the number of initiators and targets in the data transaction. In the multi-processor architecture, each task will be mapped to a group of processors, and the number of processors involved in each task can be adjusted to balance the execution time. For example, the Gaussian filtering task in Fig. 1(a), having the highest computational complexity due to the 2-D convolution, could use four processors to filter operations with different filter coefficients, whereas all of the DoG calculation is executed on a single processor. Due to the flexibility in task mapping, the resulting data of one processor is transferred to multiple processors of subsequent task or the results from multiple processors are transferred to one processor. This implies that the data transaction will occur in the forms of not only 1-to-1 but also 1-to-N and M-to-1, as shown in Fig. 1(a), where the N and M may vary for each task according to the task mapping. Furthermore, the NoC design is tailored to support this type of data transaction efficiently, which is described in Section IV in detail.

III. Processor Architecture and Hardware Components

In this section, the overall architecture of the proposed processor and hardware components for fast object recognition are briefly described.

A. Processor Architecture

The overall architecture of the proposed object recognition processor is shown in Fig. 3. The main components of the proposed processor are ten SIMD PEs, eight VIP memories, and an ARM-based RISC processor. The RISC processor controls the overall operation of the proposed processor by initiating the execution of each PE. After initialization, each PE fetches and executes an independent program for the parallel execution of multiple tasks. The eight VIP memories provide communication buffers between the PEs and accelerate the local maximum pixel search operation. Interconnection between the ten PEs and eight VIP memories is provided by the memory-centric NoC. The memory-centric NoC is composed of five crossbar switches, four channel controllers, and a number of network interface modules (NIMs). The topology of the memory-centric NoC is decided by considering the characteristics of the on-chip data transactions. For efficient support of the 1-to-N and M-to-1 data transactions shown in Fig. 1(a), using the VIP memory as a shared communication buffer is amenable to remove the redundant data transfers when multiple PEs read the same data. Because the data flows through the pipelined tasks, each PE accesses only a subset of the VIP memories to receive the source data from its former PEs and send the resulting data to its following PEs. This results in localized data traffic, which allows tailoring of the NoC topology for low power and area reduction. In the previous work of our research group [22], we concluded that a hierarchical star topology is the most efficient for reducing the area and power consumption when interconnecting a few tens of on-chip modules with localized traffic. Therefore, the memory-centric NoC is configured in a hierarchical star topology instead of a regular mesh topology. By adopting a hierarchical star topology NoC, the architecture of the proposed processor is able to be determined so that average

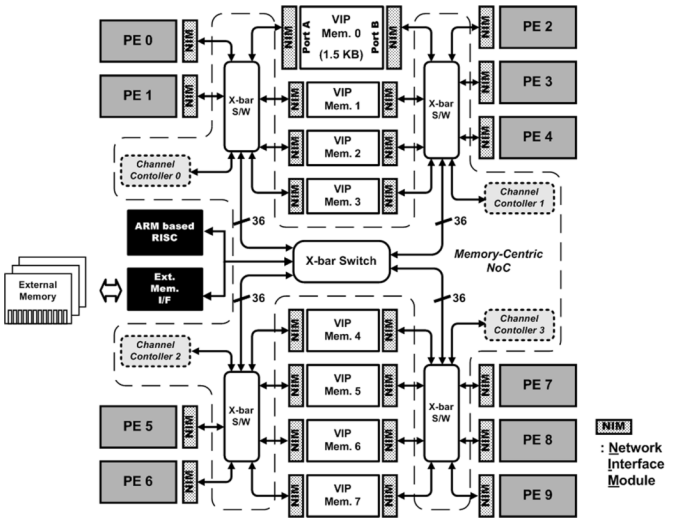


Fig. 3. Architecture of the proposed object recognition processor.

hop counts between each PE and the VIP memories are reduced at the expense of a large direct PE-to-PE hop count, which is fixed to 3. This is also advantageous because most data transactions are performed between the PEs and VIP memories, and direct PE-to-PE data transactions rarely occur. In addition, the VIP memory adopts dual read/write ports to facilitate short distance interconnections between the ten PEs and eight VIP memories. The NIMs are placed at each component of the processor to perform packet generation and parsing.

B. SIMD PE

Fig. 4(a) shows a block diagram of the SIMD PE that has five pipeline stages supporting a single thread execution. The PE contains a 512 B instruction cache, a 512 B local memory, 16 general purpose registers (GPRs), and 12 coefficient dedicated registers (CDRs). The size of the local memory is determined by inspecting the working set size of the PEs using a software simulation model. Since most of the processed data are written directly to the VIP memories instead of being stored in the local memory, 512 B is sufficient for computing the SIFT with multiple PEs in a pipelined task execution mode. To implement low-overhead reciprocal, square-root, and division operations, a modified version of the logarithmic arithmetic logic unit (ALU) [34] is integrated with a conventional ALU, sharing a configurable carry save adder (CSA) tree. Through MATLAB-based simulations, we verified that the $\sim$0.1% of error that arises from the logarithmic ALU does not degrade the ratio of correct object recognition.

To accelerate the Gaussian filtering tasks, special instructions for image filtering are implemented in the SIMD PE. The instructions are sum of dot product (SDP) and load extension (LE). For the SDP instruction, 12 CDRs to store the filter coefficients are added. As shown in Fig. 4(b), the SDP instructions calculate the 8-bit four-way SIMD multiplications and four subsequent additions including accumulation in a single cycle. The SDP instruction brings one operand from the GPR and the other operand from the CDR. The LE instruction is a combination of the 8-bit shift and byte load operation, which is designed to support a seamless filter window movement over image data. Once the filter coefficients and image data are stored in the CDR and GPR, respectively, replacing the pixels of the GPR using the LE instruction is equivalent to moving the filter mask over the input image when the SDP instruction follows the series of the LE instructions. By performing the SDP instruction, each PE performs eight operations in a single cycle and its operation frequency is designed to be 200 MHz. As a result, the ten PEs contribute to the 16 GOPS of total performance.

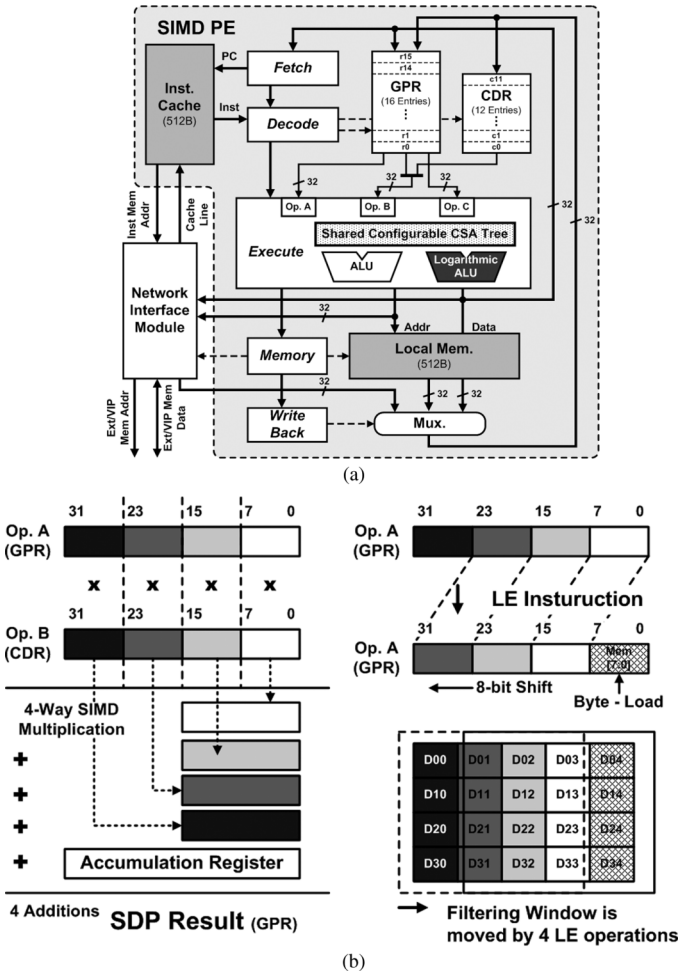


Fig. 4. PE architecture and image filtering instructions. (a) PE block diagram. (b) Operation of the SDP and LE instructions.

C. Visual Image Processing Memory

The VIP memory is specially designed to find the location of the local maximum pixel inside the 3×3 search window in a single cycle. As shown in Fig. 5(a), searching for the local maximum pixel location requires $29 \sim 53$ cycles in the ARM-based RISC. By replacing this time consuming computation with a single read operation, large performance gains are obtained. In addition to the normal memory operations, the function of the VIP memory is to read out the address of the local maximum pixel inside the 3×3 search window in response to the address input of the center pixel in the window. Since the local maximum pixel search operation requires 41 cycles on an average, the eight VIP memories operating at 200 MHz gives a 65.6 GOPS performance gain.

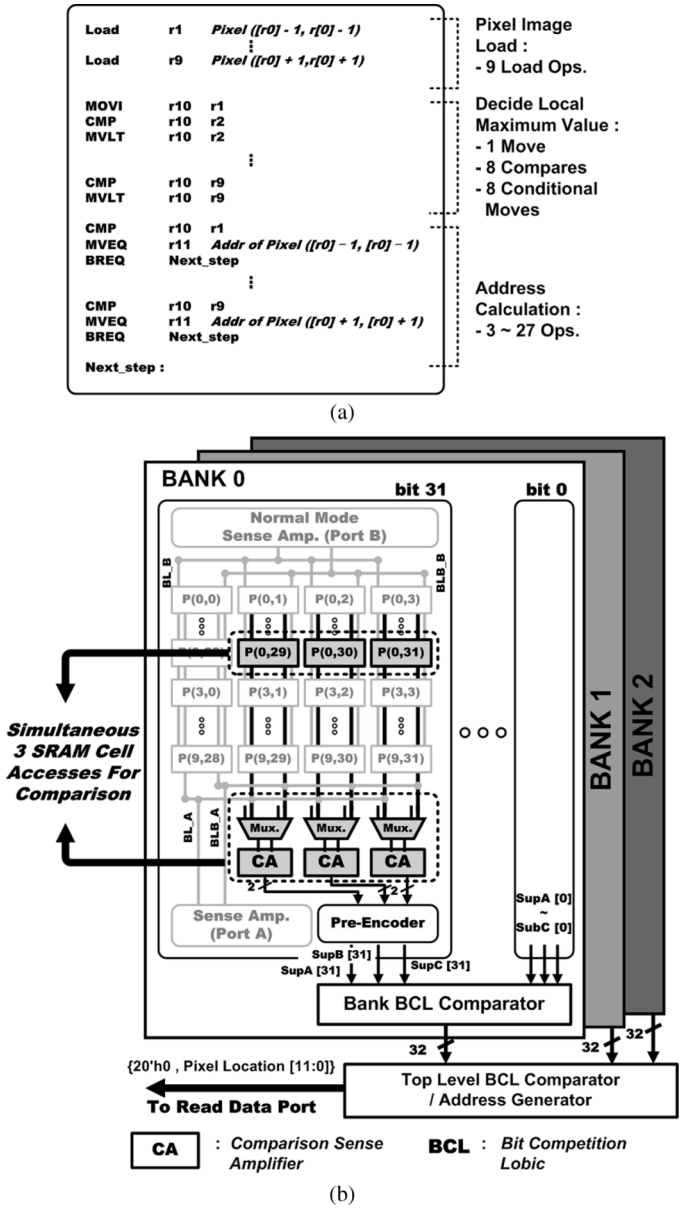


Fig. 5. Visual image processing memory. (a) Cycle counts of local maximum location search operation. (b) Architecture of the visual image processing memory.

The overall architecture of the VIP memory is shown in Fig. 5(b). In the VIP memory, 12 rows by 32 columns of 32-bit pixels are stored, which result in a total of 1.5 kB capacity. To compare the nine pixel values in one cycle, every row is interleaved into three banks so that the bank number assigned for each row is decided using a modulo-3 operation. Three pixels in the same row are compared first inside the bank, and then the results from the three banks are compared again to find the local maximum pixel from among the nine pixels. The address of the local maximum pixel is automatically generated according to the comparison result via the address generation unit. At each bank, three comparison amplifiers (CAs) are integrated into every four bit line pairs to read the three pixel values simultaneously. The transistor size of the CA is smaller than a normal sense amplifier because it does not drive long capacitive DB lines. To minimize the area overhead of the comparison logic in the memory, a bitwise competition logic (BCL) [23] is also devised. By adopting the BCL, the transistor count of the comparator is reduced from 2400 to 536 when compared with the conventional adder based comparator. More details of the VIP memory are described in [14].

IV. Memory-Centric NoC

The basic operation of the NoC is to facilitate high-bandwidth on-chip data transactions. However, as an interconnection structure of a multi-core processor, supporting frequent shared data transactions, the NoC must handle the coherence and consistency issue of the multi-core processor. In this section, previous researches concerning the coherence and consistency of the shared memory based multi-processor system are briefly reviewed. Then, we propose a coherence and consistency scheme managed by the memory-centric NoC that is tailored for efficient pipelined task execution in the proposed processor. The coherence and consistency protocols and underlying mechanisms implemented in the memory-centric NoC are explained first. Then, the resulting programming model and internal operation of the memory-centric NoC are also described.

A. Memory-Centric NoC Coherence and Consistency

A coherence scheme manages how the correct value of shared variable is read when multiple copies of the variable may exist, and the consistency scheme defines the moment of reading the correct shared variable for each task. Two basic and traditional coherence schemes are bus-snooping and directory-based coherence protocols [24]–[26]. The bus-snooping protocol is not suitable for large-scale multi-core processor due to the poor scalability of the shared bus. Even though the directory-based protocol is developed for high-performance interconnections such as an on-chip network, the overhead of managing large directories increases intolerably when bulky data transactions occur between a number of PEs. In the past decade, transactional memory [27], [28] and thread level speculation [19], [29], [30] have been proposed as more advanced coherence and consistency schemes that take advantage of speculative execution. However, a strong data dependency between adjacent stages of the pipelined task makes it difficult to obtain performance gains from the speculation of shared variables.

Together with the widespread adoption of NoCs in the MP-SoC design field, there have been studies concerning NoC architectures together with the consistency and coherency schemes [35]–[38]. Bolotin *et al.* [35] proposed to embed the priority support inside their Vanilla NoC to differentiate between short control signals and long data messages. The low-latency control signals contribute to reducing the average L2 cache access latency by making the L2 caches respond earlier to coherence packets, whereas Eisley *et al.* [36] proposed a joint optimization of the NoC and directory based coherence protocol. The average memory access latency is reduced by embedding directories within each NoC router so that each local node accesses the nearest copy of a cache line instead of the farther home directory. In the case of [37], a software-based solution that relies on the separate management

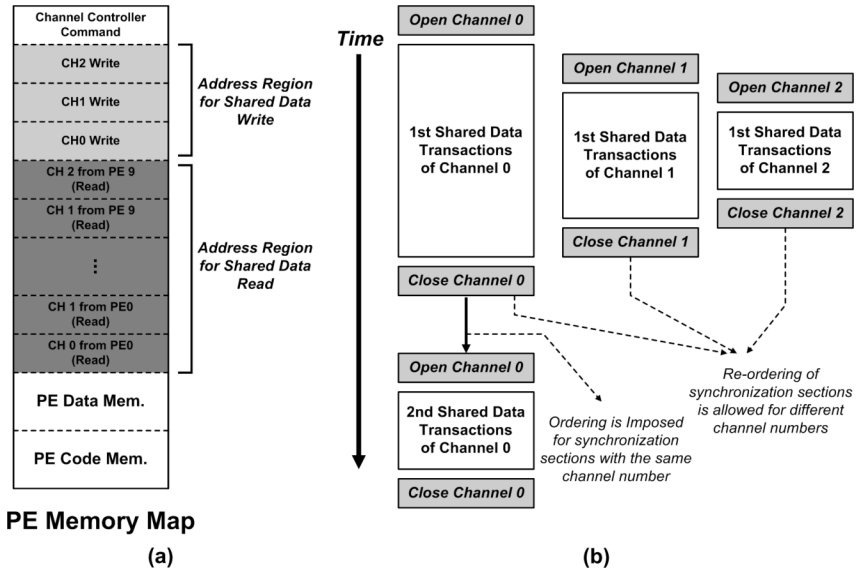


Fig. 6. (a) Fixed address regions for the read/write of shared data transaction and (b) ordering rule among synchronization sections.

of non-cacheable shared data and cacheable local data was proposed to reduce the overhead of implementing an on-chip coherence scheme in an NoC-based multi-core processor. In addition, the speed up of coherence message transactions, exploiting the knowledge of physical interconnections of an NoC, was also proposed [38]. Although the studies of [35]–[38] were performed regarding NoC-based architectures, they mainly focused on general purpose applications running with the distributed L2 caches. However, the memory-centric NoC aims to support the pipelined execution of multiple tasks with dynamically managed on-chip shared memories that are transparent to the PEs in the proposed processor.

Recently, streaming consistency [31] was proposed focusing on stream processing domains. In the streaming consistency model, the shared data are exchanged through communication buffers between explicit *acquire* and *release* synchronization sections. Efficient pipelined task execution is achieved by allowing the reordering of the synchronization sections that are associated with the different communication buffers. In the streaming consistency, the communication buffers were managed using the C-HEAP protocol [32], [33]. The drawbacks of the C-HEAP protocol in computing task pipelined object recognition are as follows. The C-HEAP protocol only supports first-input–first-output (FIFO)-based data transactions and results in inefficiencies when data access patterns are different between the producer and consumer tasks. In addition, the C-HEAP protocol requires explicit management of the read/write pointer in the FIFOs, and either polling or interrupt is required before accessing an available shared memory. Supporting 1-to-N data transactions also has an overhead of copying the administration information of the FIFO for each consumer task. In the latter part of this section, we propose the memory-centric NoC managed coherence and consistency schemes for efficient pipelined task execution on the proposed processor.

1) Coherence: At each PE of the proposed object recognition processor, the fixed address region is predefined for the read/write accesses of the shared data. As shown in Fig. 6(a), the write address region is divided into three write channels, and the read address region is divided to distinguish both the channel number and the producer PEs. For each transaction, the producer and consumer PEs are defined, and then one VIP memory is assigned as a data transaction buffer for the PEs involved. To maintain data coherence, the memory-centric NoC forwards the read/write accesses of the shared data address region to the assigned VIP memory. Until one of the available VIP memories is assigned by the memory-centric NoC, any access to the address region of the shared data read/write is blocked for both producer and consumer PEs. A limitation of this mechanism is that caching is not allowed for the shared data address region, but this does not degrade the overall performance because the shared data delivered between pipelined tasks are rarely reused.

2) Consistency: In the memory-centric NoC managed consistency, all shared data transactions are performed inside the explicit synchronization section. In a producer PE, *open* and *close channel* commands are sent to the channel controller of the memory-centric NoC at the start and end of the shared data transaction, respectively. Correspondingly, consumer PEs should assert the *end channel* command at the end of reading shared data from the producer PE. Reordering among the synchronization sections, which are associated with different communication buffers (i.e., VIP memories), is allowed for the memory-centric NoC managed consistency, similar to the stream consistency model [31]. A maximum of three synchronization sections are able to progress simultaneously for each PE using different channel numbers; imposing ordering between the former and subsequent synchronization sections is also possible using the same channel number. This ordering rule of the synchronization sections is shown in Fig. 6(b). Inside the synchronization section, the read accesses from the consumer PEs to an empty entry of the VIP memory, which is not yet written by the producer PE, are automatically blocked by the memory-centric NoC to prevent the consumer PEs reading false data.

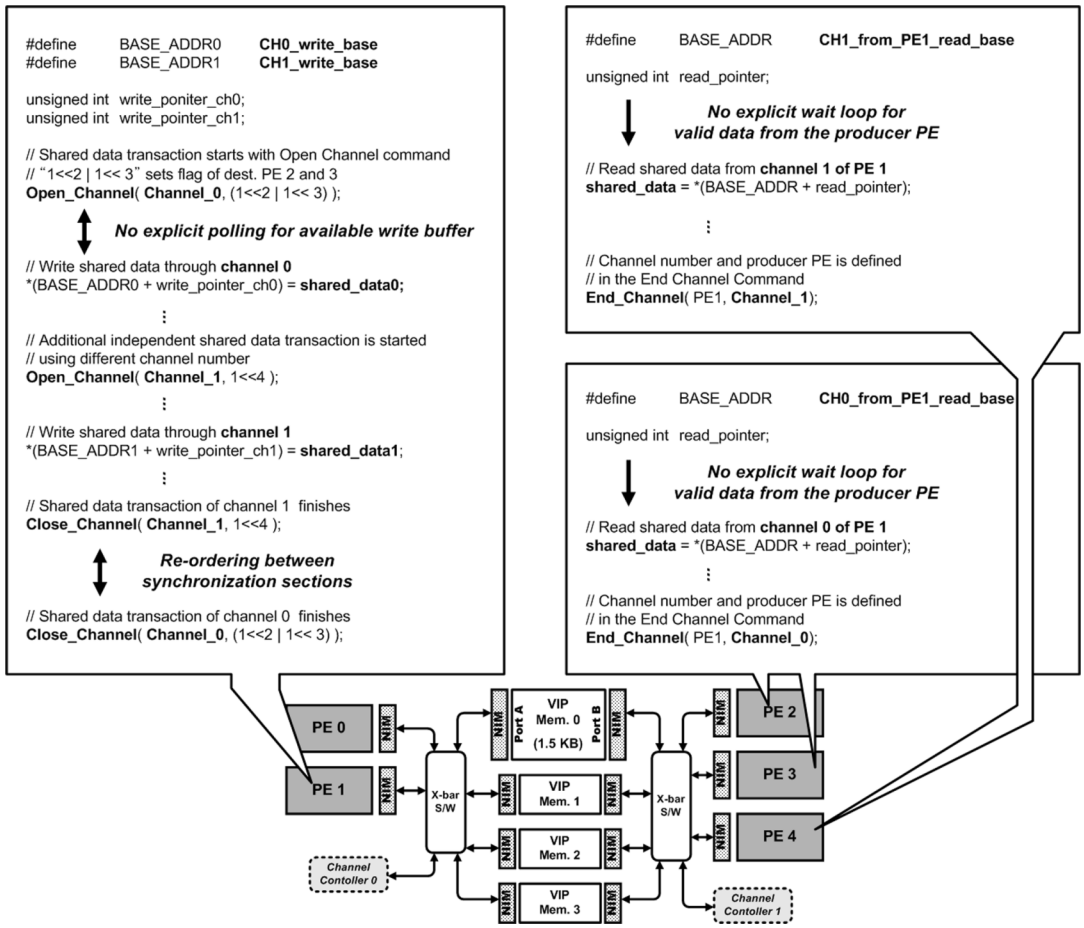


Fig. 7. Programming model of the memory-centric NoC.

The uniqueness of the memory-centric NoC is that it supports dynamic reconfiguration of the on-chip networks so that fixed address regions, instead of simple FIFO queues, are used for the shared data transactions between the PEs. This technique removes the overhead of obtaining dynamic memory pointers for shared data transactions and also reduces the processor activity. Compared with the stream consistency model [31], the memory-centric NoC is more efficient for pipelined task execution. By the hardware support of the memory-centric NoC, the overhead of polling available memory and updating the read/write pointer of the communication buffers is removed. In addition, 1-to-N data transactions are supported with a very low overhead over 1-to-1 data transactions by allowing the manipulation of the routing lookup tables (LUT) in the NIMs. Based on these underlying coherence and consistency rules, the programming model of the memory-centric NoC is explained in Section IV-B.

B. Programming Model

In the memory-centric NoC, all shared data transactions are explicit. A shared data transaction is initiated by a producer PE writing an *open channel* command to the channel controller. In the *open channel* command, a channel number and a list of consumer PEs are specified. After that, the producer PE writes the data to be stored in the write address region of Fig. 6(a) that corresponds to the channel number specified in the *open channel* command. In contrast to the stream consistency [31] model, a buffer availability check is not necessary because it is automatically performed by the memory-centric NoC, thus the buffer is transparent to the software. When there is an available memory in the eight VIP memories, the writes of the shared data are directed to the assigned VIP memory. Access to the write address region is blocked if there is no available VIP memory. For each consumer PE, reading the shared data is initiated by simply accessing the read address region of Fig. 6(a), which indicates the corresponding producer PE and channel number. No waiting loops are required to read the valid shared data from the producer PE because the memory-centric NoC also automatically blocks each read access to the read address region until the corresponding write operation by the producer PE is completed. At the end of the shared data read, the consumer PE should write an *end channel* command to the channel controller. In summary, a 1-to-N shared data transaction starts with a producer PE writing an *open channel* command to the channel controller. After that, it finishes with the channel controller receiving a *close channel* command and N *end channel* commands from the producer PE and N consumer PEs, respectively. Fig. 7 illustrates an example of a pseudo code where PE 0 delivers shared data through two write channels. Channel 0 is used to send the shared data to PE 2 and PE 3, and channel 1 is used for PE 4.

In the memory map of Fig. 6(a), the same amount of address regions are assigned to each shared data read/write channel. The

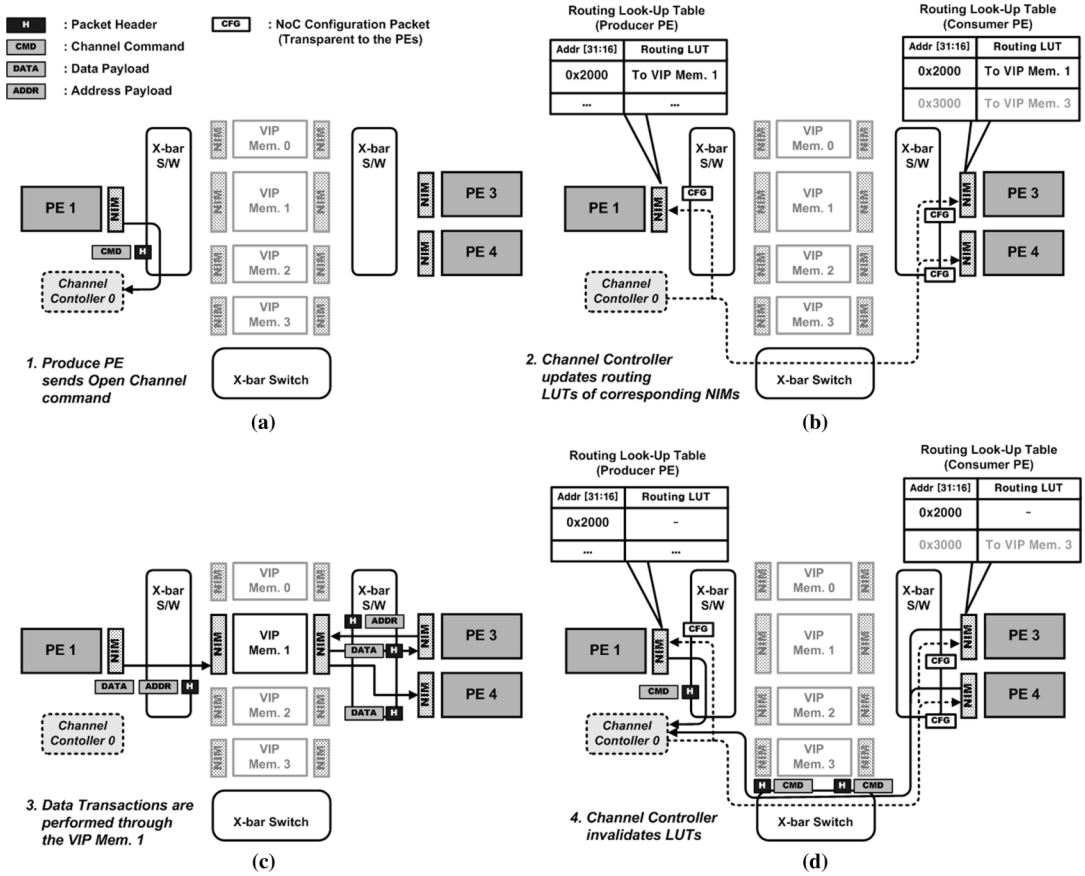


Fig. 8. Communication buffer management operation of the memory-centric NoC.

20 MSBs of the address in the PE memory map are used as the base address for the routing administration of the memory-centric NoC. The remaining 12 LSBs are used as a common pointer to the shared data in the VIP memory for both the producer and consumer PEs. This addressing scheme removes the limitation of imposing sequential access of shared data when FIFO buffers are used, instead of randomly accessible memory as in the stream consistency [31] and C-HEAP protocol [32], [33].

C. Operation

In this subsection, the internal operation of the memory-centric NoC that provides the coherence and consistency schemes is described. The operation of the memory-centric NoC is divided into two parts. The first part manages the utilization of the communication buffers, i.e. the VIP memories, between the producer and consumer PEs. The other part supports the memory transaction control after the VIP memory is assigned for the shared data transactions. The former operation removes the overhead of polling available buffer spaces and the latter one reduces the overhead of waiting for valid data from the producer PE. Based on the memory-centric NoC operation, the advantages in supporting efficient 1-to-N and M-to-1 data transactions are also explained.

Communication Buffer Management: The overall procedure of the communication buffer management in the memory-centric NoC is depicted in Fig. 8. Throughout the procedure description, we assume that PE 1 is the producer PE and PEs 3 and 4 are consumer PEs. As mentioned in Section IV-B, the operation of the memory-centric NoC is initiated by PE 1 writing an *open channel* command to the channel controller connected to the same crossbar switch [see Fig. 8(a)]. The *open channel* command is a simple memory mapped write and transfers using a normal packet. In response to the *open channel* command, the channel controller reads the global status register of the VIP memories to check the utilization status. After selecting an available VIP memory, the channel controller updates the routing LUTs in the NIMs of PEs 1, 3, and 4, so that accesses to the shared data address region are directed to the assigned VIP memory [see Fig. 8(b)]. The routing LUT update operation is performed by the channel controller sending the configuration (CFG) packets, which are transparent to the PEs. At each PE, access to the shared data address region is blocked until the routing LUT update operation finishes. Once the VIP memory assignment is completed, a shared data transaction is executed using the VIP memory as a data communication buffer. Read and write accesses to the VIP memory are performed using normal read/write packets that consist of an address and/or data fields [see Fig. 8(c)]. After the shared data transaction completes, PE 1 sends a *close channel* command, and PEs 2 and 3 send *end channel* commands to the channel controller. After that, the channel controller sends CFG packets to the NIMs of PEs 1, 3, and 4 to invalidate the corresponding routing LUT entries and to free up the used VIP memory [see Fig. 8(d)].

From the communication buffer management operation, efficient 1-to-N shared data transaction support is clearly visible.

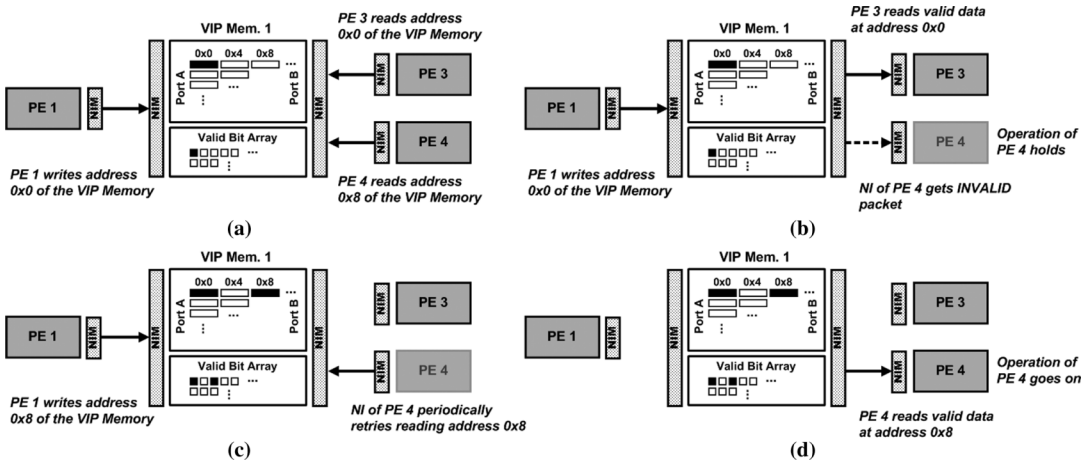


Fig. 9. Memory transaction control scheme of the memory-centric NoC.

Compared with the 1-to-1 shared data transaction, the required overhead is only sending additional $(N-1)$ CFG packets at the start/end of a synchronization section. In addition, an M-to-1 data transaction is also easily achieved by the consumer PE simply reading M address regions for the shared data read in the memory map of Fig. 6(a).

Memory Transaction Control: In the memory-centric NoC managed consistency scheme, no explicit waiting loop is necessary to prevent consumer PEs reading the shared data too early before the producer PE writes valid data. In this subsection, the operation of the memory transaction control that implicitly manages the synchronization of shared data transactions is described.

To support the memory transaction control, the memory-centric NoC tracks every write access to the VIP memory from the producer PE after the VIP memory is assigned for shared data transactions. This is realized by integrating a valid bit array and valid check logic inside the VIP memory. In the VIP memory, every word has a 1-bit valid bit entry that is dynamically updated. The valid bit array is initialized when a processor resets or at the end of shared data transactions. By the write access from the producer PE, the valid bit of the corresponding address is set to HIGH. When an empty memory address with a LOW valid bit is accessed by the consumer PEs, the valid check logic asserts an INVALID signal to prevent reading false data. Up to nine entries of the valid bit array can be checked simultaneously to support the local maximum pixel search operation, which reads the nine pixels in the 3×3 search window in a single cycle.

Fig. 9 illustrates the overall procedure of the proposed memory transaction control. We assume again that PE 1 is the producer PE, and PEs 3 and 4 are consumer PEs. Because the memory-centric NoC supports arbitrarily ordered as well as sequentially ordered shared data transactions, two consumer PEs are able to read the shared data with different access orders. In the example data transaction, PE 3 reads the shared data at address 0×0 and PE 4 reads the shared data at address 0×8, whereas PE 1 writes the valid data at only address 0×0 of the VIP memory [see Fig. 9(a)]. Because the valid bit array has a HIGH bit for the address 0×0 only, the NIM of PE 4 obtains an INVALID packet instead of normal packets with valid data [see Fig. 9(b)]. Then, the NIM of PE 4 periodically retries reading valid data at address 0×8 until PE 1 also writes valid data at address 0×8 [see Fig. 9(c)]. Meanwhile, the operation of PE 4 is in a hold state. After reading the valid shared data from the VIP memory, the operation of the PE continues [see Fig. 9(d)].

The advantages of the proposed memory transaction control are reduced NoC traffic and PE activity, which contribute to a low-power operation. For consumer PE polls on the valid shared data, four flits—two flits for a read request and two flits for a read response—traverse the NoC at every iteration of polling, and the PE should execute the compare and branch instructions repeatedly until the valid shared data is read. In the memory-centric NoC managed memory transaction control, three flits—two flits for a read request and one flit for an INVALID notification packet—traverse the NoC, and the consumer PE is in an idle state. The amount of reduction in flit transactions is reported in Section V. In the perspective of shared memory utilization, the other advantage of memory transaction control is that it enables fine grained data transactions between the producer and consumer PEs. When shared data are delivered using conventional polling-based synchronization, the unit of data transaction should be sufficiently large to avoid the overhead of checking validity of every shared variable in the program execution. In this case, double buffering is also necessary to hide the latency of producing bulky data. In the memory-centric NoC, however, a consumer PE is able to read the shared data from a producer PE as soon as it is computed, without waiting until a certain amount of shared data is accumulated in the shared memory region, which is usually divided into front and back buffers. As a result, the memory-centric NoC improves the efficiency of memory utilization by removing the overhead of double buffering.

V. Performance Evaluations

This section quantifies the performance improvement in computing the SIFT algorithm, which is obtained by active utilization of the SIMD PE, VIP memory, and the memory-centric

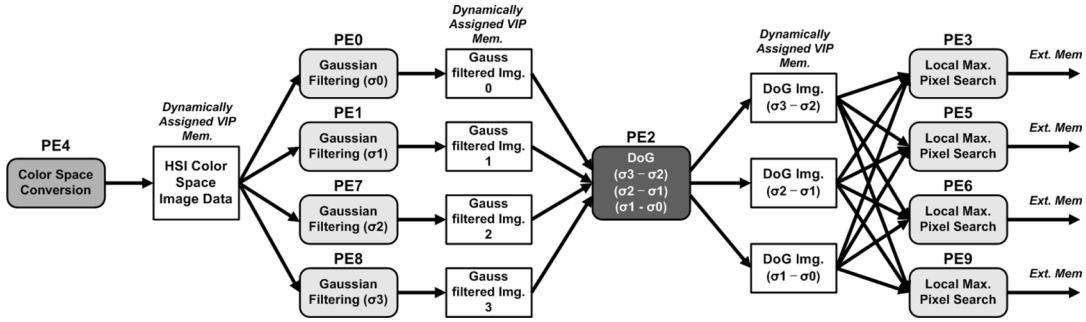


Fig. 10. Mapping of the SIFT tasks for the *Pipelined Task* execution mode.

NoC. The simulation setup is briefly described first, and the advantages of pipelined task execution over sequential task execution with data parallelism are reported. The performance gains resulting from the SIMD PE and VIP memory are also presented together. After that, the effects of the memory transaction control in the memory-centric NoC with regard to the number of internal data transactions are discussed with traffic statistics of the memory-centric NoC.

A. Simulation Setup

For chip implementation of the proposed object recognition processor, we implemented a cycle accurate Verilog model of the proposed processor, and this is the base of the performance evaluations in this section. To show the advantages of pipelined task execution in computing the SIFT algorithm, the execution times of the key-point localization stage in Fig. 1(a) are measured for two different task mappings. We named these two mappings as *data parallel* and *pipelined task* modes. In the *data parallel* mode, all PEs of the proposed object recognition processor execute the same task simultaneously by partitioning the region of operation in the input image. However, each of the key-point localization tasks is computed in a sequential manner. For the *pipelined task* mode, all tasks of the key-point localization are properly mapped on the ten PEs of the proposed processor and are executed simultaneously in the pipeline. The input image is sequentially fetched and flows through the pipeline to yield the result data to be written in the external memory. In this performance comparison, the descriptor vector generation stage of Fig. 1(b) is excluded because it is only executed in a task-parallel manner and makes no difference to the execution time. The task mapping in the *pipelined task* execution mode is shown in Fig. 10. Since the Gaussian filtering and local maxim pixel search tasks require vast amounts of computation, four processors are assigned for each task. The arrows in Fig. 10 show the data flow between the PEs performing the assigned tasks. In addition, the color conversion task, which converts red-green-blue (RGB) color space to the hue-saturation-intensity (HIS) color space [15], is added considering the external video input of the verification system in Section VI.

Table I describes the simulation cases used for the performance evaluations. From case A to case F, the proposed hardware acceleration techniques, such as the SIMD instruction and VIP memory, are sequentially applied one by one to reveal the contributions of each technique. The proposed memory transaction control described in Section IV-C is only applicable to the simulation cases adopting the pipelined task execution mode, which utilize the memory-centric NoC.

TABLE I
SIMULATION CASES AND CORRESPONDING FEATURES

Cases	Execution Mode	LE Inst.	SDP Inst.	VIP Mem.	Mem. Tran. Ctrl Support
A	Data Parallel	X	X	X	N/A
B	Data Parallel	O	X	X	N/A
C	Data Parallel	O	O	X	N/A
D	Pipelined Task	O	O	X	X
E	Pipelined Task	O	O	O	X
F	Pipelined Task	O	O	O	O

B. Data Parallel Versus Pipelined Task Execution

For the six simulation cases in Table I, the execution times of performing the key-point localization stages for a 320×240 pixel image are compared in Fig. 11. The execution times for the 2.3 GHz Intel Core 2 Duo and 200 MHz ARM9 TDMI processors are compared to prove the superior performance of the proposed processor over the conventional state-of-the-art processors. The execution times of the six simulation cases are calculated using the cycle counts obtained from the cycle-accurate Verilog simulation and considering a 200 MHz operation frequency. To measure the execution times of the Core 2 Duo and ARM9 TDMI processors, the key-point localization stage is described in C-language and compiled with a conventional gcc and ARM compiler, respectively. The execution time of the Core 2 Duo processor is measured using a *times()* function supported by the gcc compiler, and the ARMulator instruction set simulator is used to obtain the execution time of the ARM9 TDMI processor.

By tracking the reduction in the execution time from cases A to F, the advantages of the proposed techniques are clearly represented. At first, the performance gain results from use of the LE instruction is relatively small $(A \rightarrow B)$. However, the SDP SIMD instruction reduces the execution time of the Gaussian filtering task drastically $(B \rightarrow C)$. The execution time is further reduced by switching to the *pipelined task* execution mode from the *data parallel* execution mode, because it reduces the

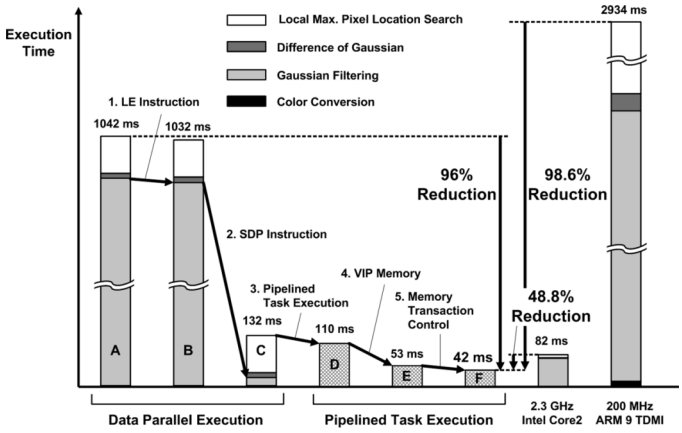


Fig. 11. Execution time comparisons for key-point localization stage of the SIFT computation.

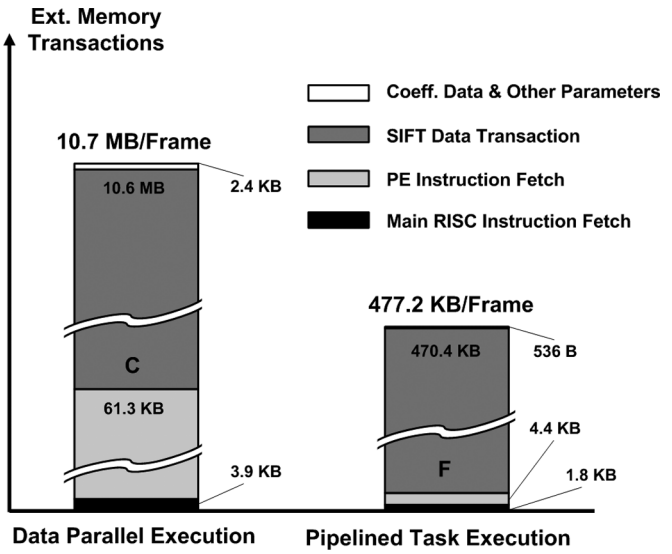


Fig. 12. Amount of external memory data transactions.

overhead of external memory transactions with long access latencies $(C \rightarrow D)$. After that, adopting the VIP memories in the task-level pipeline doubles the performance gain $(D \rightarrow E)$. The efficiency of the proposed memory transaction control in the memory-centric NoC is revealed by comparing the execution times of simulation cases E and F. The proposed object recognition processor achieves 48.8% and 98.6% reductions in execution times, compared with the Core 2 Duo and ARM9 processors, respectively.

The main advantage of the *pipelined task* execution mode is the reduced number of external memory transactions, and this is clearly visible in Fig. 12. The number of external memory data transactions is compared for simulation cases C and F. In the *data parallel* execution mode, the number of external memory transactions is approximately 22 times higher than the *pipelined task* execution mode because a vast amount of intermediate image data are fetched in and dumped out.

When it is possible to achieve a sufficient performance with a low-speed external memory, the power consumed in the external memory can be saved. The contribution of the memory-centric NoC for power-efficient object recognition is to facilitate

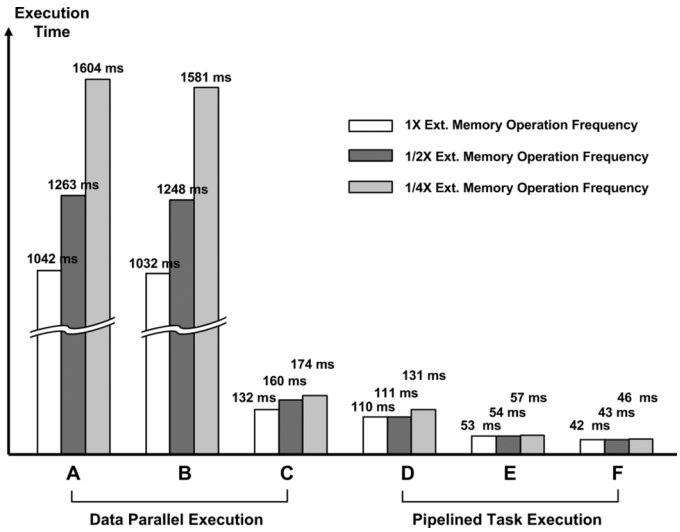


Fig. 13. Variations of total execution time according to the external memory bandwidth.

the pipelined task execution so that the overall object recognition performance is sustained insensitive to the external memory bandwidth degradation. This case is shown in Fig. 13. The total execution times of the six simulation cases are compared with respect to the different external memory operation frequencies. In the evaluation, the external memories that operate at 1, 1/2, and 1/4× speed of the proposed processor are used. For simulation cases A–C, which belong to the *data parallel* execution mode, the average performance degradations are 21.1% and 46.3% when 1/2 and 1/4× speed external memories are used, respectively. However, the performance degradations of the simulation cases that execute a *pipelined task* mode are much lower. In simulation cases D–F, the average performance degradations are 1% and 15.2% for the 1/2 and 1/4× speed external memories, respectively.

C. Effects of the Memory Transaction Control

In this subsection, the effects of the proposed memory transaction control on the internal traffic of the on-chip network are evaluated. Fig. 14 shows the number of internal data transactions for simulation case F. For comparison, the memory transaction control of the memory-centric NoC is also replaced with a polling-based shared data transaction. For the polling-based scheme, a producer PE writes the result data in the unit of the one VIP memory row and then updates the write pointer that is polled by the consumer processors. In Fig. 14, the two left bars for each PE represent the number of data transactions for the memory transaction control, and the two right bars for each PE are the results gained using the polling-based scheme. It is noticeable that the number of inbound flits increases when the memory transaction control is replaced with the polling-based shared data transaction. This is because the size of the response packet increases when the PE performs polling instead of the NIM. The number of the outbound flits also increases because the PEs send additional instructions fetch requests for the polling loops inserted. With the memory transaction control of the memory-centric NoC, the number of internal transactions is

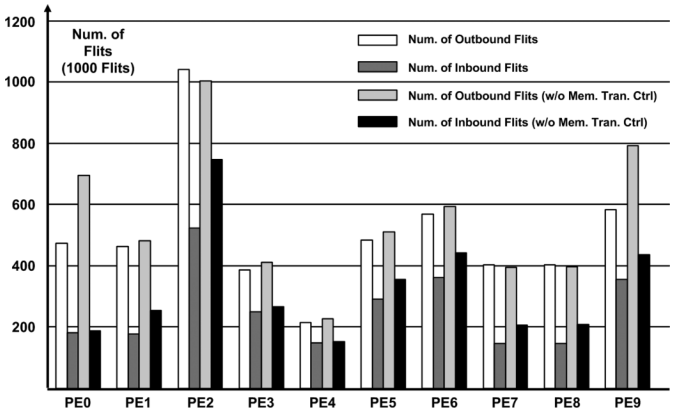
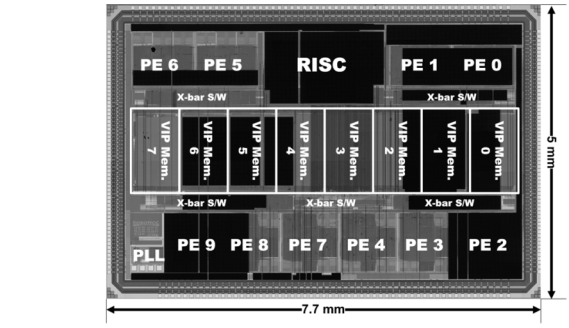


Fig. 14. Internal data transactions of the proposed processor.

reduced by 13% compared with the polling-based scheme. This also shows that the memory-centric NoC is beneficial for efficient computing of SIFT-based object recognition.

VI. Implementation Results

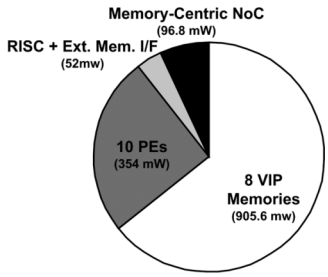
The proposed object recognition processor based on the memory-centric NoC is implemented using a 0.18-μm standard CMOS process technology. For the VIP memory implementation, we fabricated a full custom design because modification of the internal memory cell and sense amplifier structure was necessary. The other parts of the chip are designed using Verilog HDL, synthesized with logic cell libraries, and placed/routed using a Synopsys Design Compiler and Astro tool chain. The size of the implemented chip is 7.7 $\times$5 mm^2 and the operation frequency is 400 MHz for the memory-centric NoC and 200 MHz for the other parts of the chip. The chip photograph, summary of the implementation results, and power breakdown are shown in Fig. 15. The reported power consumption is the peak power consumption assuming that all VIP memories are used in the local maximum pixel location search mode, and the memory-centric NoC and all PEs are fully loaded. Due to the heavily parallel attribute of the local maximum location search operation which performs nine reads, four 3-operanded comparisons and an address calculation in a single cycle, the VIP memories account for more than half of the total power consumption. However, the power overhead of the memory-centric NoC is relatively small. The power consumption of the VIP memories is obtained by HSPICE simulation under the conditions of 25 °C, 1.8 V power supply voltage, and typical corner transistors. For the other parts of the chip, the power consumption is estimated using a Synopsys Power Compiler, which calculates the power consumption with a post-synthesis gate level netlist and the traces of switching activity result from the Verilog HDL simulation. The 81.6 GOPS peak performance results from summing the 16 GOPS obtained from the SIMD operation of the ten PEs and the 65.6 GOPS of the eight VIP memories. For verification, we also implemented a verification system as shown in Fig. 16(a). Using the field-programmable gate array (FPGA), we implemented an external memory interface, video camera subsystem, and LCD controller. The correct chip operation is separately tested up to 200 MHz, and the operation speed of the chip in the



Chip Photograph

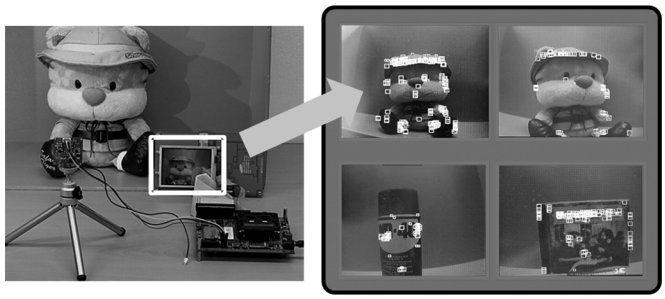
Technology	0.18um 1-poly 6 Metal
Chip Size	7.7mm x 5mm
Clock Freq.	400 MHz (NoC) / 200 MHz (Other Part)
Gate Counts (NAND2 Equiv.)	838.8K Gates
On-Chip Memory	Total : 30 KB VIP Memory : 1.5KB x 8 PE Local Mem. : 1KB x 10 RISC Cache : Data $ 4KB, Inst. $ 4KB
Peak Power Consumption	1.4W at 1.8V

Implementation Summary



Power Breakdown

Fig. 15. Chip photograph, implementation summary, and power breakdown of the proposed object recognition processor.



(a) (b)

Fig. 16. (a) Verification system and (b) key-point localization results.

verification system is 100 MHz considering the reliability of the operation that requires synchronization with the FPGA. The total measured power consumption of the verification system shown in Fig. 16(a) is 2.6 W, and the proposed processor consumes 540 mW when it computes the key-point localization stage of the SIFT at a 100 MHz operation frequency with a 1.8 V supply voltage. Fig. 16(b) shows the implemented chip performing the key-point localization stage of the SIFT. The rectangles encompass the localized key points and are placed on the points of human attention, such as the eyes, nose, and arms of the doll.

VII. Conclusion

To answer the demand for a high-performance and power-efficient object recognition processor design, we implemented an 81.6 GOPS object recognition processor based on a memory-centric NoC. In this paper, we have shown that the processor design based on the analysis of the target application achieves vast performance gains over the conventional general purpose

processors. For that purpose, three perspectives of the chip design—processor architecture, hardware acceleration, and programming model—are considered simultaneously.

At first, the architecture of the proposed processor is determined by carefully considering the intrinsic parallelism and characteristics of the data transactions in the target application. To support both task and data parallelism in the SIFT object recognition, the decision of multi-processor architecture and SIMD data path integration for each PE are made in advance. Then, we observed the unidirectional data flow in a pipelined task execution of the SIFT to place on-chip shared data buffers (VIP memories) among the PEs. The hierarchical star topology of the memory-centric NoC is adopted to provide flexibility in configuring the task-level pipeline. In the NoC, the PEs and VIP memories are located so that the average hop count between each PE and the VIP memories is minimized since direct PE-to-PE data transactions rarely occur. The reduction in power consumption and chip area is an additional advantage of adopting a hierarchical star topology instead of the conventional mesh topology. Second, the remaining performance bottlenecks resulting from the two most demanding operations of the SIFT computation are resolved by integrating the appropriate hardware accelerations, namely the SDP instruction and the VIP memories. Finally, we defined the coherence and consistency protocol regarding the proposed processor architecture and the memory-centric NoC. The corresponding programming model is also proposed to reduce the polling overhead of the shared buffer management and data transaction control between the producer and consumer PEs in the pipelined task execution.

Regarding the three design perspectives described above, we implemented an 81.6 GOPS object recognition processor using a 0.18-μm CMOS process, and it computes the key-point localization stage of the SIFT approximately two times faster than the 2.3 GHz Core 2 Duo processor while only consuming 1.4 W of power. By exploiting the proposed object recognition processor, the implementation of high performance and low power platform for mobile intelligent robots is enabled.

REFERENCES

[1] S. Kyo, T. Koga, S. Okazaki, and I. Kuroda, "A 51.2 GOPS scalable video recognition processor for intelligent cruise control based on a linear array of 128 4-way VLIW processing elements," in *IEEE Int. Solid-State Circuits Conf. Dig. Techn. Papers*, 2003, vol. 1, pp. 48–477.

[2] W. Raab, N. Bruels, U. Hachmann, J. Harnisch, U. Ramacher, C. Sauer, and A. Techmer, "A 100 GOPS programmable processor for vehicle vision systems," *IEEE Des. Test Comput.*, vol. 20, no. 1, pp. 8–15, Jan. –Feb. 2003.

[3] J. Tanabe, Y. Taniguchi, T. Miyamori, Y. Miyamoto, H. Takeda, M. Tarui, H. Nakayama, N. Takeda, K. Maeda, and M. Matsui, "Visconti: Multi-VLIW image recognition processor based on configurable processor," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2003, pp. 185–188.

[4] U. Ozguner, C. Stiller, and K. Redmill, "Systems for safety and autonomous behavior in cars: The DARPA grand challenge experience," *Proc. IEEE*, vol. 95, no. 2, pp. 397–412, Feb. 2007.

[5] J. Kumagai, "A robotic sentry for Korea's demilitarized zone," *IEEE Spectrum*, vol. 44, no. 3, pp. 16–17, Mar. 2007.

[6] S. Ahn, M. Choi, J. Choi, and W. K. Chung, "Data association using visual object recognition for EKF-SLAM in home environment," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2006, pp. 2760–2765.

[7] P. Jensfelt, S. Ekvall, D. Kragic, and D. Aarno, "Augmenting SLAM with object detection in a service robot framework," in *Proc. IEEE Int. Symp. Robot Human Interactive Commun.*, 2006, pp. 741–746.

[8] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," in *Proc. IEEE Int. Conf. Adv. Robot.*, 2005, pp. 492–497.

[9] C. Y. Lin, P. C. Jo, and C. K. Tseng, "Multi-functional intelligent robot DOC-2," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2006, pp. 530–535.

[10] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent ASIMO: System overview and integration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, vol. 3, pp. 2478–2483.

[11] D. Kim, K. Kim, J.-Y. Kim, S. Lee, and H.-J. Yoo, "Solutions for real chip implementation issues of NoC and their application to memory-centric NoC," in *Proc. ACM/IEEE Int. Symp. Netw.-on-Chip*, May 2007, pp. 30–39.

[12] D. Kim, K. Kim, J.-Y. Kim, S. Lee, and H.-J. Yoo, "An 81.6 GOPS object recognition processor based on NoC and visual image processing memory," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2007, pp. 443–446.

[13] D. Kim, K. Kim, J.-Y. Kim, S. Lee, and H.-J. Yoo, "Implementations of memory-centric NoC for 81.6 GOPS object recognition processor," in *Proc. IEEE Asian Solid-States Circuits Conf.*, Nov. 2007, pp. 47–50.

[14] J.-Y. Kim, D. Kim, D. Kim, S.-J. Lee, K. Kim, S. Jeon, and H.-J. Yoo, "Visual image processing RAM for fast 2-D data location search," in *Proc. IEEE Eur. Solid-State Circuits Conf.*, Sep. 2007, pp. 324–327.

[15] D. G. Lowe, "Distinctive image features from scale-invariant key points," *ACM Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[16] M. D. Taylor, W. Lee, S. P. Amarasinghe, and A. Agarwal, "Scalar operand networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 145–162, Feb. 2005.

[17] F. Bertolli, P. Jensfelt, and H. I. Christensen, "SLAM using visual scan-matching with distinguishable 3D points," *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 4042–4047, Oct. 2006.

[18] Z. Nan, L. Maohai, and H. Bingrong, "Active mobile robot simultaneous localization and mapping," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2006, pp. 1671–1681.

[19] S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, R. L. Stamm, and D. M. Tullsen, "Simultaneous multithreading: A platform for next-generation processors," *IEEE Micro*, vol. 17, no. 5, pp. 12–19, Sep.–Oct. 1997.

[20] A. El-Moursy, R. Garg, D. H. Albonesi, and S. Dwarkadas, "Partitioning multi-threaded processors with a large number of threads," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Mar. 2005, pp. 112–123.

[21] T. R. Jacobs, V. A. Chouliaras, and D. J. Mulvaney, "Thread-parallel MPEG-2, MPEG-4 and H.264 video encoders for SoC multi-processor architectures," *IEEE Trans. Consumer Electron.*, vol. 52, no. 1, pp. 269–275, Feb. 2006.

[22] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power networks-on-chip for high-performance SoC design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 2, pp. 148–160, Feb. 2006.

[23] J.-Y. Kim and H.-J. Yoo, "Bitwise competition logic for compact digital comparator," in *Proc. IEEE Asian Solid-States Circuits Conf.*, Nov. 2007, pp. 59–62.

[24] M. S. Papamarcos and J. H. Patel, "A low-overhead coherence solution for multiprocessors with private cache memories," in *Proc. ACM/IEEE 11th Ann. Int. Symp. Comput. Arch.*, 1984, pp. 348–354.

[25] F. Dahlgren, M. Dubois, and P. Stenstrom, "Performance evaluation and cost analysis of cache protocol extensions for shared-memory multiprocessors," *IEEE Trans. Comput.*, vol. 47, no. 10, pp. 1041–1055, Oct. 1998.

[26] D. A. Patterson and J. L. Hennessy, *Computer Architecture: A Quantitive Approach*, 3rd ed. San Mateo, CA: Morgan Kaufmann, 2003, pp. 528–664.

[27] A.-R. Adl-Tabatabi, C. Kozyrakis, and B. Saha, "Unlocking concurrency: Multicore programming with transactional memory," *ACM Queue*, vol. 4, no. 10, pp. 24–33, Dec. 2006.

[28] L. Hammond, V. Wong, M. Chen, B. D. Carlstrom, J. D. Davis, B. Hertzberg, M. K. Prabhu, H. Wijaya, C. Kozyrakis, and K. Olukotun, "Transactional memory coherence and consistency," in *Proc. 31st Ann. Int. Symp. Comput. Arch.*, 2004, pp. 19–23.

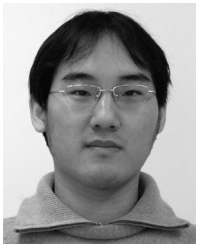
[29] M. K. Prabhu and K. Olukotun, "Using thread-level speculation to simplify manual parallelization," in *Proc. ACM Principles Practices Parallel Program.*, 2003, pp. 1–12.

[30] J. G. Steffan and T. C. Mowry, "The potential for using thread-level data speculation to facilitate automatic parallelization," in *Proc. Int. Symp. High-Perform. Comput. Arch.*, Feb. 1998, pp. 2–13.

[31] J. W. van den Brand and M. Bekooij, "Streaming consistency: A model for efficient MPSoC design," in *Proc. IEEE 10th Eur. Micro Conf. Digit. Syst. Des. Arch., Methods Tools*, Aug. 2007, pp. 27–34.
[32] O. P. Gangwal, A. Nieuwland, and P. A. Lippens, "Scalable and flexible data synchronization scheme for embedded HW-SW shared-memory systems," in *Proc. IEEE Int. Symp. Syst. Synth.*, 2001, pp. 1–6.
[33] A. Nieuwland, J. Kang, O. Gangwal, R. Sethuraman, N. Busa, K. Goosens, R. Pesetllopis, and P. Lippens, "C-HEAP: a heterogeneous multi-processor architecture template and scalable and flexible protocol for the design of embedded signal processing systems," *Kluwer Des. Autom. Embed. Syst.*, Oct. 2002.
[34] H. Kim, B.-G. Nam, J.-H. Sohn, J.-H. Woo, and H.-J. Yoo, "A 231-MHz, 2.18-mW 32-bit logarithmic arithmetic unit for fixed-point 3-D graphics system," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2373–2381, Nov. 2006.
[35] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny, "The power of priority: NoC based distributed cache coherency," in *Proc. IEEE 1st Int. Symp. Netw.-on-Chip*, May 2007, pp. 117–126.
[36] N. Eisley, L.-S. Peh, and L. Shang, "In-network cache coherence," in *Proc. IEEE/ACM 39th Int. Symp. Microarch.*, Dec. 2006, pp. 321–332.
[37] F. Petrot, A. Greiner, and P. Gomez, "On cache coherency and memory consistency issues in NoC based shared memory multiprocessor SoC architectures," in *Proc. IEEE 9th EUROMICRO Conf. Digit. Syst. Des.*, 2006, pp. 53–60.
[38] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. B. Carter, "Interconnect-aware coherence protocols for chip multiprocessors," in *Proc. 33rd Ann. Int. Symp. Comput. Arch.*, Jun. 2006, pp. 339–351.

Donghyun Kim (S'03) received the B.S. degrees in electrical engineering and computer science from Korea Advance Institute of Science and Technology (KAIST), Daejeon, Korea, in 2003, where he is currently pursuing the Ph.D. degree in electrical engineering and computer science.

His research interests include network-on-chip design, multi-processor design, and parallel image processing. He is currently working for an analysis of on-chip data transactions and task mappings of applications with streamed data flow into multi-processor SoC architectures.

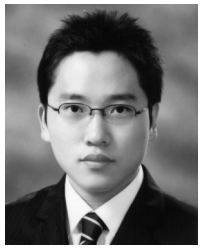


Kwanho Kim (S'04) received the B.S. and M.S. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2004 and 2006, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering and computer science.

In 2004, he joined the Semiconductor System Laboratory (SSL), KAIST, as a Research Assistant. His research interests include VLSI design for object recognition, architecture, and implementation of NoC-based SoC.

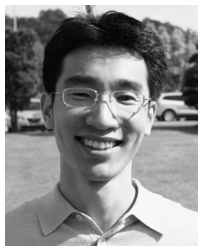
Joo-Young Kim (S'05) received the B.S. and M.S. degrees in electrical engineering and computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2005 and 2007, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering and computer science.

He has been involved with the development of the parallel processor for computer vision, as a high speed arithmetic block and embedded memory designer. Currently, his research interests include parallel architectures and sub-systems for computer vision chip.



Seungjin Lee (S'06) received the B.S. and M.S. degrees in electrical engineering and computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2006 and 2008, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering and computer science.

His research interests include low power digital signal processors for digital hearing aids and body area communication. Currently, he is investigating parallel architectures for computer vision processing.



Se-Joong Lee (S'99–M'05) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1999, 2001, and 2005, respectively.

Since 2005, he has been working for the Communication and Medical Systems Laboratory, Texas Instruments, Dallas. Currently, he is leading a hardware platform team for Media Access Control system for broadband wireless data communication. His research interests include architecture and chip implementation of network-on-chips, low-power high-performance digital circuits, and power management in system-on-chips. His network-on-chip designs were recognized from Korea Integreated Circuits Design Contest in 2002 and 2004, respectively. He contributed a book chapter to *Network-on-Chips: Technology and Tools* (Morgan Kaufmann, 2006).

Hoi-Jun Yoo (M'95–SM'04–F'08) received the B.S. degree in electronic from Seoul National University, Seoul, Korea, in 1983, and the M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1985 and 1988, respectively. His Ph.D. work concerned the fabrication process for GaAs vertical optoelectronic integrated circuits.

From 1988 to 1990, he was with Bell Communications Research, Red Bank, NJ, where he invented the 2-D phase-locked VCSEL array, the front-surface-emitting laser, and the high-speed lateral HBT. In 1991, he became Manager of a DRAM Design Group, Hyundai Electronics, and designed a family of from fast-1 M DRAMs and 256 M synchronous DRAMs. In 1998, he joined the faculty of the Department of Electrical Engineering, KAIST, and is currently a full Professor. From 2001 to 2005, he was the director of System Integration and IP Authoring Research Center (SIPAC), funded by the Korean government to promote worldwide IP authoring and its SOC application. From 2003 to 2005, he was the full time Advisor to the Minister of Korea Ministry of Information and Communication and National Project Manager for SoC and Computer. In 2007, he founded System Design Innovation and Application Research Center (SDIA), KAIST, to research and develop SoCs for intelligent robots, wearable computers and bio systems. His current research interests include high-speed and low-power network-on-chips, 3-D graphics, body area networks, biomedical devices and circuits, and memory circuits and systems. He is the author of the books *DRAM Design* (Hongleung, 1996; in Korean), *High Performance DRAM* (Sigma, 1999; in Korean), and chapters of *Networks on Chips* (Morgan Kaufmann, 2006).

Dr. Yoo was a recipient of the Electronic Industrial Association of Korea Award for his contribution to DRAM technology in 1994, the Hynix Development Award in 1995, the Korea Semiconductor Industry Association Award in 2002, the Best Research of KAIST Award in 2007, the Design Award of 2001 ASP-DAC, and the Outstanding Design Awards in 2005 and 2006, and the 2007 A-SSCC. He is a member of the executive committee of ISSCC, Symposium on VLSI, and A-SSCC. He is the TPC chair of the A-SSCC 2008.