DSIP: A Scalable Inference Accelerator for Convolutional Neural Networks

Jihyuck Jo, *Student Member, IEEE*, Soyoung Cha, *Student Member, IEEE*, Dayoung Rho, *Student Member, IEEE*, and In-Cheol Park, *Senior Member, IEEE*

***Abstract*— This paper presents a scalable inference accelerator called a deep-learning specific instruction-set processor (DSIP) to support various convolutional neural networks (CNNs). For CNNs requiring a large amount of computations and memory accesses, a programmable inference system called master–slave instruction set architecture (ISA) is newly proposed to achieve high flexibility, processing speed, and energy efficiency. The master is responsible for sending and receiving feature maps in order to deal with neural networks in a scalable way, and the slave performs CNN operations, such as multiply accumulate, max pooling, and activation functions, on the features received from the master. The master–slave ISA maximizes computation speed by overlapping the off-chip data transmission and the CNN operations, and reduces power consumption by performing the convolution incrementally to reuse input and partial-sum data as maximally as possible. An inference system can be configured by connecting multiple DSIPs in a form of either 1-D or 2-D chain structure in order to enhance computation speed further. To evaluate the proposed accelerator, a prototype chip is implemented and evaluated for AlexNet. Compared to the state-of-the-art accelerator, the DSIP-based system enhances the energy efficiency by 2.17×.**

***Index Terms*— Deep neural network, energy-efficient accelerator, heterogeneous instruction set architecture, object recognition, scalable architecture.**

I. Introduction

DUE to the excellent inference accuracy of machine-learning techniques, many companies are applying convolutional neural network (CNN)-based algorithms to various services such as speech recognition and object recognition. As CNNs can achieve better accuracy by increasing the number of layers, the computational complexity increases rapidly nowadays [1], [2]. To perform such a complicated CNN in real time, many CPUs and GPUs specialized for parallel computation have been employed at the expense of large power consumption [3]–[5].

To alleviate the power consumption, diverse application specified integrated circuit solutions have been proposed by using specific arithmetic units and buffers [6]–[9]. The accelerators were built based on the sum-of-product units or an array of processing elements (PEs) that compute convolution operations in parallel, and tried to reduce off-chip memory (OM) accesses by storing input and partial-sum data into buffers and reusing them. To further enhance the energy efficiency, the sparse characteristic of a neural network was utilized in [10] and [11], and the mixed analog-digital accelerator was proposed in [12]. The approaches were successfully applied to CNNs, but they are confined to predefined CNN models.

Instruction set architectures (ISAs) specialized for CNNs have been proposed to increase flexibility [13]–[21]. The processors in [13]–[16] were also built with the sum-of-product units or a PE array, but the control signals for the operation units and the buffers were defined as instructions. The accelerator in [13], in particular, was placed next to an image sensor, eliminating the OM accesses. The accelerators in [20] and [21] adopted memory access instructions to extend the degree of freedom in accessing the OM, and supported recurrent neural networks as well as CNNs. However, the ISAs are for a single-chip solution, assuming that available hardware resources are sufficient to process the CNN. As a large amount of computations is required in recent CNNs, the single-chip approach is not appropriate in implementing a complex CNN targeting for real-time applications.

In this paper, we propose a deep-learning specific instruction-set processor (DSIP) and its scalable structure. The proposed DSIP consists of two heterogeneous ISAs, called master–slave ISA, to ensure high flexibility. The DSIP ISA is designed to make it easy to reflect the characteristics of a neural network layer and maximize power efficiency by performing the convolution operation incrementally. To increase the processing speed of a complex CNN, furthermore, we propose a multiple-DSIP system connected with a 2-D chain (2DC) structure.

The rest of this paper is organized as follows. Section II describes the CNN and the previous accelerators. The proposed scalable architecture based on the master–slave ISA is explained in Section III, and the hardware structure of the DSIP is addressed in Section IV in detail. The implementation and evaluation results of a prototype chip are summarized in Section V, and concluding remarks are made in Section VI.

II. Background

A CNN algorithm extracts and evaluates feature maps by hierarchically applying multiple filters to input data so as to classify the most suitable object. For this purpose,

Manuscript received April 20, 2017; revised July 8, 2017 and September 21, 2017; accepted October 10, 2017. Date of publication November 10, 2017; date of current version January 25, 2018. This paper was approved by Associate Editor Marian Verhelst. This work was supported in part by the Center for Integrated Smart Sensors funded by the Ministry of Science and ICT as Global Frontier Project under Grant CISS-2011-0031860 and in part by the IC Design Education Center. *(Corresponding author: In-Cheol Park.)*

The authors are with the School of Electrical Engineering, KAIST, Daejeon 34141, South Korea (e-mail: icpark@kaist.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2017.2764045

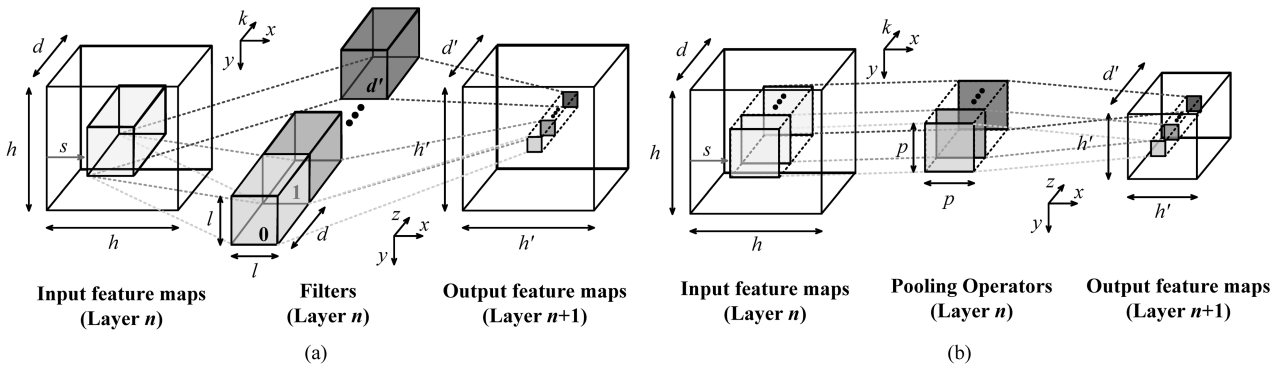

Fig. 1. Layers in CNNs. (a) Convolutional layer. (b) Pooling layer.

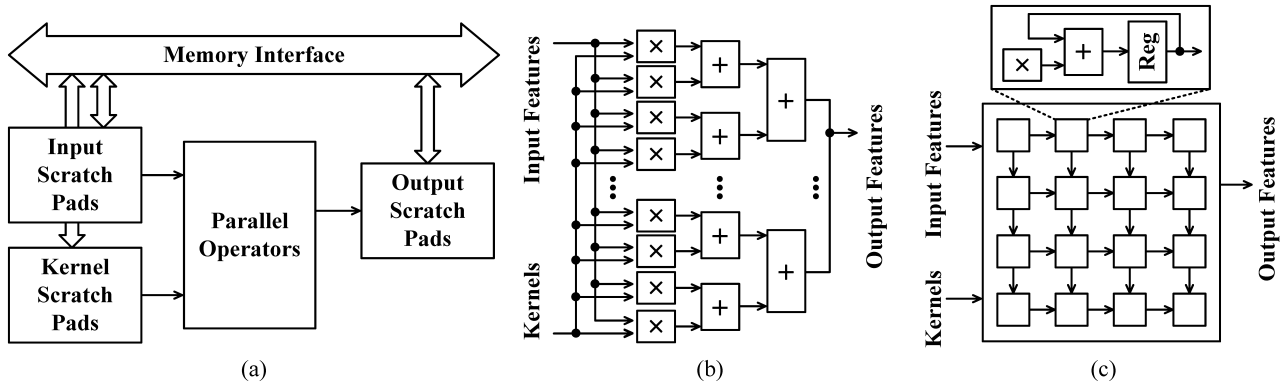

Fig. 2. Fundamental structure of previous CNN accelerators. (a) Overall structure. (b) Output-oriented structure and (c) input-oriented structure of parallel operators.

a CNN consists of multiple combinations of convolutional layers (CONV), pooling layers (POOL), and fully connected layers (FCs). The CONV shown in Fig. 1(a) can be expressed as

$$c_{xyz} = \phi\left(\left(\sum_{j=0}^{l-1}\sum_{i=0}^{l-1}\sum_{k=0}^{d-1} w_{ijkz} \times f_{(sx+i)(sy+j)k}\right) + b_z\right) \quad (1)$$

where f, c, w, s, and b denote the input feature, the output feature, the filter, the stride size, and the offset value, respectively, and ϕ is an activation function. The subscript of a parameter indicates its index. The POOL illustrated in Fig. 1(b) extracts a representative value. The FC plays the role of classification [14], but it is not described in detail because its operation is similar to that of the CONV.

The fundamental structure employed in the previous CNN accelerators [6]–[21] is depicted in Fig. 2(a). Depending on the arrangement of parallel operators, a CNN accelerator can be classified into two categories: output-oriented structure and input-oriented structure. The output-oriented structure employed in [6], [7], [14], [18], [19], and [21] consists of the sum-of-product units shown in Fig. 2(b). This structure is designed to calculate output features intuitively by loading multiple input features relevant to the output features every cycle. As a result, a relatively large amount of energy is consumed to load the same input features several times when processing a CONV layer with a small stride size.

The input-oriented structure can mitigate the redundant loading by taking the 2-D PE array proposed in [8], [9], [13], [15], and [16] or the 1-D PE array adopted in [17]. Fig. 2(c) illustrates a structure of the 2-D PE array. The dataflow specialized for 2-D convolutions passes the input and partial-sum data to neighboring PEs. However, since each PE controls the inter-PE communication independently in the array, the power consumption and the area of the control logic are much larger than those of the output-oriented structure.

III. Proposed Scalable DSIP Architecture

In this section, the DSIP architecture is proposed based on the master–slave ISA to maximize the computing speed and scalability. In the slave ISA, CNN operations are processed on a partial-sum-oriented structure to reduce the power consumption.

As explained above, all the layers in a CNN prepare feature maps and filter parameters, perform a large amount of computation, and store the resulting feature map. There are two observations that should be considered in deriving an efficient architecture.

First, in a layer, the time to prepare the operation and store the result is as long as the time to compute the required operations. If data transmission/reception and data operation are independently processed in heterogeneous processors, the throughput can be increased by overlapping the two processes. Second, a neural network can be thought of a

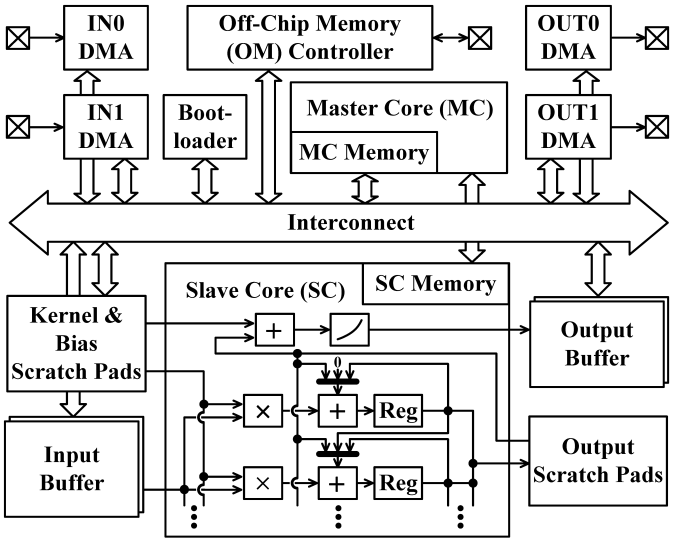

Fig. 3. Overall structure of the proposed scalable DSIP.

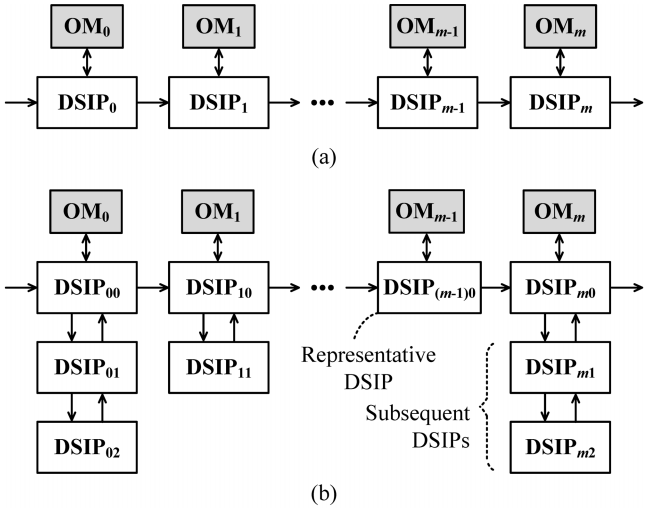

Fig. 4. Multiple-DSIP systems. (a) 1-D chain structure. (b) 2-D chain structure.

series of several small neural networks, since all the layers have similar dataflows. Once an instruction-set processor can transfer the input and output data of a CNN layer, a complex neural network can be processed by serially connecting a number of accelerators developed to process small neural networks. The two observations play an important role in deriving a new CNN accelerator based on the master–slave ISA. Note that the second observation was already utilized in [22] to mimic the behavior of biological neurons, but the proposed instruction-set processor is developed to enhance scalability and deal with larger layers.

A. Overall Structure of the Proposed DSIP

Fig. 3 shows a conceptual block diagram of the scalable DSIP structure. Following the von Neumann architecture, the OM stores both programs and necessary data such as feature maps and filter weights. The master core (MC) is responsible for sending and receiving feature map data and managing the slave core (SC). The SC performs CNN operations and reports the results to the MC.

To process multiple CNN layers in a single accelerator, the bootloader first loads programs, and relevant data from the OM to the MC memory. Following the layer information in the MC memory, the MC prepares filter weights, biases, and input feature maps and saves them to the kernel scratch pad (KSP), the bias scratch pad (BSP), and the input buffer (IB), respectively. After the transmission process, the MC invokes the SC to perform the operations specified in the SC memory on the transmitted data. When the operations are finished, the SC saves the results into the output scratch pad (OSP) and the output buffer (OB), and then wakes up the MC to resume the next work. The MC repeats the above process for all layers, and stores the output feature map of the last layer into the OM. Employing the double-buffering technique to the IB and OB, the total processing time can be reduced by performing the data transmission and the CNN operations in parallel in the master–slave core.

As shown in Fig. 3, the MC can send and receive feature map data to the neighbor DSIPs as well as the OM, so it is possible to decompose a complex neural network into several groups and distribute each group to a different DSIP chip. Two inference systems consisting of several DSIPs are shown in Fig. 4. The 1-D chain (1DC) structure depicted in Fig. 4(a) is the basic extension of a scalable DSIP, which is effective in increasing the overall processing speed by processing independent CNN layers in parallel. The 2DC structure illustrated in Fig. 4(b) connects additional DSIPs to a DSIP of the 1DC structure. The 2DC structure is utilized to improve the processing speed, when the amount of kernel data to be processed in a CNN is too large to be handled by a single accelerator. Since a vertical set of DSIPs processes the same layer such as a single accelerator, one OM is shared to prevent storing feature maps and filter data redundantly. Therefore, each vertical set has a representative DSIP that is directly connected to the OM, and the other DSIPs in the set are connected to the representative DSIP.

To perform a layer operation in a vertical set of DSIPs, the DSIPs should load data from the OM and store their outputs into the OM. As the representative DSIP plays the role of a gatekeeper, it performs a handshaking process to make the subsequent DSIPs access the OM. If the request type is load accesses, the subsequent DSIP can receive the acknowledgement of the request with the target data loaded from the OM. If the representative DSIP gets a store request, it first grants permission to the subsequent DSIP, and then, the subsequent DSIP can transmit its outputs to the OM. Since a vertical set of DSIPs utilizes the same input feature map to compute convolutions, a broadcast function is also provided to reduce the off-chip communication. The representative DSIP can broadcast the feature data loaded from the OM to all subsequent DSIPs.

B. Master–Slave Architecture

To implement the proposed structure, the MC and the SC are developed separately. Their instruction sets are constructed considering the repetitive calculations in (1).

TABLE I
REGISTERS OF THE MASTER ISA

Type	Name	Meaning
SPR	INFO	DSIP status and index in the 2DC
	LC0~LC2	Loop counters
	WDT	Width of the current layer
	HIT	Height of the current layer
	DPT	Depth of the current layer
	NSD	The number of subsequent DSIPs
GPR	GR0~GR7	General-purpose registers

TABLE II
INSTRUCTIONS OF THE MASTER CORE

Type	Name	Function
Arithmetic and logical	ADD	RD = RA + RB or RD = RA + imm16
	SUB	RD = RA − RB or RD = RA − imm16
	MUL	RD = RA × RB or RD = RA × imm16
	AND	RD = RA & imm16
	OR	RD = RA | imm16
	DEC	LCa = LCa − 1
	ASR	RD = RA >> imm16 (>>: arithmetic right shift)
Flow control	BR	PC = imm16
	CBR	If RB is 0, PC = imm16
	START	Start subsequent DSIPs and the next representative DSIP in the 2DC
	VSYN	Vertical sync signal in the 2DC
	HSYN	Horizontal sync signal in the 2DC
Data transfer	MOV	RD = RA
	MOVLO	RD = imm16
	MOVHI	RD = RD | (imm16 × 2^{16})
	OFMS	Off-chip memory store
	OFMLR	Off-chip memory load request
	OFMLT	Off-chip memory load and transmission
SC-related	SCF	Fill slave instructions
	SCR	Run the slave core

Table I summarizes the register set of the master ISA. Special-purpose registers are used to store constant parameters for a layer, and general-purpose registers store the results of data processing instructions.

Since the MC operation is relatively simple, the master ISA is designed following the reduced instruction-set computer architecture. As shown in Table II, the master ISA contains four instruction types. The arithmetic type is used to specify layer characteristics and to calculate memory addresses. In the control type, the branch instruction is to perform loop operations, and the sync instructions are to make multiple DSIPs start their processing at the same time in a multiple-DSIP system. The data transfer type is to access registers and memories. In case of memory accesses, it transfers data among the OM, scratch pads, lookup tables (LUTs), MC memories, and SC memories in DSIPs. The OFMLT instruction is utilized by the representative DSIP to transmit feature data and initial information such as the master–slave program and LUT data to a subsequent DSIP. The OFMLR instruction is to send a load request from the subsequent DSIP. Both instructions can select whether the load access is for broadcasting or not. The OFMS instruction, which can be performed by all types of DSIPs, is utilized to save the output data into the current OM or the following OM. The SC-related type is used by the MC to program the SC and make it start the execution.

TABLE III
INSTRUCTIONS OF THE SLAVE CORE

Slot	Name	Function
Flow control	SIT	Set the number of iterations
	SBP	Set a break point, memory addressing by ±1
	CLP	Conditional loop, memory addressing by ±1
Address control	ACT	Memory addressing by an immediate value
Memory load	MLD0	Memory load (IB)
	MLD1	Memory load (IB, KSP)
	MLD2	Memory load (IB, OSP)
	MLD3	Memory load (IB, KSP, OSP)
	MLD4	Memory load (OSP, BSP)
Arithmetic	MAC	Multiply and accumulate
	MAX	Take a maximum operation
Nonlinear	LUT	Take an activation function
Memory store	MST0	Memory store (OB)
	MST1	Memory store (OSP)

As described in Table III, the slave ISA is developed based on a six-slot very-long-instruction-word architecture to perform multiple operations at the same time. The flow-control slot specifies a loop of a program. The addresses of buffers and scratch pads can be modified by the offset specified in the address-control slot. The memory-load slot is utilized to read data from the IB, KSPs, OSPs, and BSP, and the memory-store slot is to store the operation result into the OSP or the OB. The arithmetic slot and the nonlinear slot perform CNN operations described in Section II.

C. CNN Operations With the Master–Slave ISA

CNN operations are associated with a large amount of computations, requiring a lot of time and energy. To mitigate this overhead, this section discusses how to apply the data reuse techniques to single-DSIP and multiple-DSIP systems.

Fig. 5 shows how a 1-D convolution is processed in different CNN accelerators based on the minimum required hardware. As illustrated in Fig. 5(a), we need to compute and add three partial sums to obtain an output, as the kernel consists of three constants. In most output-oriented structures, three input features must be read from the IB in each time to perform the 1-D convolution as shown in Fig. 5(b). It requires three multipliers to calculate the partial sums in parallel and three adders to aggregate the partial sums. In a general CNN, as the stride is usually smaller than the kernel size, the output-oriented structure reads the same features multiple times. To eliminate the redundant memory accesses, the input-oriented structures [8], [16] load input features and reuse them by using an input scratch pad memory as illustrated in Fig. 5(c) and (d), where f_i' is the ith input feature, f_i loaded from the input scratch pad memory. The former structure [8] utilizes only one PE, which contains a multiply-accumulator (MAC) unit, repeatedly to obtain output features in a serial manner, whereas the latter structure [16] contains five PEs to compute them in parallel. Since both structures

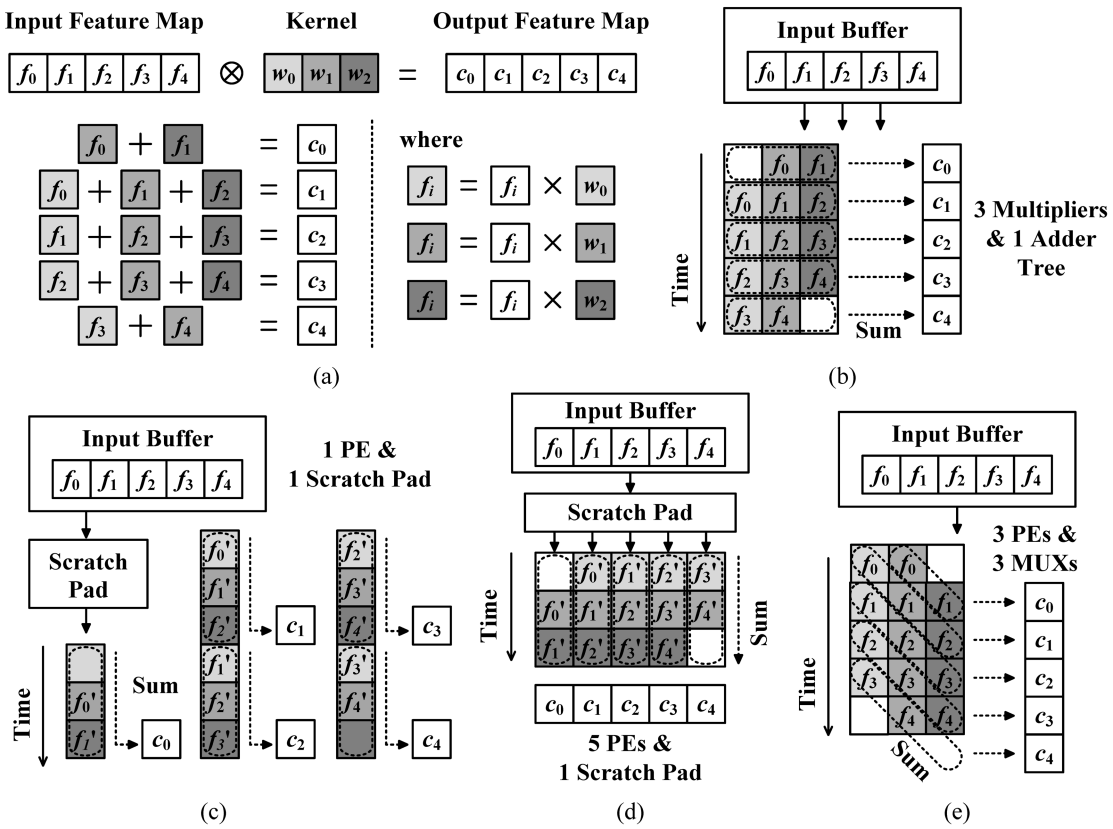

Fig. 5. 1-D convolution. (a) Convolution of 1-D features and filter weights, and the processing flows of (b) output-oriented structure, (c) input-oriented structure in [8] and (d) [16], and (e) proposed partial-sum-oriented structure.

require extra storages and control logics that carry input features, additional power is consumed to load the feature and kernel data every cycle. To enhance power efficiency, a new structure called the partial-sum-oriented structure is proposed in this paper, and the processing flow is depicted in Fig. 5(e). Unlike the previous structures, all the partial sums associated with an input feature are computed at once and passed to compute the output features incrementally. Passing the partial sums can be implemented with small multiplexers as shown in Fig. 3, resulting in relatively small power consumption. This approach is called incremental convolution because each partial sum is incrementally computed in a step.

The incremental convolution can be extended to 2-D convolution as depicted in Fig. 6. The inputs and the outputs of 2-D convolution are exemplified in Fig. 6(a). The dotted box indicates a set of input features required to compute an output feature. In the proposed architecture, 2-D convolution is performed by reading the input feature only once as illustrated in Fig. 6(b). As a row of the 2-D kernel forms a stripe, this approach is called incremental stripe convolution (ISC). In Fig. 6(b), X-Loop is a convolutional operation in the x-direction. Within an X-Loop, four input features are sequentially fed into all parallel operators, and the partial sums obtained from the third multiply-accumulate operator of a stripe are stored in the corresponding OSP. In the following X-Loop, the stored values are read and added to the newly computed partial sums. As shown in Fig. 6(b), C_{00} and C_{01} can be obtained at the end of X-Loop 3. Similarly, C_{10} and C_{11} can be obtained in X-Loop 4. Unlike the input-oriented structure, input buffers and control logics can be simplified because input features are broadcast to all PEs, and all inter-PE communication is performed in 1-D PEs only by using small multiplexers.

Fig. 7 illustrates a 2-D max-pooling process. Similar to the convolution process, overlapped input features depicted in Fig. 7(a) are read multiple times, because the stride is smaller than the region considered in a maximum operation. Hence, we aggregate all input features on the KSPs first and perform the max-pooling operation in parallel as illustrated in Fig. 7(b), where X-MAX and Y-MAX indicate the maximum operations performed in the x- and y-directions, respectively. For Y-MAX, each PE computes the maximum value and delivers it to the subsequent PE in the y-direction. As there is no need to re-compute the maximum values of the overlapped rows, it is effective in lowering energy consumption. To reduce energy consumption further, X-MAX is performed immediately after the convolution so as to decrease OM accesses, reducing the number of output features to be generated in a convolutional layer.

To increase the processing speed, multiple operations can be performed concurrently by utilizing unoccupied operators. If 18 MAC operators are available in a DSIP, for example, two 3×3 convolutions for different output feature maps in the z-direction can be calculated in parallel. The parallel

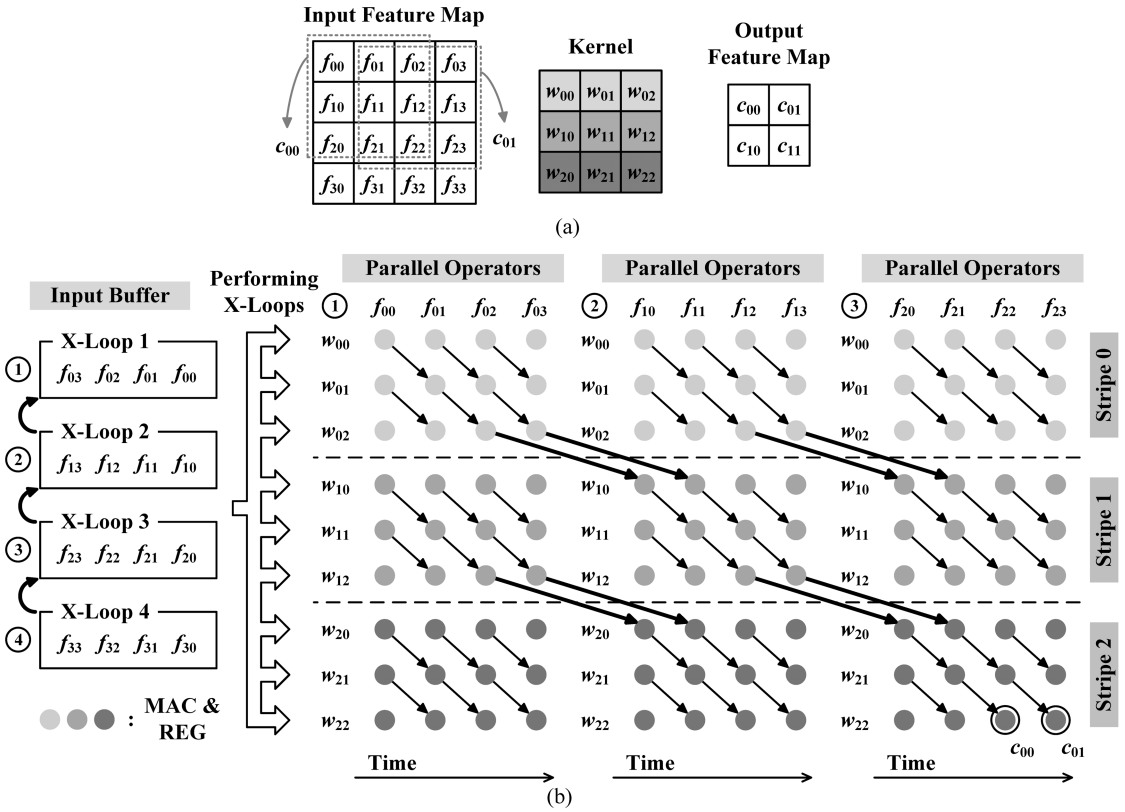

Fig. 6. 2-D convolution. (a) Inputs and outputs. (b) Proposed ISC.

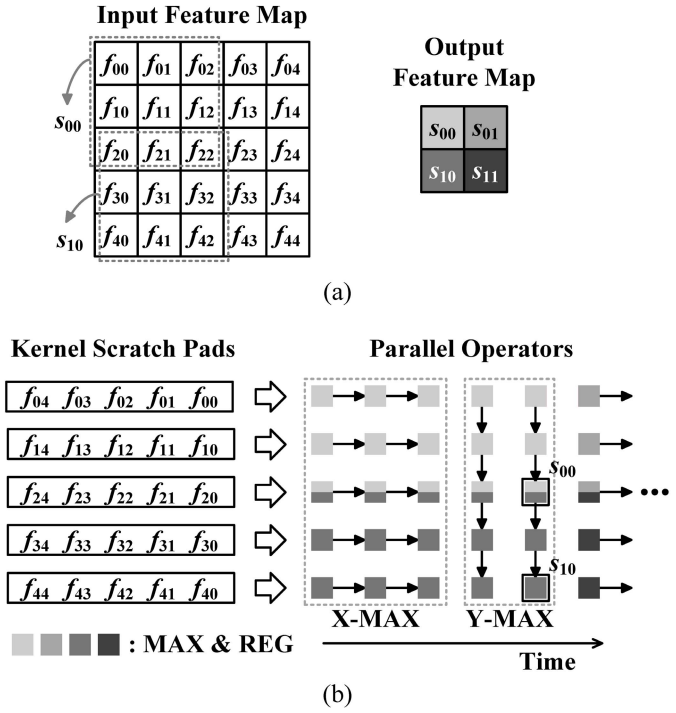

Fig. 7. 2-D max pooling. (a) Inputs and outputs. (b) Proposed incremental stripe pooling.

operations can be applied to the 2DC structure as illustrated in Fig. 8. In the 2DC structure, input feature maps are fed into 2 DSIPs that have 36 MAC operators. Therefore, up to four

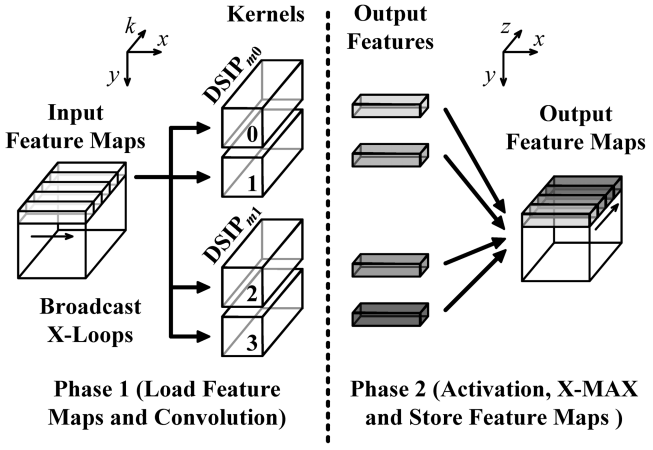

Fig. 8. CNN operations in the 2DC structure.

3×3 convolutions can be performed in parallel for different kernel data. The processing flow of the master–slave cores in the 2 DSIPs is depicted in Fig. 9. Since an MC and an SC separately performs their workload, the convolution operations can be executed while accessing the OM. In phase 1, five X-Loops are processed in parallel. Four sets of output feature maps are then saved into the OM serially in phase 2. A vertical sync instruction is executed at the end of phases 1 and 2 to make the early terminated DSIP wait until the other DSIP completes its operation. When the operations of convolution layers assigned to the DSIPs are completed, a horizontal sync instruction is executed before processing the next image.

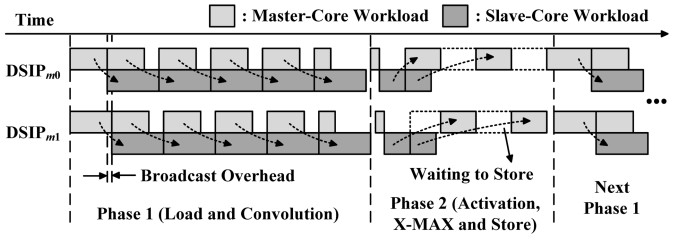

Fig. 9. Data flow of the 2DC DSIP system.

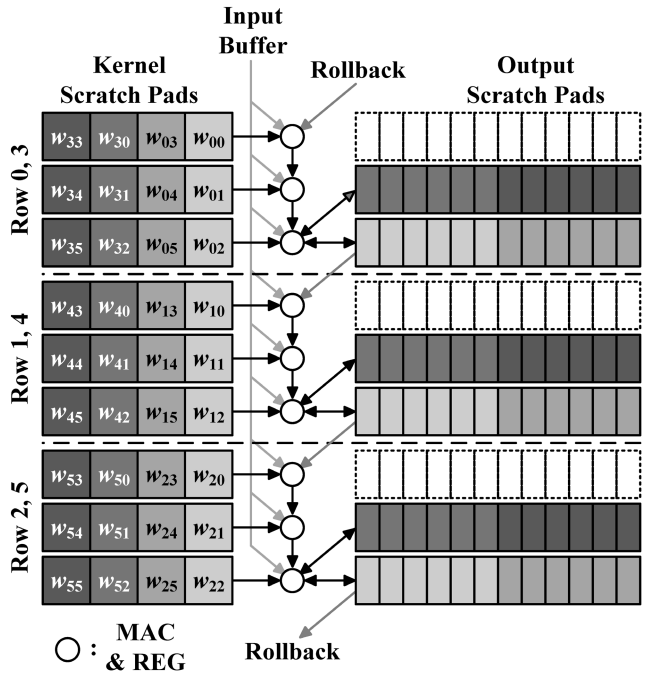

Fig. 10. ISC using kernel decomposition.

In the systems considered so far, it is assumed that a DSIP provides as sufficient hardware resources as required for the convolution operation. As the neural network becomes more complex, however, hardware resources such as MAC operators and buffers become insufficient to perform CNN operations, which make it necessary to save intermediate results into the OM and restore them later when they are needed. To resolve the lack of MAC operators, the kernel is decomposed as shown in Fig. 10, where a 6×6 kernel is segmented into four 3×3 kernels. Next, the convolutions of input features and the 3×3 kernels are separately stored into the OSPs and aggregated to produce the final output features. To manage the intermediate partial sums effectively, an OSP is divided in two parts, so four convolution results can be stored into two neighbor OSPs. In Fig. 10, the rollback connection is used to transfer the intermediate partial sums of the second row (W_{2i}) to the third row (W_{3i}).

Fig. 11 describes how a layer is decomposed when the buffers are not enough to store intermediate partial sums. To process a large input feature map with a small number of buffers, the feature map is divided into segments, and each segment is processed sequentially. During the process, some parts of the feature map, which are colored gray in Fig. 11,

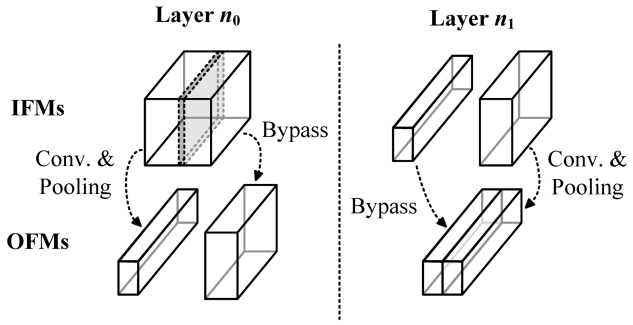

Fig. 11. Layer decomposition when IBs and OBs are insufficient.

are inevitably read twice, since the kernel stride is smaller than the kernel size.

IV. Design of a Scalable DSIP System

As an inference algorithm involves many computations and memory accesses, the operations should be overlapped in time and performed in parallel. This section describes the overall system architecture and high-speed features applied to the 2DC DSIP system.

The overall structure of an object recognition system based on the proposed 2DC DSIP system is shown in Fig. 12. The recognition system works as follows. In the beginning, the DSIP system is initialized using the firmware stored in the NOR flash memory. The firmware includes MC programs, filter weights, and LUT data for activation functions. When the initialization process is completed, the host processor transmits an image, which is captured by the camera sensor module, to the DSIP system through the DSIP interface, and then invokes the inference algorithm in the DSIP system.

In the 2DC DSIP system, representative DSIPs are connected in a chain form. Since it is possible for the previous and current representative DSIPs to access the OM at the same time, the overall performance can be degraded. The time overhead is alleviated by applying the pipelining technique, which allows different vertical sets of DSIPs to operate in parallel. For this, two separate OMs are connected to representative DSIPs. The operation of the pipelining 2DC system is described in Fig. 13. Assuming n CNN layers are processed in each vertical set of DSIPs, $T_{a,i}$ is the time that the 2DC system starts the CNN operation for the ath batch of the ith layer L_i. At the end of the L_{n-1} operation, its output feature map is directly transmitted to OM_{11} instead of OM_{00}. Then, $DSIP_{00}$ changes the role of each OM controller. Due to the double-buffered OMs, $DSIP_{10}$ can continue the CNN operation for the ath batch stored in OM_{11} at $T_{a+1,0}$ without waiting for the L_{n-1} output feature map to come out.

The CNN operation depicted in Fig. 13 can be accelerated by employing the pipeline structure to the master–slave core. As the MC executes simple instructions only, a two-stage pipeline structure depicted in Fig. 14 is enough to achieve high performance. In the first stage, an instruction is fetched (IF) from the MC MEM and decoded (ID) to generate control signals and read registers. The second stage executes (EXE) the operations specified in the instruction, and writes (WB) the results in the register file (RF) or the command queue (CQ).

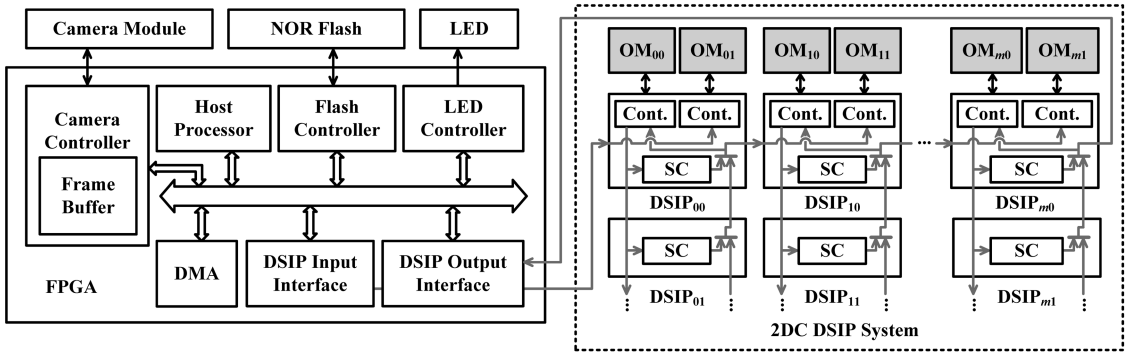

Fig. 12. Overall architecture of an object recognition system employing the proposed DSIPs.

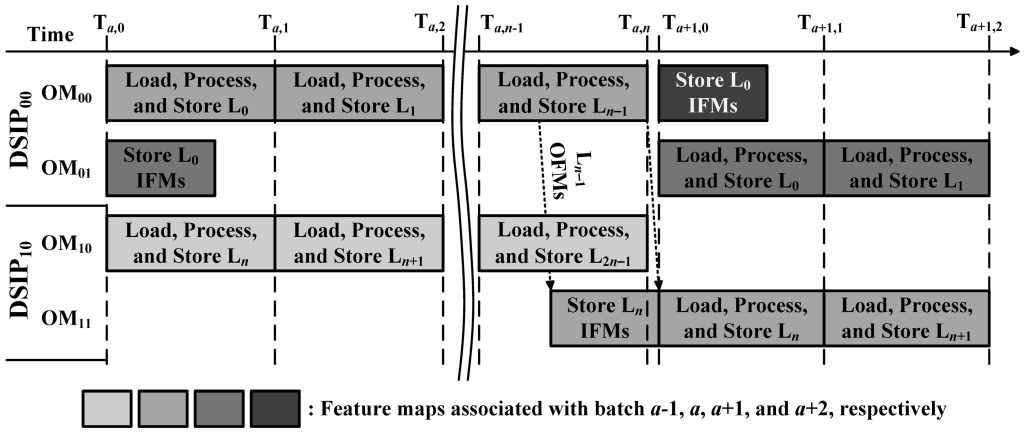

Fig. 13. Time diagram of a 2DC DSIP system with double-buffered OMs.

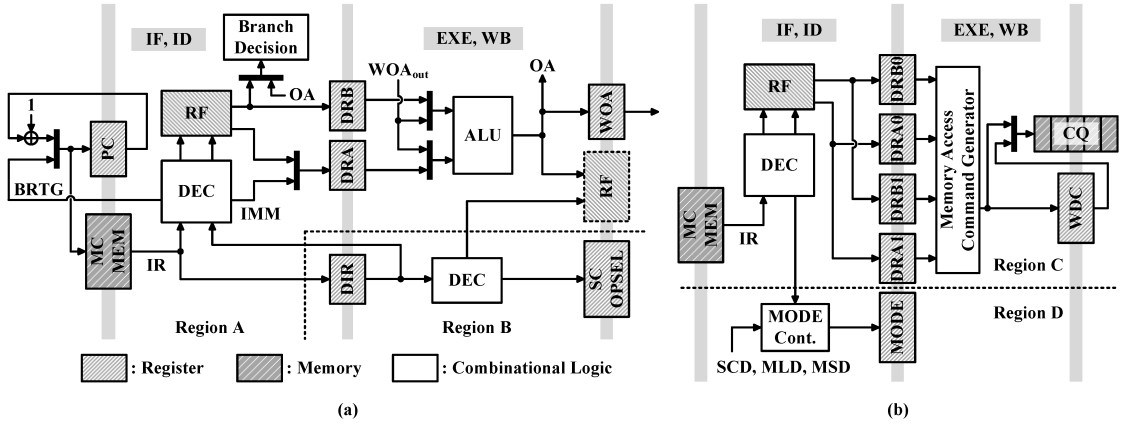

Fig. 14. Pipeline architecture of the MC. (a) Datapath for arithmetic, flow control, and SC-related instructions. (b) Datapath for memory access instructions.

The region A of Fig. 14(a) is designed for flow control and arithmetic instructions. The program counter (PC) indicates the address of the MC MEM, and a branch instruction can assign a branch target address to the PC. As two consecutive instructions can be data dependent on each other, the OA and WOA_{out} signals are forwarded. The region B manages SC operations according to the SC OPSEL that is set by an SCR instruction. It stores three parameters: the number of active parallel operators, the modulus, and the remainder. The first parameter activates a subset of parallel operators, the second determines the width of a stripe in the ISC, and the third parameter is to select one index within the stripe.

The region C of Fig. 14(b) is designed for OM-access instructions of the MC. The arguments specified in the memory access instruction are converted to an OM-access command and stored in the CQ. The commands are sequentially transmitted to either the OM controller or external DSIPs. The pending status of the MC operation, which is stored in the MODE register, is controlled in the region D. In the master–slave core, the OM-access and the CNN-operation processes can

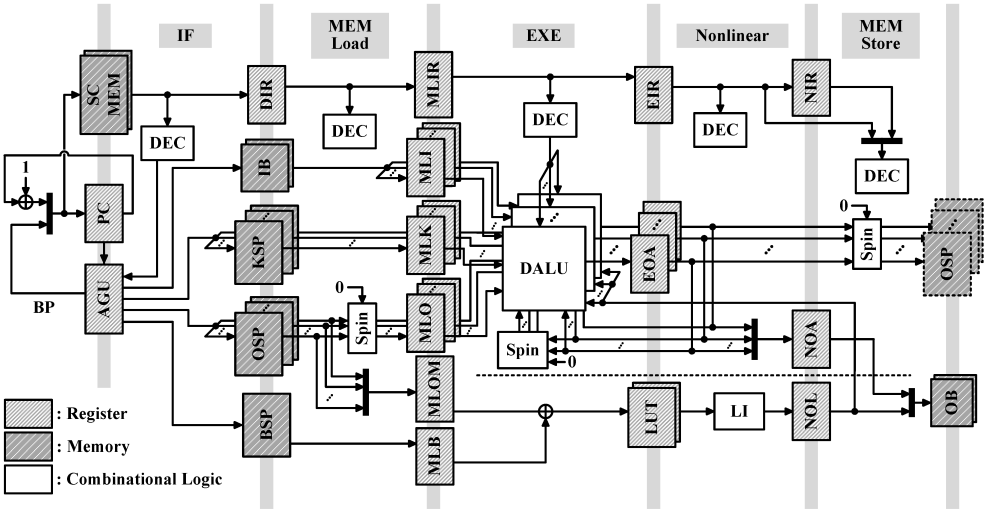

Fig. 15. Pipeline architecture of the SC.

be executed simultaneously. However, if the same processes occur consecutively, as in the case of executing several SCRs consecutively, the latter process must wait until the former operation is completed. The SCD, MLD, and MSD signals are generated by the SC and in/out DMAs to indicate the completion of the processes. The delayed command register (WDC) in the WB stage is used to store a newly entered memory access command if the current memory access operation is not completed. The register makes the next operations continue by bypassing the memory access command.

When the MC executes an SCR instruction, the SC performs a CNN operation stored in the SC MEM. A 3-D incremental stripe operation is executed in two sequential steps as illustrated in Fig. 8. Therefore, if there is only one SC MEM available, the time overhead needed to switch the two steps becomes critical, because they must be switched every *xk*-plane. To reduce the overhead, the two programs are separately stored in two SC MEMs.

Since the SC executes several instructions concurrently, it is developed by following the five-stage pipeline structure illustrated in Fig. 15. The first stage fetches a number of instructions and calculates the addresses and bank indices of the memories to be accessed. Each address is managed independently by the address generation unit (AGU). The PC indicating the address of the SC MEM is generally incremented by 1. For a loop operation, it jumps to the break point stored in the AGU. The bank indices of the memory to be accessed are calculated by performing the remainder operation which is achieved by assigning a separate LUT to each memory bank. By the nature of ISC, the maximum modulus is the square root of the number of banks, so the LUTs are very small.

The second stage loads data from the IB and scratch pads, and prepares the operations to be performed in the third stage. Note that in the proposed structure, only a single-port buffer and a spin logic are used in ISC unlike the output-oriented and input-oriented structures. The spin logic is in fact a set of two-input multiplexers, and it passes the outputs of

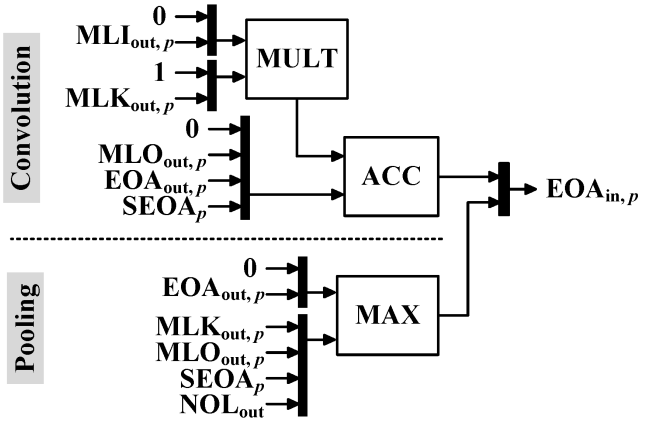

Fig. 16. Structure of the pth DSIP ALU ($DALU_p$).

the current or adjacent OSPs to MAC operators as depicted in Fig. 6.

In the third stage, there are parallel DSIP ALUs (DALUs) that perform convolution and max-pooling operations for the data received from the second stage. As shown in Fig. 16, each DALU contains MAC and MAX operators whose operands are specified in the arithmetic instruction. During an X-Loop, the spin logic conveys the partial-sum data of the current or adjacent registers to MAC operators as shown in Fig. 3.

The lower part of the third and fourth stages computes the activation function by adding bias values and performing the nonlinear operations stored in an LUT. In order to reduce the LUT, the function is approximated by adopting the piecewise linear interpolation [14]. Two LUTs are embedded to support two kinds of activation functions in a layer. The conventional architecture requires an LUT for each operator, whereas the proposed architecture has only two LUTs for the entire DALU units, as it performs the activation functions serially.

In the fifth stage, the results computed in the previous stage are stored into the OSP or OB. The OSP stores intermediate

TABLE IV
CHARACTERISTICS OF THE PROTOTYPE CHIP

Technology	65 nm 1P8M CMOS
Chip size	4 mm x 4 mm
Core area	3.2 mm x 3.3 mm
Number of PEs	64
Gate count (logic only)	282K (2-input NAND)
On-chip memory	139.6 KB
Supply voltage	Core: 1.2 V Link: 1.8 V
Maximum clock frequency	Core: 250 MHz Link: 100 MHz
Peak throughput	16 GMAC/s
Peak power consumption	153 mW (100 %)[a]
	106 mW (32 %)

[a]The numbers in parentheses are the percentage of active OSPs in parallel operators.

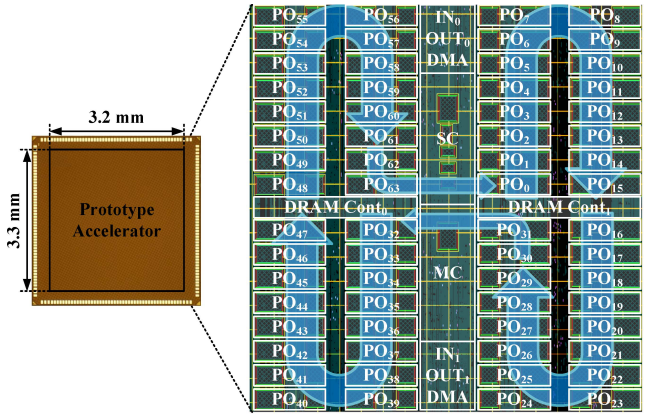

Fig. 17. Die micrograph and floorplan of the prototype DSIP.

results of CNN operations, whereas the OB stores the final results to be transmitted to the OM. The spin logic is utilized for the kernel decomposition.

Taking the five-stage pipeline structure, the SC can process the CNN operations fast. Once all the operations in the SC MEM are completed, the SC sends an SCD signal to the MC, and then either prepares to receive the next input features from the OM or sends another output features to the OM. To make this process faster, the double-buffering technique is again applied to the IB and OB, allowing the master–slave core to perform CNN operations while reading the next input feature from the OM simultaneously, and to pre-calculate the next output features while storing the current output features into the OM.

V. IMPLEMENTATION RESULTS AND EVALUATION

Based on the proposed DSIP structure, a prototype accelerator is implemented in a 65-nm CMOS process. The overall characteristics of the prototype chip are summarized in Table IV. Since the master–slave core is designed for precise inference systems, all the calculations are performed on 16-bit fixed-point operators. The proposed chip was fabricated by integrating 64 parallel operators, but note that more parallel operators can be included without changing the master–slave ISA due to the remainder operation supported in the SC. The buffers and scratch pads in the SC are set to 2 and 1 KB, respectively, so as to reuse the input and the partial-sum data efficiently. The master–slave core is 10.56 mm^2 in size, operates up to 250 MHz, and transfers 16-bit data at 100 MHz to the DRAM or other DSIPs. The peak power measured by activating all memory blocks is 153 mW. In practice, only a part of the memory banks is used in ISC. For example, only the third memory of a stripe is activated if the kernel size is 3, which reduces the peak power to 106 mW. The master–slave core is realized with 282k equivalent logic gates, where an equivalent gate means a two-input NAND gate. The MC and the SC, which occupy a significant portion of the overall logic, are realized with 42k and 203k gates, respectively.

The floorplan of the DSIP is shown in Fig. 17, where the MC is placed at the center, and the SC and external units are arranged around the MC. In Fig. 17, the pth parallel operator (PO_p) contains $DALU_p$, KSP_p, and OSP_p. Unlike the accelerators in [6]–[20], adjacent parallel operators in the SC are connected to reuse partial sums as shown in Fig. 15. As the circular operation can cause lengthy connections in routing rollback signals, parallel operators are arranged in an eight-character form to minimize the distance between adjacent operators.

To evaluate the proposed accelerator, AlexNet, which is one of the typical CNNs, was implemented. In the experiments, a batch of four image frames was processed to minimize the time to set up the layer configuration in the master–slave core. The AlexNet performance resulting from a single-chip DSIP is summarized in Table V, where $CONV_n$ indicates the nth convolutional layer. In the case of $CONV_1$, since the 11×11 kernel is bigger than the number of parallel operators available, the kernel was decomposed for ISC. Operating at 250 MHz, the single-chip DSIP takes 226.6 ms to process AlexNet and consumes 93.4 mW at 1.2 V.

The area and the power consumption of the prototype chip are analyzed by performing place-and-route simulations, which are summarized in Fig. 18. Since the MC is constructed based on a simple ISA, it only accounts for 5.2% of the total area. However, due to the large amount of memories, parallel operators including DALUs, scratch pads, and filler cells occupy 79.8% of the overall chip area. The power breakdown consumed in performing AlexNet is shown in Fig. 18(b). The results show that the SC takes about 50% of the total power consumption. The power can be minimized by adopting scalable precision techniques presented in [16] and [21], but the operation precision of the proposed SC is fixed to 16 bits in order to support diverse applications necessitating high precision.

Using the same hardware settings described in Tables IV and V, Fig. 19(a) shows the performance gain achieved by adopting the master–slave architecture. In the baseline accelerator, only an SC and control logics are embedded without the MC, the double input and OBs as in [6]–[21]. Since the data transmission and the CNN operations can be performed simultaneously in the master–slave core, the processing speed is enhanced by 1.5× compared to the baseline accelerator executing the two processes sequentially.

TABLE V
ALEXNET PERFORMANCE OF A SINGLE-DSIP SYSTEM FOR FOUR IMAGE FRAMES

Layer	Latency [ms]	Power [mW]	DRAM accesses [MB]		CONV configuration	
			Load[a]	Store	Kernel size	Active PO
$CONV_1$	64.3 (0.28)[b]	94.5	6.09	1.11	8×8	64
$CONV_2$	73.9 (0.33)	88.6	4.98	0.71	5×5	50
$CONV_3$	39.5 (0.18)	97.3	4.25	0.25	3×3	63
$CONV_4$	29.2 (0.12)	95.6	3.26	0.25	3×3	63
$CONV_5$	19.7 (0.09)	96.8	2.23	0.17	3×3	63
Total	226.6 (1)	93.4[c]	20.82	2.48	-	-

[a]It includes instructions, input features, filter weights, biases, and data for LUTs.
[b]The numbers in parentheses are normalized values.
[c]Average power consumption of the single DSIP.

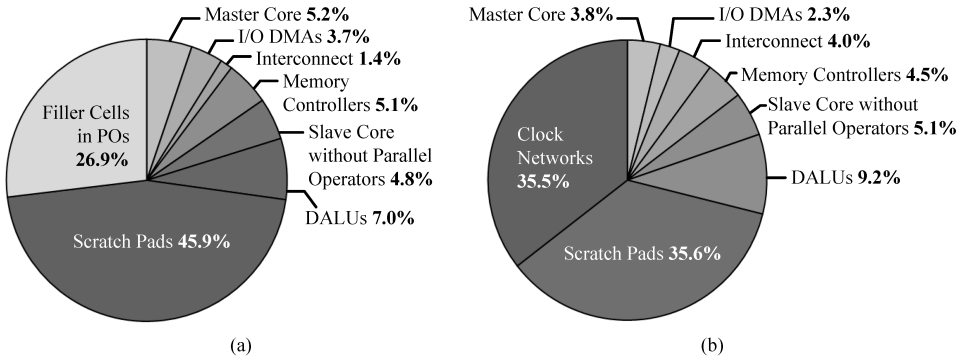

Fig. 18. Area and power analysis of the prototype DSIP. (a) Area breakdown. (b) Power breakdown.

The superiority of the incremental convolution is examined in terms of the amount of memory accesses, which includes accessing on-chip memories such as input, kernel, and OBs, and the simulation results for AlexNet $CONV_4$ are summarized in Fig. 19(b). To match the processing speed of the accelerators, the input-oriented structure [8], the output-oriented structure [7], and the proposed structure are configured with 168, 180 (five convolution accelerator units in [7]), and 162 MAC units, respectively. As exemplified in Fig. 5, the processing flow of the proposed structure is similar to that of the output-oriented structure except for input loading patterns, reducing the amount of memory accesses by 2.4× compared to the output-oriented structure. In the case of the input-oriented structure, the number of accesses to the input and kernel buffer is reduced to 37.4 MB [8], but those data are stored into the input and KSP memories in order to reuse them. Since both the input and the kernel data must be reloaded every cycle in each PE as depicted in Fig. 5, the amount of memory accesses required in the input-oriented structure is 5.8× larger than that of the proposed structure.

To investigate the performance enhancement by chaining DSIPs, post-layout simulations were conducted for various types of multiple-DSIP systems. In the energy estimation, we utilized CACTI-IO [23] to include the power consumed by inter-DSIP communication operating at 100 MHz at 1.8 V.

The performance improved by multiple DSIPs chained in the vertical direction is presented in Fig. 20(a), which shows that chaining 2 and 3 DSIPs leads to 1.94× and 2.79× speedup, respectively. As the load processes can be parallelized by chaining DSIPs as depicted in Fig. 9, the amount of memory accesses can be decreased by the number of DSIPs. However, the store operations are serially performed because the OM is shared, dominating the processing time when chaining multiple DSIPs. This effect is shown in Fig. 20(b) by presenting the activated percentage of the master–slave core. The activated time of the MC becomes almost full when connecting more than 3 DSIPs in the vertical direction, which means that the speedup of the multiple-DSIP system is saturated. The energy overhead induced by inter-DSIP communication is presented in Fig. 20(c), which reveals that, the total energy consumption is increased by 1.08× and 1.17× when chaining 2 and 3 DSIPs, respectively.

The latencies and power consumptions of multiple-DSIP systems for AlexNet are summarized in Table VI. The chain configuration lists the number of DSIPs involved in a

TABLE VI
ALEXNET PERFORMANCE OF MULTIPLE-DSIP SYSTEMS FOR FOUR IMAGE FRAMES

Chain configuration	Embedded layers	Latency [ms]	Power [mW][a]
1-1-1	$CONV_1$	64.3	101.2
	$CONV_2$	73.9	94.6
	$CONV_3$, $CONV_4$, $CONV_5$	88.4	104.3
	Total	88.4[b]	285.1[c]
2-2	$CONV_1$, $CONV_2$	69.3	199.1
	$CONV_3$, $CONV_4$, $CONV_5$	45.1	212.7
	Total	69.3[b]	382.5[c]

[a]Power consumption of a vertical set of DSIPs when running CONVs.
[b]Critical latency.
[c]Power consumption of a multiple-DSIP system.

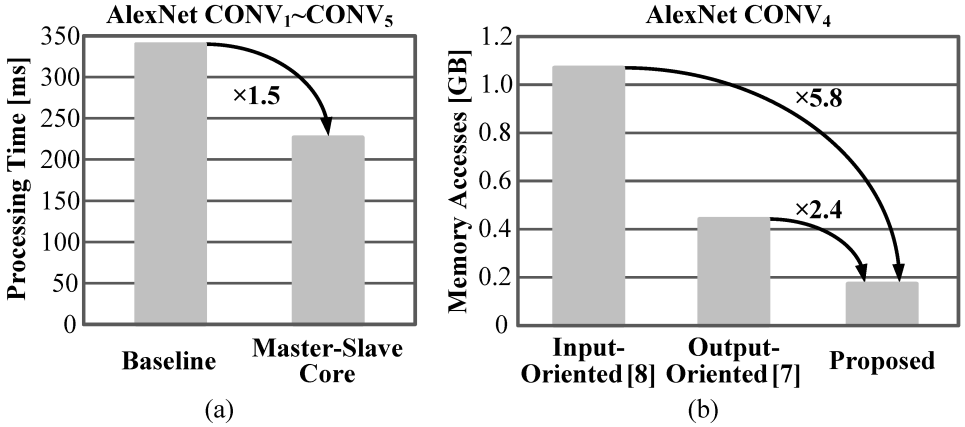

Fig. 19. AlexNet performance enhancements for four image frames by employing (a) master–slave core and (b) incremental convolution.

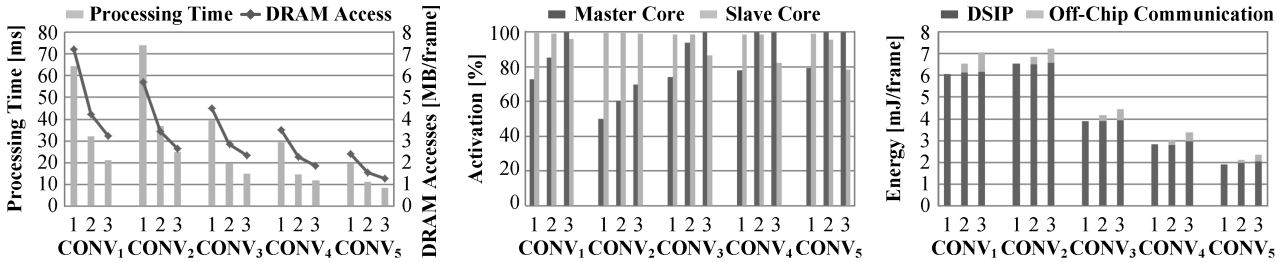

Fig. 20. AlexNet performances of 1-, 2-, 3-DSIP systems for four image frames. (a) Processing time and DRAM accesses. (b) Activation percentage. (c) Energy consumption.

TABLE VII
COMPARISON OF CNN-BASED INFERENCE SYSTEMS

		This work		[7]	[8]	[16]
Number system		16-bit fixed-point		16-bit fixed-point	16-bit fixed-point	16-bit fixed-point
Technology [nm]		65		28	65	40
Voltage [V]		1.2		0.575	1	0.9
Clock frequency [MHz]		250		200	200	204
Number of MACs		64		288	168	256
Scalable architecture		Yes		No	No	No
AlexNet	Neural network	$CONV_1$~$CONV_5$	$CONV_2$[a]	$CONV_1$~$CONV_5$	$CONV_1$~$CONV_5$	$CONV_2$[a]
	Throughput [GMACS]	Single: 11.7, 1DC 1-1-1: 30.1, 2DC 2-2: 38.4	Single: 12.1	38.9	23.1	52.2[b]
	Power [mW]	Single: 93.4, 1DC 1-1-1: 285.1, 2DC 2-2: 382.5	Single: 88.6	61	278	274
	Absolute energy efficiency [GMACS/W]	Single: 125.4, 1DC 1-1-1: 105.7, 2DC 2-2: 100.5	Single: 136.8	638.2	83.1	190.6[b]
	Off-chip accesses [MB]	1DC 1-1-1: 1.9, 2DC 2-2: 11.9	-	-	-	-
	Off-chip access energy consumption [mJ]	1DC 1-1-1: 0.8, 2DC 2-2: 4.7	-	-	-	-
	Normalized energy efficiency[c] [GMACS/W]	Single: 125.4, 1DC 1-1-1: 105.7, 2DC 2-2: 100.5	Single: 136.8	63.1	57.7	66.0[b]

[a]Only the second convolutional layer of AlexNet is taken into account due to the mismatch in the fixed-point precision.
[b]The performance is obtained by using the maximum throughput due to the lack of information: throughput = 204 MAC × 256 MHz.
[c]The energy efficiency is normalized to our 65nm process with 1.2V supply voltage by the following equation:
normalized energy efficiency = absolute energy efficiency × (technology / 65 nm) × (voltage / 1.2 V)^2 [24].

vertical set. The chain configuration of Fig. 4(b), for example, starts with 3-2-, as the 1st and 2nd vertical sets are composed of 3 and 2 DSIPs, respectively. The embedded layers represent the layer types processed in each vertical set. It is desired in increasing the processing speed to allocate layers evenly to DSIPs so that their latencies are similar, since a layer can start its operation only when the previous layer finishes its working. For this reason, the largest latency called the critical latency

should be minimized. Due to the latency differences among vertical sets, a DSIP that finishes its operation has to wait for the completion of other DSIPs in the idle state. Therefore, power consumption increases slightly as the number of DSIPs increases.

Table VII compares the processing speed and the energy efficiency of diverse CNN accelerators for AlexNet. The proposed DSIP that separates the master and the slave can increase the processing speed by overlapping different processes in a CNN. Moreover, ISC enables the single-chip DSIP to reduce the power consumption. As a result, the single-chip DSIP enhances the energy efficiency by a factor of 2.17 even compared to the state-of-the-art accelerator [8]. To increase the processing speed, 1DC and 2DC structures are applied at the expense of energy consumption. The additional energy is mainly induced by the latency mismatch between vertical sets of DSIPs and the off-chip communication between DSIPs as described in Tables VI and VII, respectively. In total, the energy efficiency of the 1DC and 2DC structures is degraded by 1.19× and 1.25× while increasing the throughput by 2.56× and 3.26× compared to those of the single-DSIP, respectively.

VI. Conclusion

In this paper, we have presented a scalable inference accelerator designed to support complex CNNs. The proposed accelerator called a DSIP has two ISAs: a master ISA and a slave ISA. Dividing the functions of data transmission and computation into two heterogeneous ISAs, the proposed accelerator can achieve a high processing speed. In addition, the proposed ISC makes it possible to reduce power consumption by replacing a multi-port input scratch pad and complex multiplexers with a single-port buffer. The improvement of power consumption is achieved without degrading the speed. Furthermore, the DSIP can be extended to 2-D structure in order to enhance speed. Due to the scalable architecture, a DSIP-based system that can accommodate complex neural networks is constructed by connecting multiple DSIPs.

To evaluate the superiority of the DSIP architecture, a prototype chip was designed and used to implement AlexNet [1]. The DSIP-based system achieved 2.17× higher efficiency than that of the state-of-the-art accelerator [8]. Note that only one DSIP type was used in the evaluation. As each layer is associated with a different amount of computations in a typical neural network, the efficiency can be improved by constructing a 2DC structure with various DSIP types. Since the master–slave ISA supports the remainder operation to control parallel operations, the same ISA can be used to program various DSIPs with different numbers of PEs.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2015, pp. 1–14.

[3] A. Coates, B. Huval, T. Wang, D. J. Wu, and Y. A. Ng, "Deep learning with COTS HPC systems," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, Jun. 2013, pp. 1337–1345.

[4] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn.*, Dec. 2011, pp. 1–8.

[5] "GPU-based deep learning inference: A performance and power analysis," NVIDIA, Santa Clara, CA, USA, White Paper, Nov. 2015. [Online]. Available: http://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf

[6] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origmai: A convolutional neural network," in *Proc. 25th Great Lakes Symp. (VLSI)*, May 2015, pp. 199–204.

[7] G. Desoli *et al.*, "A 2.9 TOPS/W deep convolutional neural network SoC in FD-SOI 28 nm for intelligent embedded systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 238–240.

[8] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[9] S. Park, K. Bong, D. Shin, J. Lee, S. Choi, and H.-J. Yoo, "A 1.93 TOPS/W scalable deep learning/inference processor with tetra-parallel MIMD architecture for big-data applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2015, pp. 80–82.

[10] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proc. 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 1–13.

[11] D. Kim, J. Ahn, and S. Yoo, "A novel zero weight/activation-aware hardware architecture of convolutional neural network," in *Proc. IEEE Des. Autom. Test Eur. (DATE)*, Mar. 2017, pp. 1462–1467.

[12] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim, and H.-J. Yoo, "A 0.62 mW ultra-low-power convolutional-neural-network face-recognition processor and a CIS integrated with always-on Haar-like face detector," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 248–250.

[13] Z. Du *et al.*, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. 42nd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2015, pp. 92–104.

[14] T. Chen *et al.*, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proc. 19th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Mar. 2014, pp. 269–284.

[15] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10 TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 246–248.

[16] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.

[17] S. Wang, D. Zhou, X. Han, and T. Yoshimura, "Chain-NN: An energy-efficient 1D chain architecture for accelerating deep convolutional neural networks," in *Proc. IEEE Des. Autom. Test Eur. (DATE)*, Mar. 2017, pp. 1032–1037.

[18] A. Pullini, F. Conti, D. Rossi, I. Loi, M. Gautschi, and L. Benini, "A heterogeneous multi-core system-on-chip for energy efficient brain inspired computing," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, to be published, [Online]. Available: http://ieeexplore.ieee.org/document/7817777/ doi: 10.1109/TCSII.2017.2652982.

[19] F. Conti *et al.*, "An IoT endpoint system-on-chip for secure and energy-efficient near-sensor analytics," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2481–2494, Sep. 2017.

[20] S. Liu *et al.*, "Cambricon: An instruction set architecture for neural networks," in *Proc. 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 393–405.

[21] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 240–242.

[22] M. M. Khan *et al.*, "SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2008, pp. 2849–2856.

[23] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas, "CACTI-IO: CACTI with off-chip power-area-timing models," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 7, pp. 1254–1267, Jul. 2017.

[24] I. Yoo, B. Kim, and I.-C. Park, "Reverse rate matching for low-power LTE-advanced turbo decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 12, pp. 2920–2928, Dec. 2015.

Jihyuck Jo (S'13) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2012 and 2014, respectively, where he is currently pursuing the Ph.D. degree with the School of Electrical Engineering.

Since 2012, he has been a Research Assistant with the KAIST, where he has developed high-performance microprocessors and embedded system-on-chip platforms. His current research interests include very large scale integration architectures for neural network accelerators and general-purpose microprocessors.

Soyoung Cha (S'13) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2008 and 2014, respectively, and the M.S. degree from the Graduate School of Culture Technology, KAIST, in 2010, where she is currently pursuing the Ph.D. degree.

Her current research interests include energy-efficient very large scale integration system design, computer vision, and digital signal processing.

Dayoung Rho (S'17) received the B.S. degree in electrical engineering from Hanyang University, Ansan, South Korea, in 2016. She is currently pursuing the M.S. degree with the Division of Future Vehicle, Korea Advanced Institute of Science and Technology, Daejeon, South Korea.

Her current research interests include the design of very large scale integration architectures for neural network and error-correcting circuits.

In-Cheol Park (S'88–M'92–SM'02) received the B.S. degree in electronic engineering from Seoul National University, Seoul, South Korea, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1988 and 1992, respectively.

From 1995 to 1996, he was with the IBM T. J. Watson Research Center, Yorktown, NY, USA, where he was involved in high-speed circuit design. Since 1996, he has been an Assistant Professor and is currently a Professor with the School of Electrical Engineering, KAIST. His current research interests include computer-aided design algorithms for high-level synthesis and very large scale integration architectures for general-purpose microprocessors.

Prof. Park received the Best Design Award at ASP-DAC in 1997 and the Best Paper Award at ICCD in 1999.