IJICTR
International Journal of Information & Communication Technology Research

Volume 16- Number 4 – 2024 (9 -19)

FUCA: a Frame to Prevent the Generation of Useless results in the Dataflows Based on Cartesian Product for Convolutional Neural Network Accelerators

Babak Narimanjahan
Department of Computer Engineering
Bonab Branch, Islamic Azad University
Bonab, Iran
nariman@bonabiau.ac.ir

Ahmad Khademzadeh*
Iran Telecommunication Research Center (ITRC)
Tehran, Iran
zadeh@itrc.ac.ir

Akram Reza
Department of Computer Engineering
Shahr-e-Qods Branch, Islamic Azad University
Tehran, Iran
a.reza@qodsiau.ac.ir

Received: 9 July 2024 – Revised: 15 August 2024 - Accepted: 22 September 2024

Abstract—**One of the most important issues in the design of CNN accelerators pertains to the accelerator's ability to effectively leverage the available opportunities in the type and processing of input data, and the task of achieving this objective mostly lies with the dataflow. Equal channel size in the input feature map and filter of CNNs is one of these opportunities, which makes it desirable to design dataflow as Channel Dimension Stationary (CDS). On the other hand, the complexity of designing computations based on the Cartesian product (due to its all-to-all nature) is lower, especially in CDS dataflows. But, since the Cartesian product method causes the generation of useless products and, as a result, reduces performance and energy efficiency, there is less desire for this type of design. This paper presents a frame called FUCA for Cartesian product-based dataflows, which avoids operations leading to useless products. The analysis revealed that FUCA reduces runtime and energy consumption in the Cartesian product-based dataflow by 1.5x, potentially surpassing the sliding window-based dataflow.**

Keywords: useless product, zero-padding, Channel Dimension Stationary (CDS), Cartesian Product based Convolution (CPC), MAERI accelerator, frame

Article type: Research Article

Publisher: ICT Research Institute

[DOI: 10.61186/itrc.16.4.9]

I. INTRODUCTION

In today's applications, deep neural networks are frequently employed for processes such as machine vision, speech recognition, and classification [1-3]. Using the Convolution Neural Network (CNN) in various Artificial Intelligence (AI) applications, Deep Learning has produced results with excellent accuracy [4]. However, these algorithms face challenges in real-time applications due to the high volume of input parameters and computations required. Therefore, CNN's hardware accelerators have emerged, and research to develop them continues [5].

* Corresponding Author

Dataflow is one of the crucial aspects in the design of accelerators, as it greatly influences the utilization of processing elements, performance, and energy efficiency [6]. In hardware accelerators that focus on exploiting opportunities such as data reuse (Eyriss [7]), sparsity (NullHop [8] and SCNN [9]), parallel computing (almost all accelerators), etc., most of this happens through dataflow optimization.

In convolutional layers, an interesting characteristic is that the number of filter channels (C) is equal to the number of input feature map (ifmap) channels (Fig. 1). Crucially, the products obtained by multiplying one Channel Dimension Array (CDA) of ifmap by one CDA of filter all pertain to a single value of output feature map (ofmap) and must be gathered together. Therefore, the design of the accelerator's dataflow as the Channel Dimension Stationary(CDS) can significantly reduce implementation complexity [10]. In particular, this approach can also enhance the exploitation of the sparsity opportunity. For instance, by adopting the same technique and without adding hardware overhead, the MCPS [10] dataflow, which is based on CDS and focuses on exploiting sparsity, has been able to increase performance by 2.9x and energy efficiency by 2.11x at a sparsity of 70%.

The computational method in convolution-based dataflows is the next issue. This can be either Sliding Windows based Convolution (SWC) or Cartesian Product based Convolution (CPC), depending on the main goal of the accelerator. For example, dataflow in MAERI [11] and Eyeriss [7] is based on SWC, while dataflow in SCNN [9] and MCPS [10] is based on CPC.

Applying the CPC method in the dataflow, particularly in the CDS dataflow, due to the simpler algorithm (the nature of all to all), can reduce the complexity of the implementation. Rather, in CPC, there is a challenge of generating useless products that manifest as overheads in computation and data transfer, adversely affecting both energy efficiency and performance. Due to this issue, the SWC is now more often employed in accelerators than the CPC.

In the current study, we concentrated on CPC-based dataflows for CNN accelerators and attempted to create a constraining frame called FUCA for this kind of computation, so as to avoid the transfer and processing of data leading to useless results. To evaluate the FUCA, we have chosen the MAERI [11] as the target accelerator. This accelerator is reconfigurable and can be configured based on various dataflows. With loop transformations and tiling, we have designed a CDS dataflow for MAERI that is based on CPC. Then, this designed dataflow was upgraded based on FUCA in several steps to prevent the generation of useless products and overheads.

Due to the nature of computations in CPC-based dataflow, it is simple to ignore channels with any position on the ifmap plane. Therefore, FUCA does not send the channels produced by zero-padding for computations. Furthermore, because FUCA also avoids sending channels that lead to useless products, the volume of computations is significantly reduced compared to the case without FUCA. Tests demonstrated that the suggested frame significantly lowers runtime and energy consumption when it applies to CPC-based dataflows. Especially in layers that have high zero-padding, this is a multi-fold reduction.

Compared to SWC-based dataflows, the computation volume of the proposed method is less when the number of zero-padding (ZP) > 0. This makes the performance and energy efficiency of the provided dataflow—which is CPC-based and enhanced by FUCA—better than the original MAERI dataflow (which is even improved by various techniques) in at least half of the cases.

The proposed idea can be used to optimize various dataflows, particularly CDS dataflows that are based on CPC, to enhance performance and reduce energy consumption.

The key contributions of this work are the following:

- The nature of all-to-all in CPC causes the generation of useless results. We offer a frame for CPC-based dataflows called FUCA, which avoids transferring and mapping data, leading to useless products.
- In order to improve performance, we strengthen FUCA to prevent the processing of zeros resulting from zero-padding.
- To evaluate the presented frame, we establish a CDS dataflow based on CPC for the MAERI accelerator. This dataflow enables the multiplication of all-to-all between ifmap and filter.
- Our study demonstrates that employing FUCA in dataflows based on CPC can effectively decrease both runtime and energy consumption.

The subsequent sections of this work are structured in the following manner. Section II presents a concise explanation of CNN accelerators and the MAERI architecture. Section III explores the CPC and how it generates useless products. The algorithm of the designed dataflow for the FUCA test is explained in Section IV. Section V provides a comprehensive description of the proposed frame and how it is applied to a CPC-based dataflow. Section VI includes evaluation measures, methodologies, and their corresponding results. Lastly, Section VII offers the conclusion.

II. BACKGROUND

To further comprehend the details, an introduction to the MAERI accelerator—the target accelerator in this work—is required. Thus, this section provides a summary of MAERI's architecture together with a brief overview of CNNs.

A. CNNs and CNN Accelerators

Notably, modern CNNs demand billions of multiply-accumulate (MAC) operations [12]. Each CNN layer's large computational volume is the reason for this huge amount of computation. In addition, the number of layers in advanced CNNs is also continuously increasing to improve accuracy [7, 13, 14]. For instance, by adding more layers, CNNs like GoogleNet [15], ResNet50 [16], DenseNet [17], and various versions of YOLO [18-21] have expanded the network.

[DOI: 10.61186/itrc.16.4.9]

CNNs generally include two main kinds of layers: convolutional layers and fully connected layers. Usually, the number of convolutional layers in CNNs is much higher. Thus, the majority of the work in CNNs is done by convolutional layers [22].

Each convolutional layer in CNN performs computations by applying filters to ifmaps in order to extract features and generate ofmaps [7]. Ifmaps, ofmaps, and filters have several dimensions in each convolutional layer (typically four dimensions). The dimensions and computations of a convolutional layer are shown in Fig. 1. Also, Fig. 3 shows a simpler and more accurate example of convolution layer computations.

While GPUs can be utilized for training CNNs, they are inefficient when used for inference in applications, particularly on energy-constrained mobile devices [8, 9]. Thus, to deal with this issue, CNN accelerators have been developed to effectively process enormous data volumes while minimizing energy consumption and delay. These accelerators incorporate a large number of MAC units. For example, the google Tensor Processing Unit (TPU) [23] consists of 64K MACs. The energy management and parallelization of this large number of MACs, efficient data transfer management between MACs, effective delay management and energy consumption optimization across memory hierarchies, and adaptability to various types of layers are among the challenges that accelerators face, and work on them continues [12].

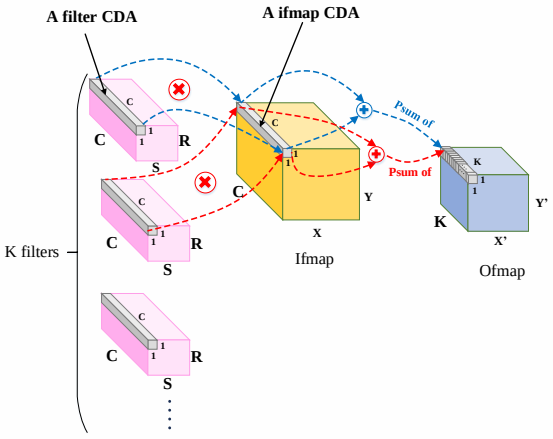
The efficiency of a spatial accelerator is significantly influenced by its mapping and dataflow [24]. Hence, some reconfigurable accelerators are engineered to adapt to diverse dataflows, with the aim of enhancing both performance and energy efficiency across various CNNs. One of these accelerators is MAERI, which is the target accelerator in the current work. Its structure is described below.

B. MAERI accelerator

MAERI [11] is the name of the accelerator developed by Kwon et al. Its main goal is reconfigurability to support various dataflow patterns. This accelerator has targeted the connections and structure of the processing elements. The MAERI architecture is depicted in Fig. 2.

Three networks—the Distribute Network (DN), Multiplier Network (MN), and Reduce Network (RN)—make MAERI's Network On Chip (NOC), as shown in Fig. 2. DN and RN networks are built based on tree topology, and the MN network has a linear topology. The role of the DN tree is to transfer the ifmap values and weights from the prefetch buffer to the MN. On the other hand, the RN is responsible for accumulating the MN products at several levels to generate the ofmap values or psums. The computed values of ofmap are first modified by activation units, such as Rectified Linear Units (ReLU), and subsequently stored back into the prefetch buffer.

VN construction. The major characteristic of MAERI is the Virtual Neuron (VN) construction, which involves the segmentation of MAERI's NOC and the setting of the MN, RN, and DN according to this division. Every VN, independently, concurrently, and parallelly with other VNs, executes the necessary MAC operations to generate a single ofmap value or psum.



N	**Number of ifmaps in batch/Number of Ofmaps (In this figure N=1)**
K	**Number of filters**
C	**Number of channels(plans) in per ifmap and filter**
X/Y	**Number of rows/Number of columns in ifmap plane**
R/S	**Number of rows/Number of columns in filter plane**
X'/Y'	**Number of rows/Number of columns in Ofmap plane**

Figure 1. Convolutional layer dimensions and operations [7, 10, 25]

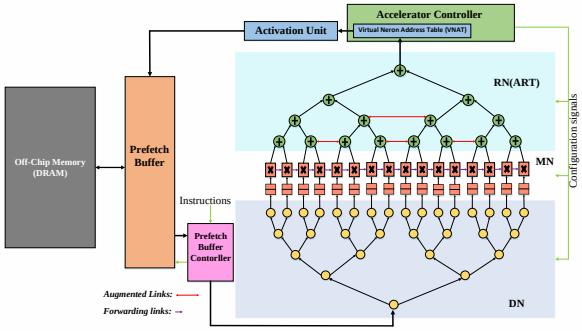


Figure 2. The MAERI accelerator's architecture [11]

A simple example of VN construction. A simple convolution layer with the dimensions $N = 1$, $X = 4$, $Y = 4$, $C = 2$, $R = 2$, $S = 3$, $K = 2$, $X' = 3$, and $Y' = 2$ is shown in Fig. 3. Fig. 4 depicts the mapping of the same layer on MAERI, providing a clear visual representation of MAERI's primary mapping method, which is based on SWC. It also illustrates VN Construction. In this case, the mapping strategy is according to the $VN_{size} = R \times S \times C$ [1] and the NOC has been set up in accordance. As shown, the MN has been divided into VN_{number} VNs based on equation 1 [11]. The MS_{number} represents all of the Multiplier Switches (MSes) in the MN.

[1] *R*, *S*, and *C* are the dimensions of a single filter, and VN_{size} is the number of MSes in each VN.

[DOI: 10.61186/itrc.16.4.9]

$$VN_{number} = \left\lfloor \frac{MS_{number}}{VN_{size}} \right\rfloor \quad (1)$$

Since *C* is equal to 2, the VN_{size} has been determined to be 12, which is equal to the total number of weights of the one filter's planes. In this example, the dataflow is Weight Stationary (WS), which means that the weights are initially transferred to and kept in MSes. Subsequently, fresh ifmap values are inserted into MSes in each cycle. For example, following the depicted cycle, the next window values from the two ifmap channels (*a2*, *a3*, *a4*, *a6*, *a7*, *a8*, *b2*, *b3*, *b4*, *b6*, *b7*, *b8*) will be transmitted to MSes for computation.

III. GENERATION OF USELESS PRODUCTS IN CPC

The CPC-based convolution involves multiplying all possible combinations. This method is simple but generates useless products, resulting in computational and transmission overheads.

A. *The Side*

A portion of the ifmap plane is defined by the term "Side" in this paper. It is described as follows: "All multiplications of the ifmap values that are not on the Side are certainly useful, but multiplying some of the ifmap values on the Side may lead to useless products". In the CPC-based convolution operation, depending on the dimensions of the filter plane, one or more rounds of the ifmap plane edges are the Side. In Fig. 5(a), the filter plane is 2 × 2. This has caused one round of ifmap plane edges to be the Side. Increasing the size of the filter will result in a larger Side size. For example, in a convolution operation with 7 × 7 filter plane dimensions, six rounds from the ifmap plane edges are the Side. Fig. 5(a) illustrates an example of CPC-based convolution. As seen, only 12 of the 32 products of ifmap values that are on the Side are useful. In this example, only the A5 is not on the Side, and all its products have become useful.

B. *The impact of zero-padding on the quantity of useless products*

The majority of convolution layers have zero-padding. In these types of layers, if the CPC operation is modified to eliminate the computation of additional data caused by zero-padding, the number of useless products will be reduced. Fortunately, during CPC operations, it is simple to avoid computing ifmap values that result from zero-padding.

Fig. 5(b) demonstrates the impact of ignoring added data by zero-padding. In this example, *ZP* is 1, and the multiplication of its values has been skipped. This has caused the number of useless products to be reduced to zero.

When this approach is applied in CPC, the number of multiplications needed is often even lower than when the SWC method is used. For example, the SWC method requires 64 multiplications in the layer of Fig. 5(b), whereas the CPC with the mentioned approach only requires 36 multiplications.

Equation 2, where $R = S$; $K = 1$; $C = 1$; $N = 1$, determines the number of useless products in the CPC operation when zero-padding values are ignored. Fig. 1 describes *R*, *S*, *K*, *C*, and *N*. *X* represents the width and height of the ifmap plane after removing the zero-padding data. This equation clearly shows that as *ZP* increases, the number of useless products decreases.

$$\text{Useless}_{Number} = 4 \times \left(\frac{(R-ZP)(R-ZP-1)}{2} \times R \times X \right) - 4 \times \left(\frac{(R-ZP)(R-ZP-1)}{2} \right)^2 \quad (2)$$

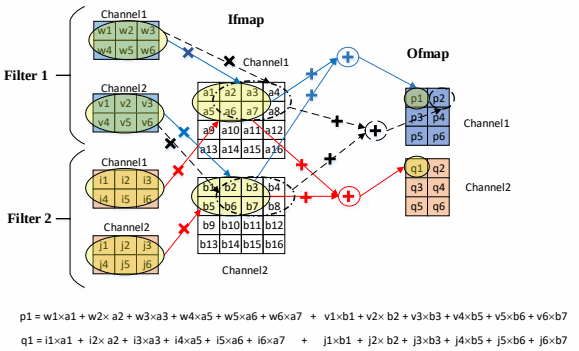


Figure 3. The computations in a simple example of a convolutional layer

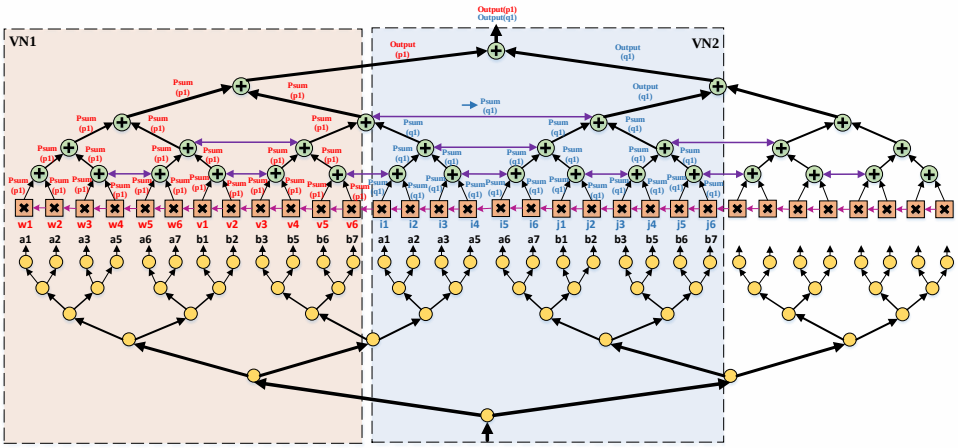


Figure 4. mapping over MAERI and VN Construction (mapping the example layer of Fig. 3 based on the main dataflow of MAERI)

[DOI: 10.61186/itrc.16.4.9]

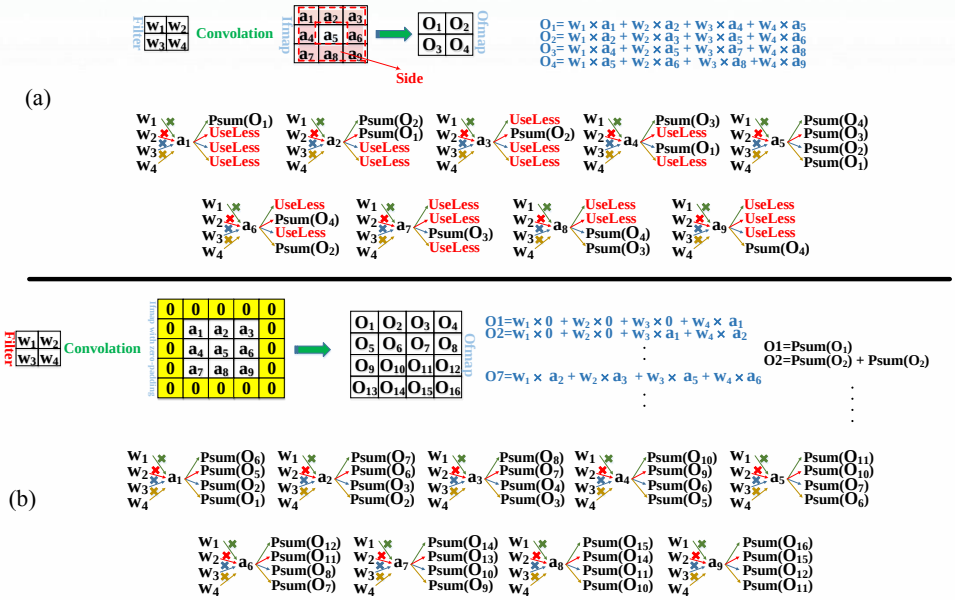


Figure 5. An illustration of CPC operation. **(a)** CPC with full computations. **(b)** CPC with ignoring computations of added zeros by zero-padding

IV. Implementing a CDS Dataflow Based on CPC to Test the Proposed Frame

To evaluate the proposed frame aimed at improving dataflows based on the Cartesian product, we have designed a new data flow for the MAERI architecture. To maintain simplicity and focus on the study's core theme, this data flow's chosen type is CDS. Because to implement the computations of CPC operation, exploiting channel features can significantly reduce design complexity. In the designed dataflow, we have taken advantage of two of these features. First, the number of filter channels (C) in convolutional layers is identical to the number of ifmap channels. Second, the psums produced by multiplying a Channel Dimension Array (CDA) of ifmap by a CDA of filter all belong to one ofmap value and need to be summed. Fig. 1 depicts this feature along with ifmap CDA and filter CDA.

The designed dataflow is shown in Fig. 6. The description of this dataflow can be summarized as follows: The CPC operation is completed during processes whose number is equal to the number of Ifmap CDAs. Each process executes and finishes all computations relating to a single ifmap CDA.

Initially, MAERI is configured based on the channel length (C), and VNs are established. Following the operation's commencement, each process begins with multicasting the current ifmap CDA to all VNs. Subsequently, in each following cycle, co-location CDAs of multiple filters are unicasted[1] to VNs for multiplication. Once all the filter CDAs are sent, the operation proceeds with the next process.

Equation 3 provides a mathematical representation of the convolution layer's operation based on the designed dataflow. As it is known, psums are produced by multiplying and accumulating filter CDAs in the ifmap CDAs, and then co-location psums are added together to produce one ofmap value.

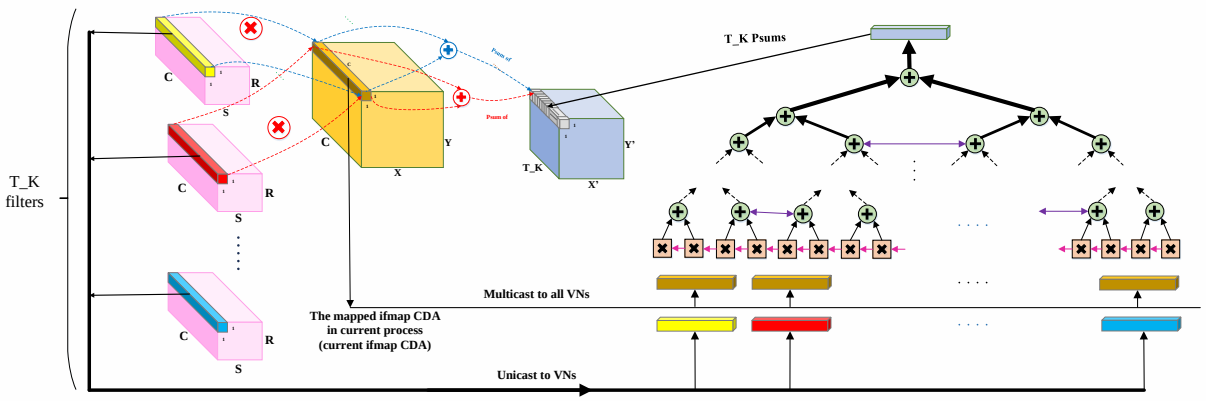


Figure 6. The designed CPC-based dataflow to test the FUCA (designed as CDS)

[1] i.e., each filter CDA is transmitted to one VN.

[DOI: 10.61186/itrc.16.4.9]

$$psum(n,k,x-r,y-s) = \sum_{c=0}^{C-1} W(k,c,r,s) \times I(n,c,x,y) \quad (3)$$

$$O(n,k,x',y') = \sum all\ of\ the\ psums\ with\ position(n,k,x',y')$$

$$0 \le n < N, \quad 0 \le k < K, \quad 0 \le x < X,$$
$$0 \le y < Y, \quad 0 \le r < R, \quad 0 \le s < S,$$
$$0 \le x' < X', \quad X' = X - R + 1$$
$$0 \le y' < Y', \quad Y' = Y - S + 1$$

Of course, in order to simplify the equation, the stride equal to 1 has been considered, and the ReLU function and the array of biases have not been applied. The parameters used in this equation are described in Fig. 1.

V. THE PROPOSED FRAME (FUCA)

This section describes the method of developing and implementing the proposed frame over the designed CPC-based dataflow. It begins by presenting the designed dataflow as pseudo-code and then proceeds with applying the FUCA in several steps.

A. *Pseudo-code of designed dataflow*

Fig. 7 shows the evolution of the main idea of the current work in three sequential segments. The pseudo-codes of the designed dataflow without any prevention of unnecessary and inefficient computations are depicted in the first segment (Figures 7(a) and 7(b)). In the second segment (Figures 7(c) and 7(d)), the dataflow of the previous segment has been modified to prevent the computation of added zeros by zero-padding. Lastly, the dataflow has been modified in the third segment (Figures 7(e) and 7(f)) such that, in addition to eliminating computations involving zero-padding zeros, it also avoids computations resulting in useless products; in other words, the dataflow has been fully equipped with FUCA.

The first segment of Fig. 7 only shows the pseudo-codes of the designed dataflow, where the tiling and transformations applied over nested loops of the convolution layer to achieve the designed dataflow can be seen. As visible, the loops of dimensions *K* and *C* have moved to the lowest level. This is because the designed dataflow operates on a CPC basis, where all ifmap CDAs are multiplied by all filter CDAs. In the presented dataflow, tiling is applied only to dimensions *K* or *C*. In most layers, due to the limited number of MSes, it is not possible to complete the computations of all filter CDAs with the current ifmap CDA in a single cycle. Therefore, tiling is applied to dimension *K* so that a portion of the filter CDAs is sent to the MSes in each cycle. Also, it is possible for the channel length (C) in some layers to be longer than the number of MSes, in which case even the computation of one filter CDA cannot be completed in a single cycle. For these situations, tiling has been applied to dimension *C*. Fortunately, MAERI's reconfigurability allows us to apply tiling to the different dimensions. In pseudo-codes 7(a), 7(c), and 7(e), tiling has been applied to the dimension *K* and is for layers in which $C < MS_{number}$. Also, pseudo-codes 7(b), 7(d), and 7(f) are equivalent to Figures 7(a), 7(c), and 7(e), respectively, for layers in which $C > MS_{number}$. In these pseudo-codes, tiling has been applied to dimension *C*.

The red dotted part in Fig. 7(a) illustrates the computations of each cycle. As clear, during each cycle, computations occur out on one ifmap CDA and multiple CDAs from various filters. The dataflow's CPC-based structure has resulted in the offsets $+r$ and $+s$ being negative, and, together with d[1], they have been transferred from I[*n*][*c*][*x*][*y*] to O[*n*][*k*][*x*][*y*].

In the second segment of Fig. 7, we provided another dataflow to better demonstrate the impact of the FUCA. This dataflow performs similarly to pseudo-codes 7(a) and 7(b), but it ignores computations of zero values resulting from zero-padding. This can be considered the first step in improving the CPC-based dataflow. Because preventing the transmission and computation of ineffective data will save energy and reduce runtime. Based on this, Figures 7(c) and 7(d) are improvements of pseudo-codes 7(a) and 7(b), respectively.

B. *The FUCA's foundation and applying it to CPC-based dataflow*

As mentioned in the description of CPC (Section III), if an ifmap value located at position (*n*, *x*, *y*) relative to R and S is on the Side of the ifmap plane, multiplying that ifmap value by a number of weights will result in useless products. Here, we have articulated this issue in a formulaic manner. In a way, the result of multiplying an ifmap value at position (*n*, *x*, *y*) by a filter weight at position (*r*, *s*) will be useless if at least one of the following conditions is met (where the zero-padding zeros are ignored in the computations).

$$(x - r + ZP < 0) \quad (4)$$
$$(y - s + ZP < 0)$$
$$(x - r > X - R + ZP)$$
$$(y - s > Y - S + ZP)$$

In the convolution layer, if $K = 1$ and $C = 1$, multiplying the position (*n*, *x*, *y*) by the position (*r*, *s*) under the above conditions results in a single useless. However, *K* and *C* typically have values greater than 1, so multiplying those two positions instead of one leads to $K \times C$ useless products. This amount is certainly considerable and hurts performance and efficiency.

In contrast, the advantage of CDS dataflow based on CPC is that the *K* and *C* loops have moved to the lowest level (Fig. 7(a)). As a result, only one ifmap plane position and one filter plane position are processed in each tile. Consequently, either all multiplications will be useless in a cycle or all will lead to useful products. Therefore, we can simply avoid the cycles that result in useless products. To do this, we could put the conditions of equation 4 into pseudo-code 7(c) or 7(d) after the loops for dimensions *R* and *S*. So that a comparison between the position (*r*, *s*) of the filter and the position of the current ifmap CDA is performed first, and if one of the conditions is true, the sending and processing of all filter values with position (*r*,*s*) are

[1] The stride of convolution.

[DOI: 10.61186/itrc.16.4.9]

prevented. In this case, though, the issue is that the equation 4 comparisons will be executed a great number of times. Notably, if $C > MS_{number}$ (Fig. 7(d)), the frequency of these comparisons will also be increased. This issue, as the overhead, can negatively impact the runtime and the energy consumption.

Therefore, we have converted the comparison operation into a constrained frame named FUCA (Fig. 7(e) and Fig. 7(f)). The FUCA, at two higher levels of pseudo-code after the Y loop, establishes the frame for filter positions. With the assurance that none of its positions in computation with the current ifmap CDA would generate useless products. Thus, at the start of a process, when the current ifmap CDA is mapped to MSes, the FUCA sets the frame at the same time. Then, only the filter CDAs in the frame are sent to MSes for computations in subsequent cycles of the process. This strategy results in the number of comparisons being equivalent to the number of ifmap CDAs, which is significantly lower than in the preceding case. Figures 7(e) and 7(f) depict the final pseudo-codes of the designed dataflow that improved with FUCA.

VI. EVALUATION

First, this section discusses the functionality of the selected tool to evaluate the FUCA. Then, it details the chosen layers and dataflows for the experiment. Finally, the merits and weaknesses of the suggested idea are analyzed by illustrating the results.

A. Analyzer tool

We assessed the designed and enhanced dataflows using the mRNA tool (mapper for Reconfigurable Neural Accelerators) [25]. Based on the MAERI architecture, mRNA is an open-source tool for analyzing dataflow and mapping strategies at various layers and finding the best mappings.

The mRNA first receives the parameters of the input layer (dimension sizes, stride, etc.) and accelerator resources (number of MSes, bandwidth, etc.) and then lists mapping strategies with the most potential for MS utilization. Next, it assesses each strategy listed and presents the results of all of them, including metrics like energy consumption, runtime, utilization, etc.

The convolutional layer's dataflow in mRNA is the same as the MAERI dataflow and is based on SWC. This dataflow is described in Section II-B. In reality, mRNA defines and evaluates different tilings on input layer dimensions as mapping strategies and then finds the optimal one from the set. Thus, the dataflow is constant across all mapping strategies, and only the values of the tiling parameters vary. For this reason and also because the purpose and focus of this work are to evaluate the dataflows, not different strategies, we have considered the average results of different strategies in a dataflow as the results of that dataflow.

To support the designed and enhanced dataflows, we have upgraded the mRNA. We have included a parameter in the mRNA input file that, when set to "*true*", lets mRNA run the extended part and evaluate the input layer mapping according to the designed dataflow.

B. Methodology

To illustrate the impact of FUCA, four distinct dataflows, each with the names and characteristics listed below, have been taken into consideration for analysis and comparison.

MAERI-MDF (MAERI's Main DataFlow) [11, 25]. The basic MAERI's dataflow for convolutional layers, which is based on SWC (described in Section II-B). The original dataflow in mRNA is likewise this one.

CPDF (Cartesian Product based DataFlow). The presented CDS dataflow, which is based on CPC (described in Section IV and displayed as pseudo-code in Figures 7(a) and 7(b)). No improvements have been made to this dataflow.

CPDF-AIZ (Cartesian Product based DataFlow with the Ability to Ignore Zero-padding Zeros). The same dataflow as CDPS, where only the transmission and computation of zeros resulting from zero-padding are prevented (displayed as pseudo-code in Figures 7(c) and 7(d)).

CPDF-FUCA (Improved Cartesian Product-based DataFlow with FUCA). Improved final dataflow by FUCA, which not only prevents unnecessary computations caused by zero-padding but also avoids processing data with useless results (displayed as pseudo-code in Figures 7(e) and 7(f)).

To better compare the mentioned dataflows and evaluate the influence of zero-padding, we defined a set of identical custom convolution layers with different numbers of zero-paddings and, consequently, different output dimensions. Table 1 presents the parameters for these layers. We also chose multiple convolutional layers from various advanced DNNs with varying strides and zero-paddings to assess the effectiveness of the proposed approach across different DNNs. These layers are displayed in Table 2.

C. Results

The findings from evaluating the impact of FUCA on CPC-based dataflow. The runtime and energy consumption of convolution layers in Table 1 have been evaluated by the mRNA in four dataflows (MAERI-MDF, CPDF, CPDF-AIZ, and CPDF-FUCA), and the results are shown in Fig. 8 as two charts. Figures 8(a) and 8(b) show the results of normalized runtime and normalized energy consumption, respectively. As can be observed, both in terms of runtime and energy consumption, the final presented dataflow (CPDF-FUCA) performs better than other dataflows, even topping MAERI's original dataflow (MAERI-MDF). The results of CPDF-FUCA and MAERI-MDF are nearly identical at $ZP = 0$, but CPDF-FUCA has performed significantly better in layers where $ZP > 0$.

[DOI: 10.61186/itrc.16.4.9]

Figure 7. the pseudo-code of designed dataflow ((a),(c),(e) for $C < MS_{number}$ cases and (b),(d),(f) for $C > MS_{number}$ cases). **(a),(b)** Cartesian Product based DataFlow with no improvement (CPDF). (c),(d) **C**artesian Product based DataFlow with the Ability to Ignore Zero-padding Zeros (CPDF-AIZ). **(e),(f)** Improved Cartesian Product-based DataFlow with FUCA (CPDF-FUCA)

TABLE I. THE CUSTOM CONVOLUTION LAYERS DEFINED FOR EVALUATING THE PROPOSED FRAME

LayerName	Input				Output		Filter				
	X	Y	C	N	X'	Y'	R	S	K	d	ZP
Custom Layer-0	27	27	64	1	11	11	7	7	128	2	0
Custom Layer-1	27	27	64	1	12	12	7	7	128	2	1
Custom Layer-2	27	27	64	1	13	13	7	7	128	2	2
Custom Layer-3	27	27	64	1	14	14	7	7	128	2	3
Custom Layer-4	27	27	64	1	15	15	7	7	128	2	4
Custom Layer-5	27	27	64	1	16	16	7	7	128	2	5

TABLE II. THE CHOSEN LAYERS FROM DIFFERENT ADVANCED DNNS TO EVALUATE THE INFLUENCE OF FUCA

DNN-Name/ LayerName	Input				Output		Filter				
	X	Y	C	N	X'	Y'	R	S	K	d	ZP
DarkNet-19 [26]/ CONV13	14	14	256	1	14	14	3	3	512	1	1
ResNet18 [16]/ CONV4a-1	28	28	128	1	14	14	3	3	256	2	1
ResNet50 [16]/ CONV4a-shortcut	28	28	512	1	14	14	1	1	**1024**	2	0
DenseNet121 [17]/ Transition3(CONV)	14	14	**1024**	1	14	14	1	1	512	1	0
YOLO [19]/ Block2-CONV1	112	112	64	1	112	112	3	3	192	1	1
AlexNet [27]/ CONV2	27	27	96	1	27	27	5	5	256	1	2

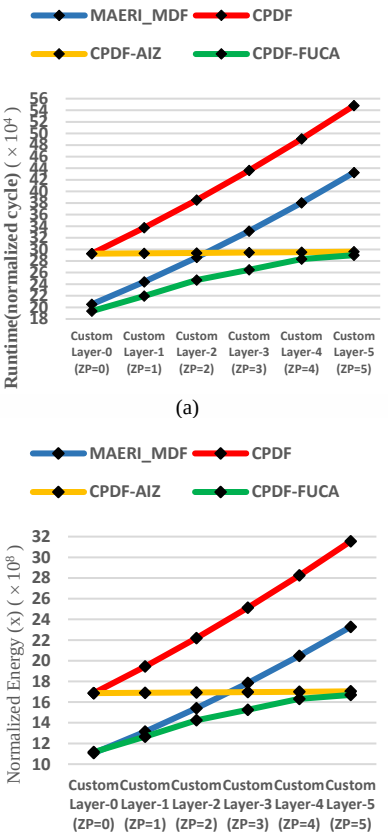


Figure 8. Evaluation results of MAERI-MDF, CPDF, CPDF-AIZ, and CPDF-FUCA dataflows on identical custom layers with different zero-paddings. (a) Normalized runtime results. (b) Normalized energy consumption results

[DOI: 10.61186/itrc.16.4.9]

The correctness of the discussion in Section III-B can be observed by the comparison of the CPDF-FUCA and CPDF-AIZ results. In *Custom Layer-0* (where *ZP* = 0), based on equation 2, the number of useless products will be high. So, CPDF-FUCA has a lower energy consumption and runtime in this layer compared to CPDF-AIZ due to the prevention of useless product generation[1]. But as the results of the next custom layers show, because as *ZP* increases, the number of useless products decreases, the results of these two dataflows are closer to each other.

According to charts, the CPDF performs worse than the other dataflows in all custom layers, but the CPDF-AIZ does perform better than even the MAERI-MDF in high *ZP*s. However, since the *ZP* of the convolution layers of most DNNs is between 0 and 2, it's safe to say that the CPDF-AIZ's good efficiency in high *ZP*s doesn't matter.

Evaluation results of the chosen DNN layers. The normalized runtime results of CPDF-FUCA and MAERI-MDF dataflows for chosen DNN layers (Table 2) are displayed in Fig. 9. As it is known, CPDF-FUCA has a lower runtime than MAERI-MDF in all layers except for *Alexnet-CONV2*, indicating that the proposed method has performed better in these layers.

The greater runtime of the CPDF-FUCA in the Alexnet-CONV2 compared to the MAERI-MDF has been caused by a lack of work on mapping strategies in the designed dataflow, which is outside the scope of this paper. CPDF-FUCA utilizes only 75% of the MSes in each cycle for the AlexNet layer, resulting in an approximate utilization rate of 75%. Enhancing the utilization rate of the designed dataflow through tiling on various dimensions in future work will lead to a reduction in runtime in layers similar to AlexNet.

Fig. 10 displays the normalized total energy consumption results of the CPDF-FUCA and MAERI-MDF dataflows for selected DNN layers. Each bar in the one-layer chart displays the energy consumption based on related dataflow, both total and separated in different parts of MAERI. The red color indicates energy consumption in the DN network, green in the MN network, yellow in the RN network, blue in the SPM (ScratchPad Memory, which is the same prefetch buffer), and black in the DRAM.

As the charts show, in the layers of ResNet50, ResNet18, and DenseNet121, the CPDF-FUCA dataflow has lower energy consumption compared to MAERI-MDF. However, in the remaining layers, MAERI-MDF exhibits better energy consumption. If we pay attention to the energy consumption of the parts of MAERI separately in all charts, we find that the very high energy consumption of CPDF-FUCA in DN and SPM for DarkNet, Yolo, and Alexnet layers has caused its total energy consumption to be higher for these layers. High energy consumption based on the presented dataflow in DN and SPM usually occurs where $R > 1$, $S > 1$, and $d < 2$. The reason for this is that in MAERI-MDF, which is a WS dataflow based on SWC, ifmap values are reused in two ways. First, in each cycle, the values of a window from the ifmap plane are multicast to VNs instead of unicast. Second, only a portion of the new window of the ifmap is multicast to MSes per cycle due to MAERI's store-and-forward multicast capability. These two factors reduce DN transmissions and the number of accesses to the prefetch buffer during sequential cycles, leading to an improvement in energy consumption in both the SPM and DN. Efforts to reuse data in the presented dataflow can result in improved energy in future works.

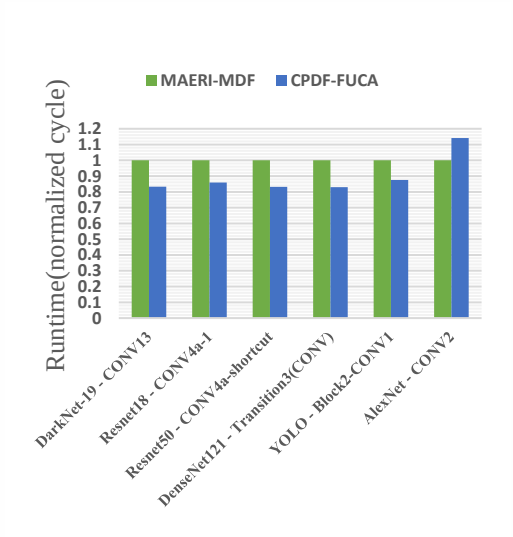


Figure 9. Normalized runtime results of CPDF-FUCA and MAERI-MDF dataflows for DNN layers of Table 2

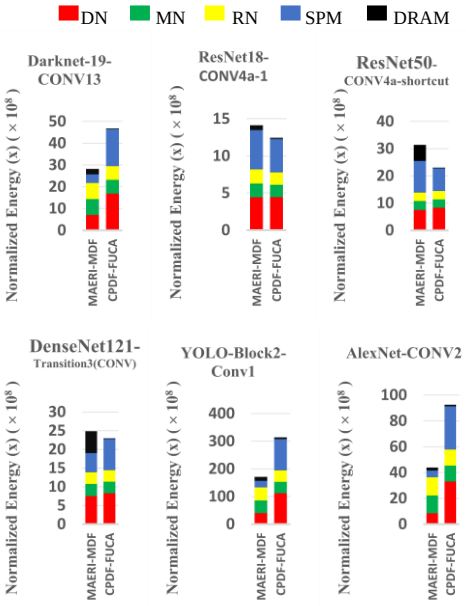


Figure 10. Normalized energy consumption results of CPDF-FUCA and MAERI-MDF dataflows for DNN layers of Table 2

[1] Note that the sole distinction between CPDF-FUCA and CPDF-AIZ is that CPDF-FUCA prevents the generation of useless products, whereas CPDF-AIZ does not.

[DOI: 10.61186/itrc.16.4.9]

VII. Conclusion

This paper presents a frame called FUCA for CPC-based dataflows with an all-to-all nature. FUCA effectively eliminates the processing of data that leads to useless results. This frame ensures that the number of computations in CPC-based dataflows not only becomes equivalent to the number of computations in SWC-based dataflows, but at some layers, it becomes even less due to the consideration of a policy for zero-padding.

To evaluate the FUCA, we designed a CDS dataflow based on CPC and implemented the FUCA on it. The FUCA algorithm, placed in the middle of the designed dataflow, prevents the transmission of filter values that result in useless products. We assessed the current work using the mRNA tool, which is a dataflow and mapping analyzer for MAERI. Of course, we upgraded this tool to support the suggested idea.

Our experiments were conducted on two sets of layers. In the first set, all layers are custom and the same, only differing in the number of zero-padding and, naturally, the output dimensions. The second set includes layers selected from various advanced DNNs.

The experiments on the first set indicated that FUCA in CPC-based dataflows leads to an average reduction of 39% in both energy consumption and runtime. Based on these experiments, we even observed that the CPC-based dataflow with FUCA outperforms the SWC-based dataflow of MAERI when $ZP > 0$. Also, when ZP is equal to zero, the performance of the developed dataflow is nearly identical to that of MAERI's dataflow. The findings from the experiments conducted on the second set also demonstrated that, in some instances, MAERI's dataflow performs better. The use of techniques such as store-and-forward multicast and data reuse in the dataflow and architecture of MAERI is the reason for this. As part of future work, one can try to incorporate such techniques into the proposed dataflow to address this issue.

References

[1] Lee, K., et al. *SecureLoop: Design Space Exploration of Secure DNN Accelerators.* in Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture. 2023.

[2] Moghaddasi, I., S. Gorgin, and J.-A. Lee, *Dependable DNN Accelerator for Safety-critical Systems: A Review on the Aging Perspective*. IEEE Access, 2023.

[3] Yang, J., H. Zheng, and A. Louri. *Venus: A versatile deep neural network accel-erator architecture design for multiple applications*. in Proceedings of Design Automation Conference (DAC). 2023.

[4] Parchamdar, B. and M. Reshadi. *Data Flow Mapping onto DNN Accelerator Considering Hardware Cost*. in 2020 IEEE 14th Dallas Circuits and Systems Conference (DCAS). 2020. IEEE.

[5] Rasch, M.J., et al., *Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators*. Nature Communications, 2023. **14**(1): p. 5282.

[6] Moon, G.E., et al., *Evaluating spatial accelerator architectures with tiled matrix-matrix multiplication.* IEEE Transactions on Parallel and Distributed Systems, 2021. **33**(4): p. 1002-1014.

[7] Chen, Y.-H., et al., *Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks*. IEEE journal of solid-state circuits, 2016. **52**(1): p. 127-138

[8] Aimar, A., et al., NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps. IEEE transactions on neural networks and learning systems, 2018. **30**(3): p. 644-656.

[9] Parashar, A., et al., *SCNN: An accelerator for compressed-sparse convolutional neural networks.* ACM SIGARCH computer architecture news, 2017. **45**(2): p. 27-40.

[10] Narimanjahan, B., et al., *MCPS: a mapping method for MAERI accelerator base on Cartesian Product based Convolution for DNN layers with sparse input feature map*. Cluster Computing, 2022. **25**(5): p. 3213-3230.

[11] Kwon, H., A. Samajdar, and T. Krishna, *Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects*. ACM SIGPLAN Notices, 2018. **53**(2): p. 461-475.

[12] Armeniakos, G., et al., *Hardware approximate techniques for deep neural network accelerators: A survey*. ACM Computing Surveys, 2022. **55**(4): p. 1-36.

[13] Taheri, M., et al., *Exploration of Activation Fault Reliability in Quantized Systolic Array-Based DNN Accelerators*. arXiv preprint arXiv:2401.09509, 2024.

[14] Zhang, S., et al. *Cambricon-X: An accelerator for sparse neural networks*. in 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). 2016. IEEE.

[15] Szegedy, C., et al. *Going deeper with convolutions*. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[16] He, K., et al. *Deep residual learning for image recognition*. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[17] Huang, G., et al. *Densely connected convolutional networks*. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[18] Huang, Z., et al., *DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection*. Information Sciences, 2020. **522**: p. 241-258.

[19] Redmon, J., et al. *You only look once: Unified, real-time object detection*. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[20] Redmon, J. and A. Farhadi, *Yolov3: An incremental improvement.* arXiv preprint arXiv:1804.02767, 2018.

[21] Wang, G., et al., *UAV-YOLOv8: a small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios*. Sensors, 2023. **23**(16): p. 7190.

[22] Hanson, E., et al. *Cascading structured pruning: enabling high data reuse for sparse DNN accelerators*. in Proceedings of the 49th Annual International Symposium on Computer Architecture. 2022.

[23] Jouppi, N.P., et al. *In-datacenter performance analysis of a tensor processing unit*. in Proceedings of the 44th annual international symposium on computer architecture. 2017.

[24] Chatarasi, P., et al., *Marvel: a data-centric approach for mapping deep learning operators on spatial accelerators* . ACM Transactions on Architecture and Code Optimization (TACO), 2021. **19**(1): p. 1-26.

[25] Zhao, Z., et al. *mrna: Enabling efficient mapping space exploration for a reconfiguration neural accelerator*. in 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). 2019. IEEE.

[26] Redmon, J. and A. Farhadi. *YOLO9000: better, faster, stronger*. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[27] Krizhevsky, A., I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems, 2012. **25**.

[DOI: 10.61186/itrc.16.4.9]

Babak NarimanJahan received his B.Sc. degree in Computer Engineering (Hardware) from Qazvin Branch, Islamic Azad University, Qazvin, Iran, in 2002. He also received his M.Sc. and Ph.D. degrees in Computer Engineering (Computer Architecture) at the Department of Computer Engineering, Science and Research Branch, Islamic Azad University (SRBIAU), Tehran, Iran, in 2005 and 2022, respectively. He is currently a faculty member at the Department of Computer Engineering, Bonab Branch, Islamic Azad University, Bonab, Iran. His research interests include Deep Neural Network Accelerators, Computation Mapping Over DNN Accelerators and On-Chip Interconnection Networks.

Ahmad Khademzadeh was born in Mashhad, Iran, in 1943. He received his B.Sc. degree in Applied Physics from Ferdowsi University, Mashhad, Iran, in 1969 and the M.Sc., Ph.D. degrees respectively in Digital Communication and Information Theory and Error Control Coding from the University of Kent, Canterbury, U.K. He is currently the Head of Education and National Scientific and Informational Scientific Cooperation Department at Iran Telecom Research Center (ITRC). He was the head of Test Engineering Group and the director of Computer and Communication Department at ITRC. He is also a lecturer at Tehran Universities and he is a committee member of Iranian Computer society and also a committee member of the Iranian Electrical Engineering Conference Permanent Committee. Dr. Khademzadeh has been received four distinguished national and international awards including Kharazmi International Award, and has been selected as the National outstanding researcher of the Iran Ministry of Information and Communication Technology.

Akram Reza received her B.Sc. degree in Computer Hardware Engineering from Qazvin Branch, Islamic Azad University, Qazvin, Iran, in 2004. He also received his M.Sc and Ph.D. degrees in Computer Architecture at the Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2008 and 2014, respectively. She currently faculty member at the Department of computer Engineering, Qods Branch, Islamic Azad University, Tehran, Iran. Her research focuses on Different Aspects on Network-On-Chip Architectures, Parallel Systems Architecture and Machine Learning Algorithms.

[DOI: 10.61186/itrc.16.4.9]