electronics

MDPI

Article

Power-Efficient Deep Neural Network Accelerator Minimizing Global Buffer Access without Data Transfer between Neighboring Multiplier—Accumulator Units

Jeonghyeok Lee, Sangwook Han, Seungwon Choi * and Jungwook Choi *

Department of Electronic Engineering, Hanyang University, Seoul 04763, Korea; jeonghyeok.lee@dsplab.hanyang.ac.kr (J.L.); ssssang3234@dsplab.hanyang.ac.kr (S.H.)
* Correspondence: choi@dsplab.hanyang.ac.kr (S.C.); choij@hanyang.ac.kr (J.C.)

Abstract: This paper presents a novel method for minimizing the power consumption of weight data movements required by a convolutional operation performed on a two-dimensional multiplier–accumulator (MAC) array of a deep neural-network accelerator. The proposed technique employs a local register file (LRF) at each MAC unit in a manner such that once weight pixels are read from the global buffer into the LRF, they are reused from the LRF as many times as desired instead of being repeatedly fetched from the global buffer in each convolutional operation. One of the most evident merits of the proposed method is that the procedure is completely free from the burden of data transfer between neighboring MAC units. It was found from our simulations that the proposed method provides a power saving of approximately 83.33% and 97.62% compared with the power savings recorded by the conventional methods, respectively, when the dimensions of the input data matrix and weight matrix are 128 × 128 and 5 × 5, respectively. The power savings increase as the dimensions of the input data matrix or weight matrix increase.

Keywords: deep learning accelerator; field-programmable gate array (FPGA); deep neural networks (DNNs)

Citation: Lee, J.; Han, S.; Choi, S.; Choi, J. Power-Efficient Deep Neural Network Accelerator Minimizing Global Buffer Access without Data Transfer between Neighboring Multiplier—Accumulator Units. *Electronics* **2022**, *11*, 1996. https://doi.org/10.3390/electronics11131996

Academic Editors: Daniel Casini, Alessandro Biondi and Akash Kumar

Received: 5 April 2022
Accepted: 23 June 2022
Published: 25 June 2022

1. Introduction

Recently, several studies on deep neural networks (DNNs) have been conducted to apply them to various fields, including image classification [1–3], object detection [4,5], speech recognition [6,7], and autonomous vehicles [8]. These DNNs are mostly used in edge devices, such as Internet of Things devices or cell phones, wherein power consumption is a critical issue. This study addresses the problem of minimizing the power consumption of DNN accelerators.

Recalling that the main functionality of a DNN is the convolutional operation between the input data and weight pixels, we propose a novel procedure for minimizing the power consumption required by the data movements involved in the convolutional operation. As DNN accelerators generally consume far more power in data movements than that consumed in arithmetic operations [9,10], a power-efficient DNN accelerator cannot be accomplished without reducing the power consumption of data movements. In particular, this study addresses the issue of minimizing the power consumption of data movements by reducing the number of global buffer accesses required to obtain the weight pixel to be convolved with a given input data pixel. The proposed technique introduces a novel control procedure for managing a local register file (LRF) to be installed at each multiplier–accumulator (MAC) unit of a two-dimensional (2-D) MAC array in a given DNN. This management is conducted in a manner such that using the proposed LRF, each weight pixel fetched from the global buffer can be reused as many times as required in the convolutional operation. Consequently, the global buffer access is replaced with the LRF

access, where the power consumption of the latter is approximately six times that of the former [11].

Several dataflow methods such as weight stationary (WS) [12–17], output stationary (OS) [18–20], and row stationary (RS) [11] have been proposed to reduce the global buffer access for the purpose of power consumption during the convolutional operation. Among them, ref. [11] has demonstrated that when both input data and weight pixels are properly reused, the power consumption can be far lower than that produced by other [12–20] methods. However, as the dimension of the input data matrix or that of the weight matrix increases, the method in [11] suffers owing to the fact that the required number of data transfers between neighboring MAC units and the size of each LRF should simultaneously be proportionally increased. This is because the LRF at each MAC unit, proposed by [11], has to provide both input data and weight pixels, which should be transferred to the neighboring MAC unit.

Meanwhile, in [21], by properly rearranging the convolutional operation procedure, the MAC units in the 2-D MAC array can share input data according to the columns, and the results indicate that the LRF for input data pixels can be installed outside the MAC unit. Consequently, the amount of hardware (HW) resources required by the LRF of the input data pixels can be far less than that of other dataflow methods. However, the method in [21] focuses only on the reuse of input data and does not discuss the reuse of weight pixels. In this paper, we propose a DNN accelerator that requires minimal HW resources compared with that required in all previous studies [11] and is more power efficient than the method in a previous study [21]. This superior efficiency results from the fact that the proposed methods allow weight pixels as well as input data to be reused. Furthermore, the proposed weight data dataflow method provides considerable power saving in comparison with that provided by the method in [11]. This is because it does not include the weight data transfer between the neighboring MAC units.

The main contributions of this paper can be summarized as follows.

- Proposal of a novel control sequence for reusing weight pixels by utilizing an LRF installed in each MAC unit in a 2-D MAC array.
- Analysis of global buffer access reduction ratio and power saving in comparison with that in the RS dataflow method as well as the methods involving no data owing to the reuse of weight pixels through the proposed control sequence.

The remainder of this paper is organized as follows. In Section 2, we introduce a novel method for minimizing power consumption in a DNN accelerator employing a 2-D MAC array by replacing global buffer access with LRF access. Section 3 presents a numerical analysis to demonstrate the superiority of the proposed accelerator in comparison with typical conventional methods [21] and RS dataflow [11]. Finally, Section 4 concludes the paper.

2. Proposed Method for Minimizing Global Buffer Access by Using a Local Register File

In this section, we introduce a novel procedure for implementing control sequences that control the reuse of weight pixels using an LRF to be installed at each MAC unit of a DNN accelerator, the main functionality of which is a convolutional operation. The key issue is to minimize the power consumption required for data movements of the weight pixels during the convolutional operation by minimizing the number of global buffer accesses. To accomplish this, weight pixels are stored at the proposed LRF to be reused as many times as needed instead of being repeatedly fetched from the global buffer. Consequently, the dataflow of the weight pixels should be modified in accordance with the new structure of each MAC unit that includes the proposed LRF. More specifically, we propose a new DNN accelerator with novel control sequences that properly controls the reuse of weight pixels using the LRF at each MAC unit of the 2-D MAC array [21,22].

Figure 1 illustrates the structure of a 16×128 2-D MAC array at each MAC unit, where a $W_{n(x)} \times W_{n(y)}$ LRF is installed. Here, $W_{n(x)}$ and $W_{n(x)}$ denote the row and column dimen-

sions of the weight matrix, respectively. The LRF is to be implemented as a $W_{n(x)} \times W_{n(y)}$ register at each MAC unit of the 2-D MAC array to store the weight pixels to be convolved with the corresponding input data. On the other side, the LRF for the input data does not have to be installed at each MAC unit. Because the input data pixels can be properly rearranged such that all the LRF contents corresponding to the input data pixel at each column of a given 2-D MAC array are identical to one another [21]. Consequently, we need 128 LRFs to store all the input data pixels involved in a given convolutional operation, while each of the 128 LRFs is shared by all the MAC units of the corresponding column. Compared to the method of [21], the LRF in the proposed method is to be installed at each MAC unit whereas the method of [21] requires the whole LRFs to be installed outside the MAC array, which means the proposed method is more advantageous than [21] in terms of the power consumption required for data movements between the LRF and MAC unit.

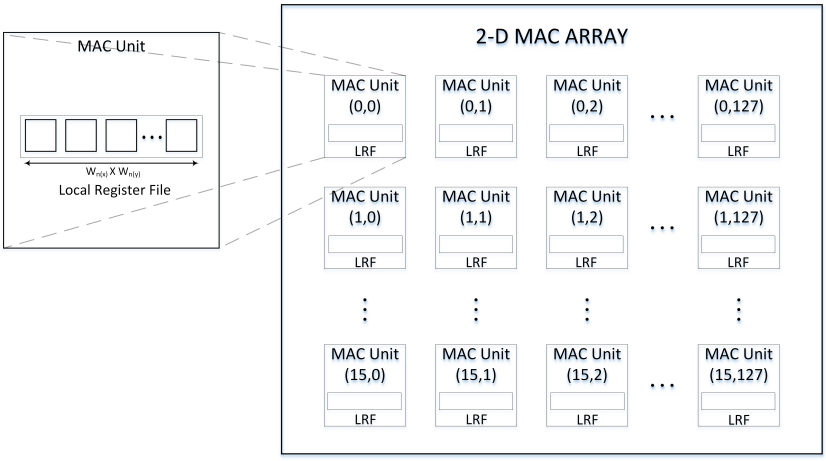

Figure 1. Two-dimensional multiplier–accumulator (2-D MAC) array structure with the proposed local register file (LRF) at each MAC unit.

Figure 2 illustrates a conceptual block diagram depicting the manner in which each pixel of 16 weights of 128 channels is read from the global buffer into the LRF at each corresponding MAC unit of the 16 × 128 2-D MAC array. As shown in Figure 2, the weight pixels from the *i*th weight matrix of *j*th channel are to be stored at the LRF installed at the MAC unit of the *i*th row of the *j*th column of the 2-D MAC array.

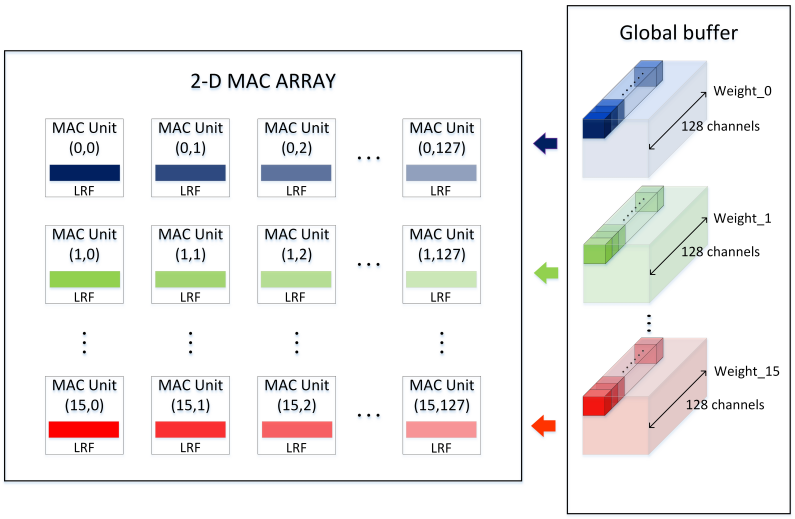

Figure 2. Conceptual block diagram depicting the manner in which each pixel of 16 weights of 128 channels is read from the global buffer into the LRF at each corresponding MAC unit of the 16 × 128 2-D MAC array.

The objective here is to reduce the power consumption required for data movements during convolutional operations in a given DNN accelerator by exploiting data reuse for weight pixels. The reduction in power consumption presented in this paper will further enhance the existing power saving that was accomplished through the reuse of input data pixels, as discussed in [21]. The proposed control sequences explained in this section are implemented in such a way that each weight pixel is reused as many times as the input data reuse factor by storing $W_{n(x)} \times W_{n(y)}$ weight pixels at each corresponding MAC unit, as shown in Figures 1 and 2. Such an implementation is followed to reuse the weight pixels as many times as required for the convolutional operation instead of repeatedly fetching them from the global buffer whenever needed. Note that the proposed control sequences can arbitrarily set the data reuse factor for each weight pixel to the data reuse factor of the input data pixels given a priori. This indicates that the data reuse factor for each weight pixel can be flexibly controlled according to the given data reuse factor of the input data pixels.

Figure 3 illustrates the MAC unit structure prepared for the convolutional operation between the input data and weights. Figure 3a corresponds to the situation wherein a convolution operation is performed without the merit of data reuse, meaning that the weight pixel is read from the global buffer whenever needed. Consequently, the required number of global buffer accesses is the same as the required number of convolutional operations, which results in a tremendous amount of power consumption. Figure 3b depicts the MAC unit structure employing the concept of RS dataflow [11] to reuse the weight pixel. Note that each MAC unit includes LRFs for storing weight pixels as well as input data pixels and partial sums. In the RS dataflow method, all the weight pixel elements in each row of the given weight matrix are stored in the LRF set up at each MAC unit. Each MAC unit should also provide additional LRF resources to store the corresponding input data pixels to be convolved with the row of the weight matrix. Consequently, the contents of the LRFs at each MAC unit, that is, weight, input, and partial sum, should be transferred successively to the next corresponding MAC unit. The LRF resources installed for the weight pixels, input data pixels, and partial sums are denoted as green, blue, and red, respectively, in Figure 3b. Figure 3c illustrates the proposed MAC unit structure, including the LRF installed for storing the weight pixels. The control signal determines the dataflow around the $W_{n(x)} \times W_{n(y)}$ LRF and convolutional operator. During the initial stage, the weight pixels read from the global buffer are selected to be stored at the LRF as well as to be convolved with the corresponding input data. As the LRF is filled with the weight pixels read from the global buffer, the initial stage is complete. The control signal then selects the correct weight pixel stored at the LRF, which is the weight pixel to be convolved with the corresponding input data. This operation, that is, the convolution of each LRF element with the corresponding input data pixel, is continued as many times as the weight data reuse factor, which can be set arbitrarily according to the value of the input data reuse factor set a priori. As a matter of fact, the weight data reuse factor, R_w, is determined as $R_i - 1$, where R_i denotes the input data reuse factor. This indicates that each of the weight pixels stored at the LRF is reused for the convolutional operation as many times as the weight data reuse factor. Consequently, global buffer access is not required, while the weight pixels stored at the LRF at each MAC unit are repeatedly used for convolution with the corresponding input data. Mathematical DNN convolution expressions at the proposed MAC unit structure can be summarized as

$$O = \sum_{k=0}^{C-1} \sum_{j=0}^{R_i-1} \sum_{i=0}^{K-1} I[k][j] \times W[i]. \quad (1)$$

where O, I, W, C, and K denote output data, input data, weight, Input data size, and weight size, respectively.

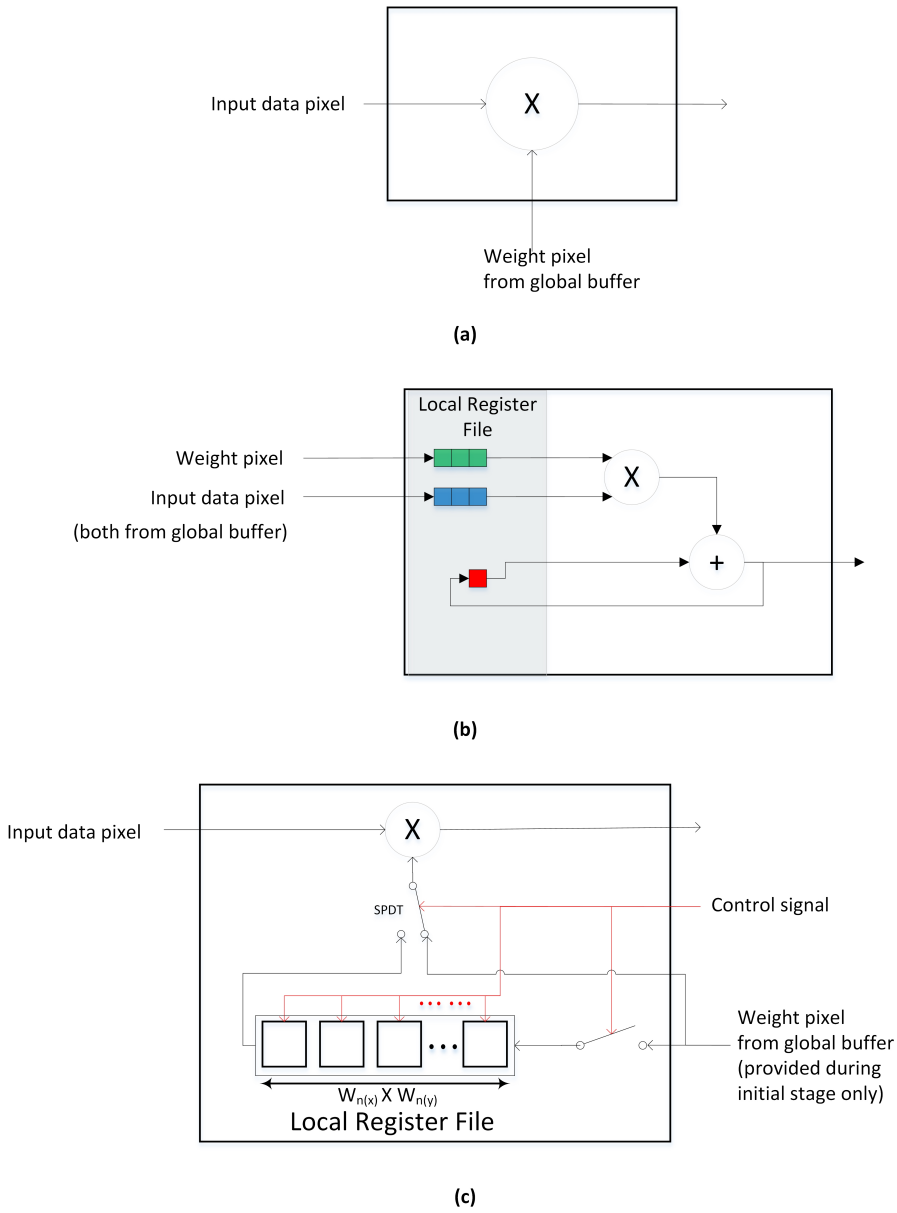

Figure 3. (**a**) MAC unit structure allowing no data reuse. (**b**) MAC unit structure allowing data reuse with row stationary dataflow [2]. (**c**) MAC unit structure allowing data reuse with proposed method.

Figure 4 illustrates a flowchart of the proposed control sequences that control the reuse of the weight pixels. More specifically, the flowchart shown in Figure 4 explains the manner in which the proposed MAC unit structure operates in terms of data reuse for the weight pixels. At the start of the convolutional operation, the global buffer address for the target weight matrix is updated, and the weight data reuse factor is reset, that is, $R_w \leftarrow 0$. Note that the input data reuse factor, R_i, is provided a priori such that the weight pixels are reused $R_i - 1$ times. It particularly means that the proposed method can arbitrarily adjust the reuse factor of each weight in accordance with the input data reuse factor of the method of [21], which means the proposed method can very efficiently be combined with the method of [21]. During the initial stage, the weight pixel successively read from the global buffer is not only used for convolution with the corresponding input data but also stored at the proposed LRF. When the initial stage ends, that is, when the LRF is completely stored with $W_{n(x)} \times W_{n(y)}$ weight pixels, the convolutional operation is continued repeatedly using the stored weight pixels. As each of all the $W_{n(x)} \times W_{n(y)}$ weight pixels stored in the LRF is reused for the convolution operation, the present value

for the weight data reuse factor is incremented, that is, $R_w \leftarrow R_w + 1$. After reusing all the stored $W_{n(x)} \times W_{n(y)}$ weight pixels $R_i - 1$ times, we are required to check whether all the weight matrices have been considered. If so, the entire procedure of the convolutional operation for a given MAC unit is completed. Otherwise, the procedure returns to the step of updating the global buffer address for the next target weight matrix.

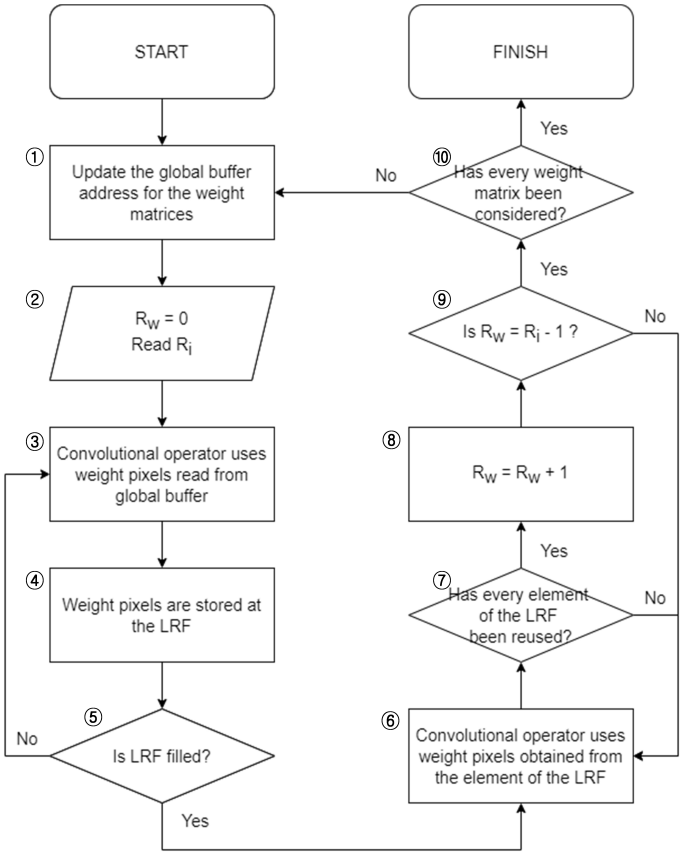

Figure 4. Proposed control sequences determining the dataflow of weight pixels.

3. Numerical Results

3.1. Power Saving

This subsection presents the numerical results regarding the gain of the proposed method obtained from various computer simulations.

First, we summarize the extent to which the proposed method reduces the required number of global buffer accesses using the concept of data reuse discussed in the previous section. With no data reuse, as shown in Figure 3a, the required number of global buffer accesses at each MAC unit for a single-channel weight matrix is $R_i \times W_{n(x)} \times W_{n(y)} \times M \times (I_{n(y)} - (W_{n(y)} - 1))$, where $I_{n(x)}$ and $I_{n(y)}$ denote the row and column dimensions of the input data matrix, respectively, and $M = (I_{n(x)} - (W_{n(x)} - 1))/R_i$.

By applying the proposed MAC unit structure shown in Figure 3c, we can significantly reduce the required number of global buffer accesses by reusing the weight pixel as many times as the weight data-reuse factor, $R_w = R_i - 1$. More specifically, the required number of global buffer accesses becomes $W_{n(x)} \times W_{n(y)} \times M$. Furthermore, when M is greater than 1, the required number of global buffer accesses can be reduced to $W_{n(x)} \times W_{n(y)}$ based on the fact that an identical convolutional operation process M times [1]. Consequently, the global buffer access reduction ratio can be summarized as shown in (2). It indicates how much the global buffer access can be reduced by using the proposed method.

$$100 \times (1 - \frac{W_{n(x)} \times W_{n(y)}}{R_i \times W_{n(x)} \times W_{n(y)} \times M \times (I_{n(y)} - (W_{n(y)} - 1))})\%. \quad (2)$$

Figure 5 illustrates the global buffer access reduction ratio as a function of the input data matrix dimension according to three different values of the weight matrix dimensions. It can be observed that the global buffer access reduction ratio monotonically increases as the dimension of the input data matrix increases. However, the global buffer access reduction ratio decreases as the dimension of the weight matrix increases. Note that the required number of global buffer accesses for the weight data in the method of [21] is the same as the required number of convolutional operations, which is reduced as the weight matrix size increases for a given input data matrix dimension. It can also be observed that the global buffer access reduction ratio increases a little whenever the integer part of M is incremented, which in our case occurs when the dimensions of the input data matrix become 34, 36, and 38 for the weight matrix dimensions to be 3×3, 5×5, and 7×7, respectively. Note that the integer part of M is changed from 1 to 2 in these situations.

It is noteworthy that the global buffer access reduction ratio for weight pixels is not reduced even when the output feature width, that is, the dimension of the convolutional operation results, is not set to an integer multiple of R_i. The global buffer access reduction ratio for input data pixels is reduced considerably when the output feature width is not set to an integer multiple of R_i [21].

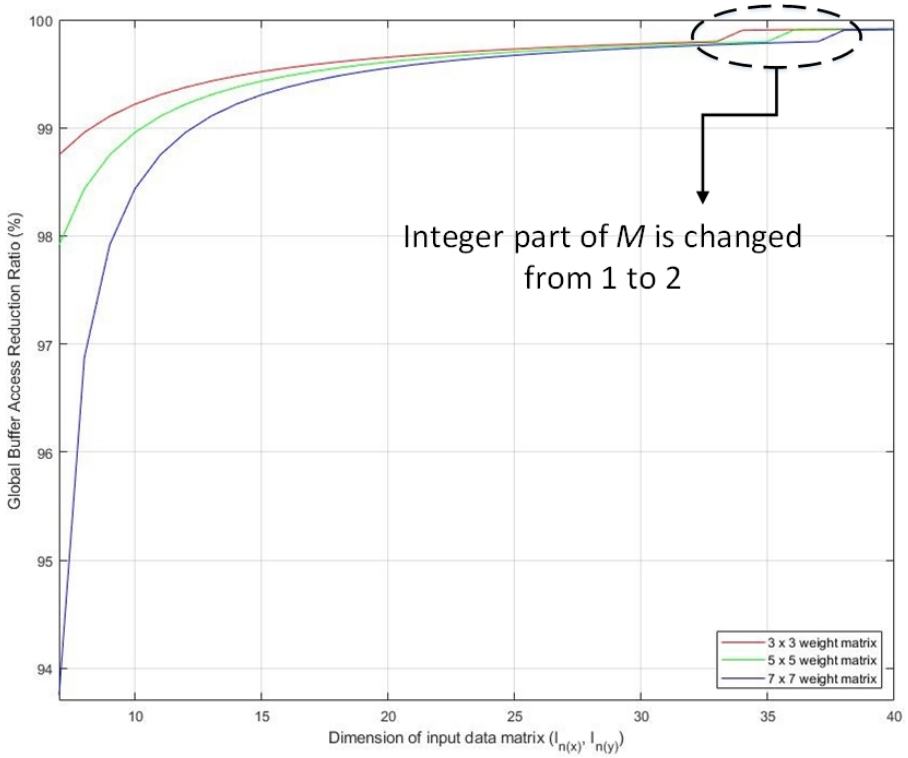

Figure 5. Global buffer access reduction ratio for the weight data movements yielded by the proposed method in comparison with [21].

Table 1 summarizes the global buffer access reduction ratio and the power savings for the weight data movements yielded by the proposed method. To attain the level of power savings shown in Table 1, it was assumed that the power consumption during the data movement required for the global buffer access from each MAC unit was approximately six times larger than that for the LRF access within each MAC unit [11]. It can be observed that the proposed method results in the power saving of, for instance, approximately 83.16%, 83.15%, and 83.13% when the weight matrix dimension is 3×3, 5×5, and 7×7, respectively, for the input matrix with a dimension of 32×32.

Table 1. Global buffer access reduction ratio and resultant power saving for the weight data movements provided by the proposed method in comparison with [21].

Dimension of Input Data Matrix	3 × 3 Weight		5 × 5 Weight		7 × 7 Weight	
	Global Buffer Access Reduction Ratio	Power Saving	Global Buffer Access Reduction Ratio	Power Saving	Global Buffer Access Reduction Ratio	Power saving
32 × 32	99.79%	83.16%	99.78%	83.15%	99.76%	83.13%
64 × 64	99.97%	83.31%	99.96%	83.30%	99.96%	83.30%
128 × 128	99.99%	83.33%	99.99%	83.33%	99.99%	83.33%

Now, let us compare the proposed method with the RS dataflow method [11]. It can be observed that the required number of global buffer accesses for both methods is identical, that is $W_{n(x)} \times W_{n(y)}$, for a single channel of the same weight matrix dimension. It is noteworthy, however, that each row of data of the weight matrix should be successively transferred from the LRF of the first MAC unit to that of the last MAC unit to share the given row data among all the corresponding MAC units in a given 2-D MAC array. Consequently, the RS dataflow method suffers from additional power consumption owing to the data movements required for inter-MAC unit transmissions. Because the required number of inter-MAC unit transmissions in the RS dataflow method is $W_{n(x)} \times W_{n(y)} \times (I_{n(y)} - W_{n(y)})$, the data movement gain in the proposed method in comparison to that in the RS dataflow method can be written as

$$100 \times (1 - \frac{(W_{n(x)} \times W_{n(y)})_{global\ buffer}}{(W_{n(x)} \times W_{n(y)})_{global\ buffer} + (W_{n(x)} \times W_{n(y)} \times (I_{n(y)} - W_{n(y)}))_{inter-mac\ unit}}). \quad (3)$$

Assuming that the power consumption for the abovementioned inter-MAC unit transmission is approximately twice that of the LRF access [11], we can summarize the power savings yielded by the proposed method in comparison to the power savings produced by the RS dataflow method as

$$100 \times (1 - \frac{3}{I_{n(y)} - W_{n(y)} + 3}). \quad (4)$$

Figure 6 illustrates the power saving produced by the proposed method in comparison to that produced by the RS dataflow method as a function of the input data matrix dimension according to three different values for the weight matrix dimensions. Based on this figure, we can compare the power consumption required for the data movements of weight pixels for both methods. It can be observed that the power saving monotonically increases as the dimension of the input data matrix increases, simply because the number of inter-MAC unit transmissions required in the RS dataflow method increases as the input matrix size increases. As depicted earlier in Figure 5, because the required number of convolutional operations decreases as the dimension of the weight matrix increases, the level of power saving decreases as the weight matrix size increases.

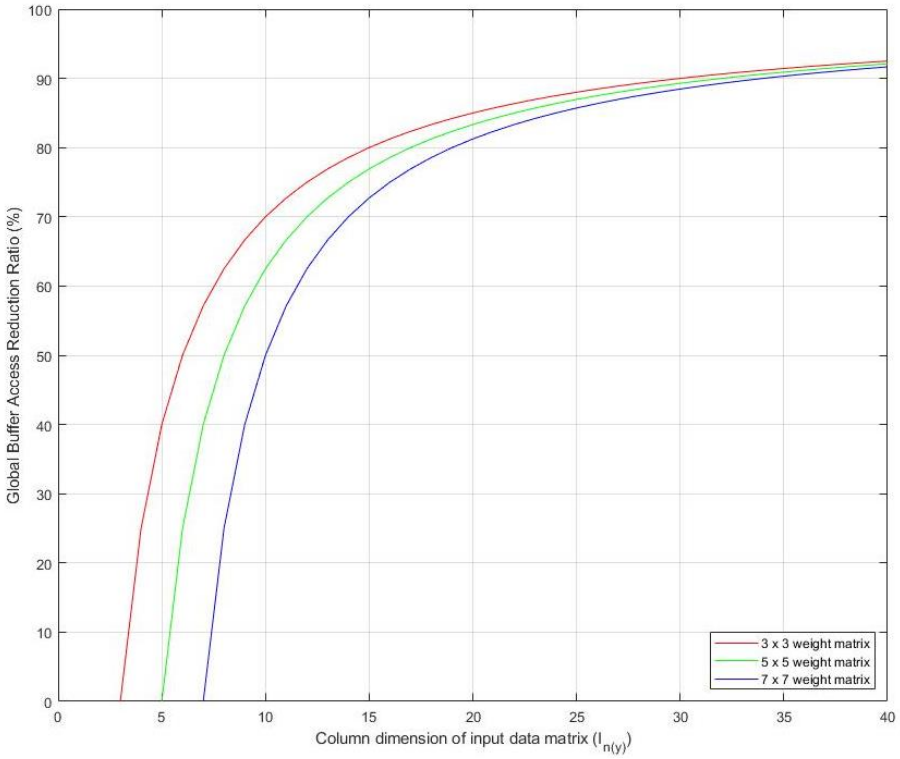

Figure 6. Power saving in data movements for weight pixels provided by the proposed method in comparison with [11].

Table 2 summarizes the number of inter-MAC unit transmissions required in the RS dataflow method, while Table 3 presents the resultant power savings produced by the proposed method.

Table 2. Required number of inter-MAC Unit transmissions in the method of [11].

Dimension of Input Data Matrix	Required Number of Inter-MAC Unit Transmissions		
	3 × 3 Weight	5 × 5 Weight	7 × 7 Weight
32 × 32	261	675	1225
64 × 64	549	1475	2793
128 × 128	1125	3075	5929

Table 3. Power saving provided by the proposed method in comparison with [11].

Dimension of Input Data Matrix	Power Saving		
	3 × 3 Weight	5 × 5 Weight	7 × 7 Weight
32 × 32	90.63%	90%	89.29%
64 × 64	95.31%	95.16%	95%
128 × 128	97.66%	97.62%	97.58%

Table 4 lists the power saving of the proposed method evaluated on a convolutional layer of AlexNet [1]. In comparison with the [11], the power saving in terms of the weight pixel movements provided by the proposed method was found to be 98.63%, 88.89%, 76.92%, 76.92%, and 76.92% in the convolutional layer of AlexNet, respectively.

Table 4. Power saving in the case of AlexNet in comparison with [11].

Convolutional Layer	Power Saving Compared to Method of [11]
1	98.63%
2	88.89%
3	76.92%
4	76.92%
5	72.92%

3.2. Hardware Resources

As depicted earlier in Figures 1–3 in Section 2, the proposed MAC unit structure includes an LRF for storing the weight pixels only, whereas the LRF for the reuse of input data is installed separately outside the 2-D MAC array. Once again, this is possible only when the convolutional operation is rearranged [21] in a way such that the input data for all MAC units of each column of the 2-D MAC array become identical.

Now, we summarize the number of HW resources that can be saved by the proposed method for installing the LRF required for input data reuse. Table 5 shows the required LRF size for the reuse of input data pixels according to various input data matrices. Including both intra—and inter-blocks, the DNN accelerator employing the proposed MAC unit structure shown in Figure 3c requires the LRF of $(R_i + (W_{n(x)} - 1) \times (R_i + W_{n(x)} - 1)) \times 128$ Kbytes for input data reuse regardless of the input data matrix size [21]. When the input data reuse factor and weight matrix dimension are given as $R_i = 16$ and $W_{n(x)} = 3$, respectively, the LRF size becomes 6.5 Kbytes as listed in Table 5. Meanwhile, for the RS dataflow method [11], in which the MAC unit structure is shown in Figure 3b, the required LRF size for the reuse of input data pixels is $I_{n(x)} \times 16 \times 128$ Kbytes for the 16×128 2-D MAC array. Each row of the input data matrix should be stored at each MAC unit, where $I_{n(x)}$ denotes the row dimension of the input data matrix. From Table 5, it can be observed that the proposed MAC unit structure saves approximately 81.94% of the HW resources required by all the input data LRFs when $I_{n(x)}$ is 18. The savings increase as the input data matrix size increases, as indicated in Table 5.

Table 5. Required LRF size for input data pixels in a 2-D MAC array of the method of [1] in comparison with [11].

Dimension of Input Data Matrix	Size of Entire LRF Required for Storing Input Data Pixels (bytes)	
	Method of [21]	Method of [11]
18×18	6.5 K	36 K
32×32	6.5 K	64 K
64×64	6.5 K	128 K
128×128	6.5 K	256 K

Table 6 shows the FPGA hardware resources required to implement the weight LRF as proposed by our method, which has been measured by using Vivado tools [23] on the FPGA of ZCU102 board [24]. It has been found that the proposed method requires 8121 units of flip-flop (FF), 5768 units of a look-up table (LUT), 0 units of block random access memory (BRAM), and 13 units of a digital signal processor (DSP) slice to install the weight LRFs on the FPGA of ZCU102 board for the 16×128 2-D MAC array shown in Figure 1. According to the method of [21], the implementation of the input LRFs requires 81,375 units of FF, 72,759 units of LUT, 100 units of BRAM, and 105 units of a DSP slice. Consequently, the extra amount of hardware resources for implementing the proposed weight LRFs is approximately 9.98% of the FF, 7.93% of the LUT, and 7.62% of the DSP slice in comparison with [21].

Table 6. Additional FPGA hardware resources required for implementing the proposed weight LRFs.

	Hardware Resources Required for Weight LRFs	Extra Hardware Value Compared to [21]
FF	8121	9.98%
LUT	5768	7.93%
BRAM	0	0
DSP slice	8	7.62%

4. Conclusions

This paper presents a novel procedure for maximizing the data reuse for weight pixels through the installation of the proposed LRF at each MAC unit of a 2-D MAC array in a DNN accelerator, whose main functionality is carrying out the convolutional operation between the input data and weight pixels. The proposed method was applied to a case wherein the convolutional operation is rearranged such that the LRF for reusing the input data can be shared by all MAC units at each column of the 2-D MAC array [21]. Although the method introduced in [21] produces tremendous savings in terms of input data movements, the proposed method further reduces the power consumption required for the data movements of weight pixels. For instance, compared with the case of no data reuse, the global buffer access reduction ratio in terms of the weight pixel movements provided by the proposed method was found to be 99.78%, 99.96%, and 99.99% for input data matrices with dimensions of 32×32, 64×64, and 128×128, respectively, when the weight matrix dimension was 5×5. Consequently, the proposed method reduced the power consumption by approximately 83.15%, 83.3%, and 83.33% in the respective cases. The proposed method was compared with the well-known RS dataflow method [11] to discover that the power consumption required for the weight pixel movements can be reduced by 90%, 95.16%, and 97.62% for input data matrices with dimensions of 32×32, 64×64, and 128×128, respectively. The main advantage of the proposed method over the RS Dataflow method is that it does not include any inter-MAC unit transfers of weight pixels, which is unavoidable in the RS dataflow method.

Author Contributions: Conceptualization, J.L.; Data curation, J.L.; Funding acquisition, J.C.; Investigation, J.L.; Methodology, J.L.; Project administration, S.C. and J.C.; Resources, S.C.; Software, J.L.; Supervision, S.C. and J.C.; Validation, S.H.; Visualization, J.L.; Writing—original draft, J.L.; Writing—review & editing, S.C. and J.C. All authors have read and agreed to the published version of the manuscript.

Funding: Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT): 2020-0-01297.

Acknowledgments: This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT) (2020-0-01297, Development of Ultra-Low Power Deep Learning Processor Technology using Advanced Data Reuse for Edge Applications).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAC	multiplier-accumulator
LRF	local register file
DNNs	deep neural networks
WS	weight stationary
OS	output stationary
RS	row stationary
HW	hardware
SPDT	single-pole double-throw

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
2. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
3. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
4. Dollár, P.; Wojek, C.; Schiele, B.; Perona, P. Pedestrian Detection: An Evaluation of the State of the Art. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 743–761. [CrossRef] [PubMed]
5. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788
6. Graves, A.; Mohamed, A.-R.; Hinton, G. Speech Recognition with Deep Recurrent Neural Networks. In Proceedings of the IEEE International Conference Acoustic, Speech Signal Process, Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.
7. Deng, L.; Li, J.; Huang, J.-T.; Yao, K.; Yu, D.; Seide, F.; Seltzer, M.; Zweig, G.; He, X.; Williams, J.; et al. Recent Advances in Deep Learning for Speech Research at Microsoft. In Proceedings of the International Conference on Acoustics [Speech], and Signal Processing (ICASSP), Vancouver, BC, Canada, 26–31 May 2013; pp. 8604–8608.
8. Chen, X.; Kundu, K.; Zhang, Z.; Ma, H.; Fidler, S.; Urtasun, R. Monocular 3D Object Detection for Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2147–2156.
9. Yang, T.-J.; Chen, Y.-H.; Emer, J.; Sze, V. A Method to Estimate the Energy Consumption of Deep Neural Networks. In Proceedings of the 51st Asilomar Conference Signals, System, Computability, Pacific Grove, CA, USA, 29 October–1 November 2017; pp. 1916–1920.
10. Mao, M.; Peng, X.; Liu, R.; Li, J.; Yu, S.; Chakrabarti, C. MAX2: An ReRAM-Based Neural Network Accelerator That Maximizes Data Reuse and Area Utilization. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 398–410. [CrossRef]
11. Chen, Y.-H.; Emer, J.; Sze, V. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. *ACM SIGARCH Comput. Archit. News* **2016**, *44*, 367–379. [CrossRef]
12. Sankaradas, M.; Jakkula, V.; Cadambi, S.; Chakradhar, S.; Durdanovic, I.; Cosatto, E.; Graf, H.P. A Massively Parallel Coprocessor for Convolutional Neural Networks. In Proceedings of the 2009 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors, Boston, MA, USA, 7–9 July 2009.
13. Sriram, V.; Cox, D.; Tsoi, K.H.; Luk, W. Towards an Embedded Biologically Inspired Machine Vision Processor. In Proceedings of the 2010 International Conference on Field-Programmable Technology, Beijing, China, 8–10 December 2010.
14. Chakradhar, S.; Sankaradas, M.; Jakkula, V.; Cadambi, S. A Dynamically Configurable Coprocessor for Convolutional Neural Networks. In Proceedings of the 37th Annual International Symposium on Computer Architecture, Saint-Malo, France, 19–23 June 2010.
15. Gokhale, V.; Jin, J.; Dundar, A.; Martini, B.; Culurciello, E. A 240 G-Ops/S Mobile Coprocessor for Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 696–701.
16. Park, S.; Bong, K.; Shin, D.; Lee, J.; Choi, S.; Yoo, H.-J. A 1.93TOPS/W Scalable Deep Learning/Inference Processor with Tetra-Parallel MIMD Architecture for Big-Data Applications. In Proceedings of the 2015 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 22–26 February 2015.
17. Cavigelli, L.; Gschwend, D.; Mayer, C.; Willi, S.; Muheim, B.; Benini, L. Origami: A Convolutional Network Accelerator. In Proceedings of the 25th Edition on Great Lakes Symposium on VLSI, Pittsburgh, PA, USA, 20–22 May 2015.
18. Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; Narayanan, P. Deep Learning with Limited Numerical Precision. In Proceedings of the International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015.
19. Du, Z.; Fasthuber, R.; Chen, T.; Ienne, P.; Li, L.; Luo, T.; Feng, X.; Chen, Y.; Temam, O. ShiDianNao: Shifting Vision Processing Closer to the Sensor. *ISCA* **2016**, *43*, 92–104. [CrossRef]
20. Peemen, M.; Setio, A.A.A.; Mesman, B.; Corporaal, H. Memory-Centric Accelerator Design for Convolutional Neural Networks. In Proceedings of the 2013 IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, USA, 6–9 October 2013.
21. Lee, M.; Zhang, Z.; Choi, S.; Choi, J. Minimizing Global Buffer Access in a Deep Learning Accelerator Using a Local Register File with Rearranged Computational Sequence. *Sensors* **2022**, *22*, 3095. [CrossRef] [PubMed]
22. NVDLA. Unit Description [Online]. 2018. Available online: http://nvdla.org/hw/v1/ias/unit_description.html (accessed on 1 April 2022).
23. Vivado Tools. Unit Description [Online]. 2021. Available online: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2021_2/ug910-vivado-getting-started.pdf (accessed on 1 April 2022).
24. ZCU102 Board. Unit Description [Online]. 2019. Available online: https://www.xilinx.com/support/documents/boards_and_kits/zcu102/ug1182-zcu102-eval-bd.pdf (accessed on 1 April 2022).