Optimizing FPGA-based Convolutional Neural Networks Accelerator for Image Super-Resolution

Jung-Woo Chang and Suk-Ju Kang
Dept. of Electronic Engineering, Sogang University, Seoul, South Korea
{zwzang91, sjkang}@sogang.ac.kr

Abstract – Convolutional neural networks (CNN) are widely used in various computer vision applications. Recently, there have been many studies on FPGA-based CNN accelerators to achieve high performance and power efficiency. Most of them have been on CNN-based object detection algorithms, but researches on image super-resolution have been rarely conducted. Fast super-resolution CNN (FSRCNN), well known for CNN-based super-resolution algorithm, are a combination of multiple convolutional layers and a single deconvolutional layer. Since the deconvolutional layer generates high-resolution (HR) output feature maps from low-resolution (LR) input feature maps, its execution cycles are larger than those of the convolutional layer. In this paper, we propose a novel architecture of the FPGA-based CNN accelerator with the efficient parallelization. We develop a method of transforming a deconvolutional layer into a convolutional layer (TDC), a new methodology for the deconvolutional neural networks (DCNN). There is a massive parallelization source in the deconvolutional layer where multiple outputs within the same output feature map are created with the same inputs. When this new parallelization technique is applied to the deconvolutional layer, it generates the LR output feature maps the same as the convolutional layer. Thus, the performance of the accelerator increases without any additional hardware resources because the kernel size required to generate the LR output feature maps is smaller. In addition, if there is a DSP underutilization problem in the deconvolutional layer that some of the processors are in an idle state, the proposed method solves this problem by allowing more output feature maps to be processed in parallel. Experimental results show that the proposed TDC method achieves up to 81 times higher throughput than the state-of-the-art DCNN accelerator with the same hardware resources. We also improve the speed by 7.8 times by having all layers in the hourglass-type FSRCNN to be processed in inter-layer parallelism without additional DSP usage.

I. Introduction

Convolutional neural networks (CNN) can solve the problems faced by existing machine learning algorithms in such as computer vision [1, 2] and natural language processing [3]. However, high computational complexity is required to achieve the satisfactory performance with CNN in related fields. It is difficult to solve by using multi-core central processing units (CPUs), and graphic processing units (GPUs) can run complex CNN at high speeds. However, significant energy consumption of the GPUs is limited in energy-constrained embedded systems [1]. On the other hand, FPGAs and ASIC chips have been attracting much attention as CNN accelerators because of their high power efficiency and massive parallel processing which are faster than CPUs [4, 5]. In particular, FPGAs are flexible enough to cope with CNN's rapid evolution into better models. Therefore, many researchers have studied the hardware implementation for CNN using FPGAs [5, 6].

CNN are used not only for classifying objects in an image, but also for improving the image quality on displays. In recent years, many researches have been actively conducted to apply CNN to a super-resolution (SR), and it has greatly improved the performance compared to existing methods [2]. Fast super-resolution convolutional neural networks (FSRCNN) are one of the most powerful and efficient algorithm among CNN-based super-resolution algorithms. It has a combination of multiple convolutional layers and a single deconvolutional layer [7]. In the convolutional layer, the input blocks are downscaled by the kernel size, while in the deconvolutional layer the input pixels are upscaled by the kernel size to generate high-resolution (HR) output feature maps from the low-resolution (LR) input feature maps [1, 8]. However, there are two main problems in the hardware implementation for the deconvolutional layer.

First, unlike the convolutional layer, which simply stores the output feature map, the deconvolutional layer requires additional operations to load the previously obtained output feature map, add it to the current output feature map, and finally store it in the memory. This problem is called an overlapping sum problem [9]. Second, in order to avoid the overlapping sum problem, multiply-accumulate (MAC) operations should be performed at the loop dimensions in HR output feature map [9]. Therefore, the execution cycles for the deconvolutional layer are larger than the convolutional layer, and the more the output resolution increases, the greater the load on the computation. Therefore, the above problems must be solved in order to run the deconvolutional layer as a hardware platform practically.

Zhang et al. [9] develop a state-of-the-art deconvolutional neural networks (DCNN) accelerator. They propose a reverse looping method that determines which inputs are needed to get the desired output. However, with the reverse looping method, the positions of input pixels should be calculated through the formulas for each loop iteration, and loop dimensions are increased as large as the HR image.

Zhang et al. [5] propose a tiling technique that effectively parallelizes the dimensions of the matrix. They also devise a convolutional layer processor (CLP) based on a roofline model [10]. However, since feature maps of all layers are different, it is inefficient to apply all layers with the single-CLP. This problem is called a resource under-utilization [11]. In addition, since the single-CLP occupies most of the DSPs, there are not enough DSPs to parallelize multiple convolutional layers. To solve this problem, a

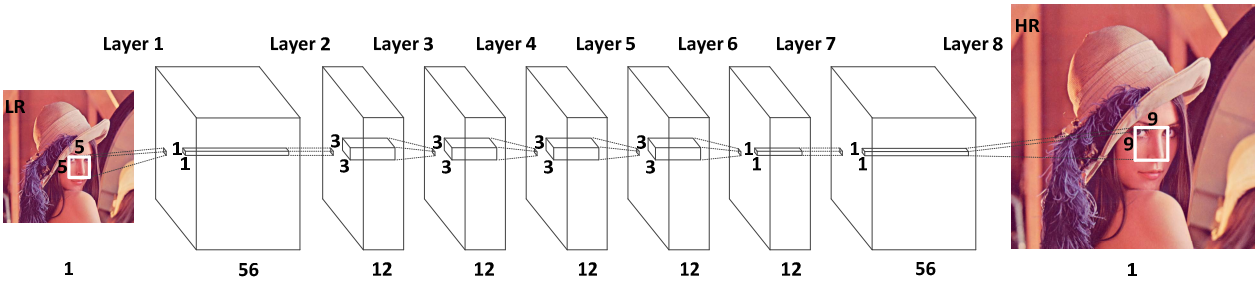

Fig. 1. Fast super-resolution convolution neural network structure [7].

TABLE I
FSRCNN configurations.

Layer	1	2	3	4	5	6	7	8
input_feature_map (N)	1	56	12	12	12	12	12	56
output_feature_map (M)	56	12	12	12	12	12	56	1
row (I_H)	256	256	256	256	256	256	256	256
col(I_W)	256	256	256	256	256	256	256	256
kernel (K)	5	1	3	3	3	3	1	9
stride (S)	1	1	1	1	1	1	1	S

multi-CLP method [11] is proposed. Each CLP is determined according to the characteristics of each layer. Therefore, they increase the DSP utilization up to 97%.

In this paper, we propose a new design methodology for FPGA-based CNN accelerator with the efficient parallelization. The main contributions of this paper are as follows.

First, we propose a method of transforming a deconvolutional layer into a convolutional layer (TDC). This method is a new design methodology for DCNN, which reduces the kernel size by performing the MAC operations in the loop dimensions of the LR image. If there is the DSP underutilization problem in the deconvolutional layer that some of the processors are in an idle state, the proposed method solves this problem. Since the TDC method increases the number of output feature maps that are performed in parallel, the processors that are in the idle state are activated to execute the MAC operations. In addition, due to the reduced kernel size in the newly created convolutional layer, we can gain the spatial benefit of the overall architecture when we design an accelerator that depends on kernel size. This is because the kernel size of the deconvolutional layer is the largest in the FSRCNN. Second, the hourglass-type FSRCNN have a serious performance degradation when it is processed with the single-CLP method. The proposed CNN accelerator is designed to maximize the effect of the multi-CLP on the layers of the FSRCNN.

II. Background

A. FSRCNN

Fig. 1 represents the FSRCNN, well known as CNN-based super-resolution algorithm. The FSRCNN totally consist of eight layers. The first seven layers are the convolutional layers and last layer is the deconvolutional

```
for ( i_h = 0 ; i_h < I_H ; i_h ++)
 for ( i_w = 0 ; i_w < I_W ; i_w ++)
  for ( m = 0 ; o_c < M ; o_c ++)
   for ( n = 0 ; i_c < N ; i_c ++)
    for ( k_h = 0 ; k_h < K ; k_h  ++)
     for ( k_w = 0 ; k_w < K ; k_w ++)
      output_feature_map[ m ][ i_h ][ i_w ] +=
      Wc[ m ][ n ][ k_h ][ k_w ]×
      input_feature_map[ n ][ S × i_h + k_h ][ S × i_w + k_w ]
```

Fig. 2. Pseudo code of a basic convolutional layer.

```
…
for ( ti_h = 0 ; ti_h < T_h ; ti_h ++)
 for ( ti_w = 0 ; ti_w < T_w ; ti_w ++)
#pragma HLS PIPELINE
   for ( t_m = 0 ; t_m < T_m ; t_m ++)
#pragma HLS UNROLL
    for ( t_n = 0 ; t_n < T_n ; t_n ++)
#pragma HLS UNROLL
     output_feature_map[ t_m ][ ti_h ][ ti_w ] +=
     Wc[ t_m ][ t_n ][ k_h ][ k_w ]×
     input_feature_map[ t_n ][ S × ti_h + k_h ][ S × i_w + k_w ]
…
```

Fig. 3. Pseudo code with tiling technique of [5].

layer. The approximate procedure of the FSRCNN is as follows. First, the convolutional layer performs feature extraction process from LR image through 56 kernels with 5×5 pixel resolution. Next, with 12 kernels, 56 output feature maps shrink to 12 output feature maps. Then, non-linear mapping processes through four layers with 12 kernels with 3×3 pixel resolution are performed. After the mapping is completed, the 56 kernels with 1×1 pixel resolution expand the feature maps back to 56 output feature maps. Finally, the deconvolutional layer ends up producing an HR image from LR input feature maps. Table I shows the FSRCNN configurations. N and M are the number of input and output feature maps, respectively I_H and I_W are the number of rows and columns in the input feature maps, respectively. K is the kernel size, and S is the stride of the layer.

B. Approaches of the CNN Hardware Implementation

The FSRCNN can be implemented in the hardware using

Fig. 4. Difference for operation of (a) the conventional method [9] and (b) the proposed TDC method.

two approaches: intra-layer parallelism and inter-layer parallelism.

- Intra convolutional layer parallelism
  Fig. 2 represents the basic pseudo code for computing the convolutional layer. It assumes that all loop parameters (i_h, i_w, m, n, k_h, and k_w) are correlated. For accelerating CNN accelerator, a single-CLP method [5] enables loop pipelining, loop tiling, and loop unrolling in the convolutional layer as shown in Fig. 3. The tiling factors (T_n, T_m) for the output feature maps and input feature maps are loaded from the off-chip memory into buffers for the parallel processing.
- Intra deconvolutional layer parallelism
  The state-of-the-art DCNN accelerator applies the tiling technique like Fig. 3 to the deconvolutional layer and parallelizes the output feature maps in the same way. However, they do not consider that the output pixels in the same output feature map can be processed in parallel. If the source for the new parallelism is taken into consideration, the deconvolutional layer can be converted to a convolutional layer.
- Inter convolutional layer parallelism
  The CNN is known to have low performance when convolutional layers are executed in parallel because there is the dependency between layers [12, 13]. However, the multi-CLP method shows that convolutional layers can be processed in parallel.

III. Proposed architecture

Fig. 4 shows the difference between the conventional method [9] and the proposed TDC method. The conventional method has a large kernel size and cannot generate the output pixels in parallel. For efficient parallelization of the FSRCNN accelerator, we propose the following two methods: TDC method and multi-CLP optimization for the FSRCNN.

A. TDC method

As shown in Fig. 4, our method first determines an input block size K_C that can generate an output pixel that does not overlap between neighboring blocks. Since the output block overlaps with the neighboring blocks by the deconvolutional layer's kernel size K_D, we first need to calculate the distance between the output pixel and the center pixel of the output block that furthest overlaps. Since each output block is spaced by S_D, which denotes the stride of the deconvolutional layer, the distance D between theses pixels

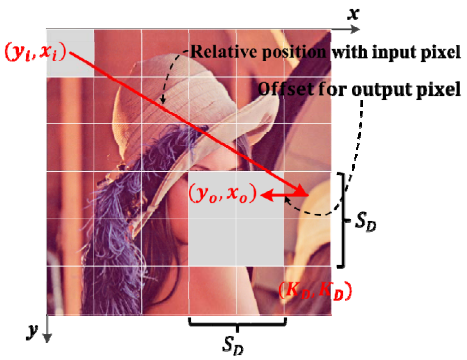

Fig. 5. Example of inverse coefficient mapping.

TABLE II
FSRCNN configurations after TDC method.

Layer	1	2	3	4	5	6	7	8
input_feature_map (N)	1	56	12	12	12	12	12	56
output_feature_map (M)	56	12	12	12	12	12	56	S_D^2
row (I_H)	256	256	256	256	256	256	256	256
col(I_W)	256	256	256	256	256	256	256	256
kernel (K)	5	1	3	3	3	3	1	K_C
stride (S)	1	1	1	1	1	1	1	1

is given by:

$$D = \left\lfloor \frac{K_D}{2} \right\rfloor \times \frac{1}{S_D}. \tag{1}$$

At this time, a position of the center pixel in the output block is the same as that of the pixel for when the input pixel is mapped to the output feature map as shown in Fig. 4(b). Thus, D is the number of input pixels in one direction that can determine the value of the output pixel. Therefore, K_C is equal to the number of pixels considered in both directions as follows:

$$K_C = \begin{cases} 2\times\lfloor D \rfloor + 1 = 2\times\left\lfloor \left\lfloor \frac{K_D}{2} \right\rfloor \times \frac{1}{S_D} \right\rfloor + 1, & \text{if } D < 0.5 \\ 2\times\lceil D \rceil = 2\times\left\lceil \left\lfloor \frac{K_D}{2} \right\rfloor \times \frac{1}{S_D} \right\rceil, & \text{otherwise} \end{cases}. \tag{2}$$

If D is greater than 0.5, K_C is even because the center pixel of the output block is not unique.

In the FSRCNN, K_D is set to 9 and S_D is 2 or 3 or 4 depending on the output feature map size, and hence, K_C can be 3 or 5 using (2). Therefore, an input block with $K_C \times K_C$ pixel resolution produces an output pixel without the overlapping sum problem.

In the deconvolutional layer, each input block moves by S_D pixels, so multiple output pixels can be created with the same input block. Therefore, there is an important source for parallel processing. In this process, output blocks with $S_D \times S_D$ pixel resolution can be generated through same input block. If each output pixel is generated by parallel processing, the individual outputs are the same as results for performing the convolution of inputs and corresponding weights. By doing

TABLE III
Performance comparison between layer specific optimal solution and cross-layer optimization.

	Optimal Unroll Factor $< T_m, T_n >$	Execution Cycles $\times 1000$
Conv 1	<56, 1>	1638
Conv 2	<6, 56>	131
Conv 3	<12, 12>	589
Conv 4	<12, 12>	589
Conv 5	<12, 12>	589
Conv 6	<12, 12>	589
Conv 7	<56, 6>	131
Conv 8	<4, 56>	1638
Total	-	5898
Cross-Layer Optimization	<9, 56>	18415

so, it is possible to parallelize all output blocks with $S_D \times S_D$ pixel resolution in the same output feature map. At this time, the number of the output feature maps is multiplied by $S_D \times S_D$ pixels, the resolution of each output feature map is reduced from HR to LR, and the kernel size is reduced from K_D to K_C. Therefore, the proposed TDC method solves the overlapping sum problem and the low performance of the state-of-the-art DCNN accelerator.

Next, we introduce how to obtain the weights of newly created convolutional layer. Let (x_i, y_i), (x_o, y_o), and (x_d, y_d) denote coordinates of an input pixel, an output pixel, and an weight of a deconvolutional layer, respectively. The range of each pixel is:

$$\begin{aligned} 0 \le x_i, y_i < K_C, \\ 0 \le x_d, y_d < K_D, \\ 0 \le x_o, y_o < S_D. \end{aligned} \tag{3}$$

To obtain weights of the convolutional layer, we need to find (x_d, y_d) corresponding to (x_i, y_i). (x_d, y_d) can be obtained by the following inverse coefficient mapping process.

The relative position for an input pixel and its corresponding weight is calculated by subtracting (K_D, K_D) from (x_i, y_i) multiplied by S_D. Then, by subtracting the offset associated with each of the $S_D \times S_D$ output pixels from the relative position, (x_d, y_d) corresponding to (x_i, y_i) can be obtained. The calculation for the whole process is as in Fig. 5. Thus, we get the following equations.

$$x_d = K_D - S_D \times x_i - (S_D - (x_o \bmod S_D)), \tag{4}$$

$$y_d = K_D - S_D \times y_i - (S_D - (y_o \bmod S_D)). \tag{5}$$

Therefore, the weights of the newly created convolutional layer $\mathbf{W}_C$ can be mapped to the weights of the deconvolutional layer $\mathbf{W}_D$ using (x_d, y_d) corresponding to (x_i, y_i). The process of obtaining $\mathbf{W}_C$ is illustrated by (6).

$$\begin{aligned} &\mathbf{W_C}[S_D^2 \times t_m + S_D \times y_o + x_o][t_n][y_i][x_i] = \\ &\mathbf{W_D}[t_m][t_n][y_d][x_d]. \end{aligned} \tag{6}$$

Finally, both K_C and $\mathbf{W}_C$ are obtained. Compared with [9], the proposed TDC method does not need to calculate every loop iteration to obtain K_C and $\mathbf{W}_C$ since they are always fixed according to K_D, S_D, and $\mathbf{W}_D$. Thus, we can reduce the pipeline depth. Table II shows the changed FSRCNN configurations after applying the TDC method.

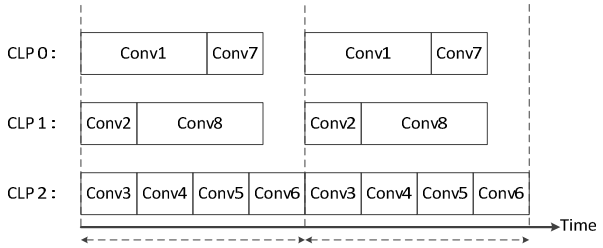

Fig. 6. Example of a multi-CLP processing.

After applying the TDC method, the output feature maps and the input feature maps are processed in parallel by the loop tiling technique as in [9]. The number of execution cycles can be obtained from (7), where R_H and C_H denote the height and width of the HR image, respectively, R_L and C_L denote the height and width of the LR image, respectively, and P denotes pipeline depth.

$$\begin{aligned} &\text{Number of execution cycles} \\ &= \left\lceil \frac{S_D{}^2 \times M}{T_m} \right\rceil \times \left\lceil \frac{N}{T_n} \right\rceil \times \frac{R_H}{S_D} \times \frac{C_H}{S_D} \times (K_C \times K_C + \frac{P}{T_r \times T_c}) \\ &= \left\lceil \frac{S_D{}^2 \times M}{T_m} \right\rceil \times \left\lceil \frac{N}{T_n} \right\rceil \times R_L \times C_L \times (K_C \times K_C + \frac{P}{T_r \times T_c}). \end{aligned} \tag{7}$$

Depending on the range of M, there are three different cases in terms of the performance enhancement compared to the state-of-the-art DCNN accelerator.

Case 1. $M \le T_m / (S_D \times S_D)$

The TDC method allows parallelization of all the output feature maps with $(S_D \times S_D) \times M$. Therefore, the performance enhancement is:

$$\text{Performance enhancement} = S_D^2 \times \frac{K_D^2}{K_C^2}. \tag{8}$$

Case 2. $T_m / (S_D \times S_D) < M < T_m$

The TDC method is capable of the limited parallelism for output feature maps with $(S_D \times S_D) \times M$. The performance enhancement is:

$$\text{Performance enhancement} = \frac{S_D^2}{\left\lceil \frac{(S_D{}^2 \times M)}{T_m} \right\rceil} \times \frac{K_D^2}{K_C^2}. \tag{9}$$

Case 3. $M \ge T_m$

The TDC method does not fully parallelize the output feature maps with $(S_D \times S_D) \times M$. However, since the convolution is performed using the smaller kernel size in the LR image, the execution speed is faster in all cases. The performance enhancement is:

$$\text{Performance enhancement} = \frac{S_D{}^2 \times \left\lceil \frac{M}{T_m} \right\rceil}{\left\lceil \frac{(S_D{}^2 \times M)}{T_m} \right\rceil} \times \frac{K_D^2}{K_C^2}. \tag{10}$$

TABLE IV
Comparison between the conventional method [9] and proposed method for the DCNN accelerator.

Conventional method [9]		Proposed method	
S_D	*Cycles*×1000	K_D	*Cycles*×1000	K_C
2	21233	9	1638	5
3	47775	9	589	3
4	84934	9	1179	3

B. Multi-CLP Optimization for the FSRCNN

CNN with hourglass-type has a serious problem of DSP underutilization when processing all layers with one CLP. Table III shows the comparison of the execution cycles when all layers are processed with optimal tiling factors through roofline mode and they are treated with an equivalent tiling factor. At this time, S_D is set to 2. As shown in Table III, the throughput is reduced by about 212% when the single-CLP method is used. In the case of AlexNet [1], the performance degradation was within 5% [5]. Comparing the two results, there is a more severe performance degradation in FSRCNN. Therefore, in the hourglass-type CNN like the FSRCNN, the multi-CLP method that reflects the configuration of each layer should be used.

Fig. 6 shows an example of convolutional layers handled by multi-CLP method. The multi-CLP configuration used in the hourglass-type FSRCNN is determined by the input feature maps and output feature maps of the convolutional layers. First, 1st and 7th layers can be treated as same CLP with high proportion of T_m because the output feature map is larger than the input feature map. Second, since the input feature map and the output feature map are the same in 3rd, 4th, 5th, and 6th layers, T_n and T_m of the processor can be assigned the same weight. Finally, 2nd and 8th layers can be processed by a processor with a large proportion of T_n because the input feature map is larger than the output feature map. However, the output feature map of 8th layer can be increased depending on S_D. Since the resource allocation of CLP that processes 8th layer is out of order, the resource redistribution is required depending on S_D. The optimization method to solve this problem is to reduce the resources of other CLPs and increase the resources of the CLP that processes the 8th layer. This increases the efficiency of parallel processing of FSRCNN.

IV. Experimental results

We demonstrated the performance of the proposed accelerator on the Xilinx Virtex-7 485T FPGA. We compared the performance of the proposed TDC method with the state-of-the-art DCNN accelerator. Then, we confirmed how much the performance of the hourglass-type FSRCNN was improved by processing with the multi-CLP method than the single-CLP method. We implemented the proposed architecture and other architectures using Vivado HLS 2016.4 and used a single-precision floating point.

Table IV shows the execution cycles obtained when the state-of-the-art DCNN accelerator is applied to the deconvolutional layer of the FSRCNN and when the proposed TDC method is applied. Since both methods used the tiling technique, the loop tiling factors (T_n, T_m) used in the experiment were set to (56, 9) using the roofline model. Experimental results showed that the proposed method had a throughput up to 81 times higher than the conventional method. There were two reasons for the increase in the throughput. First, when the TDC method was applied, the kernel size was reduced from K_D to K_C as shown in Table IV. Second, since the number of output feature maps of the deconvolutional layer was one, most of the processors were idle. On the other hand, when TDC method was applied, the number of output feature maps increased from 1 to $S_D \times S_D$, so the DSP underutilization problem could be resolved. If S_D was 2, the number of output feature maps was increased to 4, but since the tile factor T_m was 9, resource underutilization still existed. Thus the performance improvement at this time was 12.96 times, which was the lowest among other results.

TABLE V
Resource utilization for the single-CLP and multi-CLP.

		BRAM	DSP	FF	LUT
Single-CLP		574 (27%)	2520 (90%)	151395 (24%)	142711 (47%)
Multi-CLP	$S_D = 2$	627 (30%)	2413 (86%)	232323 (38%)	175972 (58%)
	$S_D = 3$	627 (30%)	2413 (86%)	232330 (38%)	176034 (57%)
	$S_D = 4$	606 (29%)	2333 (83%)	224834 (37%)	170418 (56%)

Next, we set the number of CLPs of the FSRCNN accelerator to three. Table V shows resource usage of each design. Since there were more computation modules in the single-CLP, the DSP usage was higher than the multi-CLP. However, the multi-CLP required more flip-flops and look-up tables (LUT) counts because it required more logics for the parallel processing and address calculation. In addition, since the computation engine varied according to S_D, but the loop tiling factors did not change much, so there was little difference in resources by S_D. The layers processed by each CLP are shown in Table VI. Each CLP built a specialized computation engine depending on the feature maps of the processing layers. Experimental results showed that the performance of the multi-CLP method was up to 7.8 times faster than the single-CLP method, even though the single-CLP method used more DSPs when S_D was 2. When S_D was 4, the FSRCNN lost a little of hourglass shape because of the 8th layer, so the performance improvement was 3.2 times.

V. Conclusion

In this paper, we proposed the TDC method based DCNN accelerator. The proposed method had the following advantages. First, as the output resolution was reduced from HR to LR by transforming the deconvolutional layer into the convolutional layer, the kernel size was reduced from K_D to K_C. It improved throughput and could resolve the spatial problem of accelerator architecture. Second, since the pixels

TABLE VI
Performance comparison between the single-CLP method and the multi-CLP method according to S_D for the FSRCNN accelerator

		$S_D = 2$				$S_D = 3$				$S_D = 4$			
		T_n	T_m	Layers	Cycles×1000	T_n	T_m	Layers	Cycles×1000	T_n	T_m	Layers	Cycles×1000
Single CLP	CLP0	56	9	1	11468	56	9	1	11468	56	9	1	11468
				2	131			2	131			2	131
				3	1179			3	1179			3	1179
				4	1179			4	1179			4	1179
				5	1179			5	1179			5	1179
				6	1179			6	1179			6	1179
				7	458			7	458			7	458
				8	1638			8	1638			8	3276
	Overall				**18415**				**18415**				**20054**
Multi CLP	CLP0	2	56	1	1638	1	56	1	1638	1	56	1	1638
				7	393			7	786			7	786
	CLP1	56	4	2	196	56	5	2	196	56	6	2	131
				8	1638			8	3276			3	1179
												8	4915
	CLP2	12	12	3	589	12	12	3	589	6	12	4	1179
				4	589			4	589			5	1179
				5	589			5	589			6	1179
				6	589			6	589				
	Overall				**2359**				**3473**				**6225**

in the same output feature map were extended to different output feature maps, the parallelism of the output feature maps could be increased. Experimental results showed that the proposed DCNN accelerator was performed up to 81 times faster than the conventional DCNN accelerator. We also demonstrated that our FSRCNN accelerator improved the throughput up to 7.8 times when it was processed in the multi-CLP method rather than the single-CLP method.

Acknowledgements

This research was supported by the Ministry of Trade, Industry and Energy (MOTIE) and the Korea Institute for the Advancement of Technology (KIAT) (N0002436, 2017) and Korea Electric Power Corporation. (Grant number R17XA05 -28)

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." in *Advan- ces in neural information processing systems*. 2012.

[2] C. Dong, C. C. Loy, K. He, and X. Tang. "Learning a deep convolutional network for image super-resolution." in *European Conference on Computer Vision 2014, 184-199. Springer,* 2014.

[3] R. Collober and J. Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning." in *Proceedings of the 25th international conference on Machine learning*. *ACM*, 2008.

[4] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning." in *ACM Sigplan Notices. Vol. 49. No. 4. ACM*, 2014.

[5] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. "Optimizing fpga-based accelerator design for deep convolutional neural networks." in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. *ACM*, 2015.

[6] M. Peemen, A. A. Setio, B. Mesman, and H. Corporaal. "Memory-centric accelerator design for convolutional neural networks." in *Computer Design (ICCD), 2013 IEEE 31st International Conference on. IEEE*, 2013.

[7] C. Dong, C. Loy, and X. Tang. "Accelerating the Super-Resolution Convolutional Neural Network." in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

[8] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. "Deconvolutional networks." in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE*, 2010.

[9] X. Zhang, S. Das, O. Neopane, and K. Kreutz-Delgado. "A Design Methodology for Efficient Implementation of Deconvolutional Neural Networks on an FPGA." in *arXiv preprint arXiv:1705.02583*, 2017.

[10] S. Williams, W. Andrew, and D. Patterson. "Roofline: an insightful visual performance model for multicore architectures." in *Communications of the ACM*, Apr 2009.

[11] Y. Shen, M. Ferdman, and P. Milder. "Overcoming resource underutilization in spatial CNN accelerators." in *Field Programm- able Logic and Applications (FPL), 2016 26th International Conference on. IEEE*, 2016.

[12] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun. "Neuflow: A runtime reconfigurable dataflow processor for vision." in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on, pages 109-116. IEEE*, 2011.

[13] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi. “Design space exploration of fpga-based deep convolutional neural networks.” in *Design Automation Conference (ASP-DAC), 21st Asia and South Pacific. IEEE*, 2016.