A Rotation-based Data Buffering Architecture for Convolution Filtering in a Field Programmable Gate Array

Zhijian Lu
College of Computer Science and Technology Harbin Engineering University, Harbin, China
Email: luzhijian@hrbeu.edu.cn
Yanxia Wu, Zhenhua Guo, Guochang Gu
College of Computer Science and Technology Harbin Engineering University, Harbin, China
Email: {wuyanxia, guozhenhua, guguochang}@hrbeu.edu.cn

Abstract—**Convolution filtering applications range from image recognition and video surveillance. Two observations drive the design of a new buffering architecture for convolution filters. First, the convolutional operations are inherently local; hence every pixel of the output feature maps is calculated by the neighboring pixels of the input feature maps. Even though the operation is simple, the convolution filtering is both computation-intensive and memory-intensive. For real-time applications, large amounts of on-chip memories are required to support massively parallel processing architectures. Second, to avoid access to external memories directly, the data that are already stored in on-chip memories should be used as many times as possible. Based on these two observations, we show that for a given throughput rate and off-chip memory bandwidth, a rotation-based data buffering architecture provide the optimum area-utilization results for a particular design point, which are commonly used applications in recognition area.**

Index Terms—**convolution filtering, Field Programmable Gate Arrays (FPGAs), data buffering**

I. Introduction

Convolution filters are the computational models that are widely used in recognition and video processing domains [1][2][3][4]. The computation of convolution requires not only the high computational capability but also large memory bandwidth, especially when high-definition images and videos have to be processed in real-time. In these applications, convolution filtering plays an essential role [5][6]. Generally, external memories are used to contain input image pixels, but the memory bandwidth cannot satisfy the requirement of the optimal throughput directly. Hence intermediate buffers by means of on-chip memories are adopted to avoid access to external memories directly [7][8]. To load as many pixel values as needed to the convolution filter in one cycle, multiple memory ports are attached to intermediate data buffers. Once a pixel value is loaded, it can be reused for the corresponding successive convolutions to avoid accessing it from off-chip memories repetitively. As a result, the requirements for off-chip memory bandwidth are reduced.

Convolution architecture with a complete convolution architecture is adopted in [7], where a set of linear shift registers are used to move a $R \times S$ window over the input image. The input image is divided in rows, each with a fixed length according to the input image row length, and the height according to the convolution window height. Each pixel in the input image needs to be loaded only once to the intermediate data buffer and with a fixed minimum external memory bandwidth. In case the size of input image or convolution window become large, FPGA implementations become very expensive, which will cost a significant amount of FPGA resources [7][8].

There are alternative buffering architectures that internal buffers only store a small portion of pixels [7][9]. Each group of shift registers in the convolution window receives the pixels belonging to consecutive rows of input image. Compared with the aforementioned methods, a great shift register reduction is achieved. However, multiple-dataflow is needed to feed data to the internal buffer. Pixels in the input image need to be read repetitively from external memories depending on the size of convolution window. And to keep the maximum throughput rate, this leads to a sharp increase in terms of external memory bandwidth requirement.

In this paper, we are concerned with the implementation of convolution filters in FPGA and we design a alternative buffering architecture for convolution filters that shows good balance between on-chip resource utilization and external memory bus bandwidth.

II. Rotation-based Data Buffering Architecture

Yanxia Wu is the corresponding author.

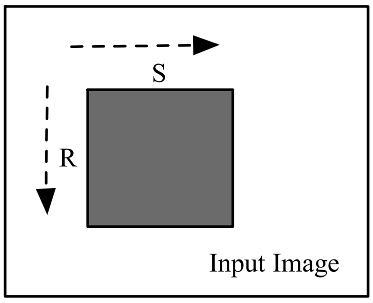

Figure 1. Conceptual view of an $R \times S$ convolver and an $M \times N$ image

In this section, we will first introduce the convolution filtering implementation strategy. The advantages and disadvantages of existing implementation architectures will be discussed. Then we will present the rotation-based data buffering architecture. In Fig. 1, we show the conceptual view of a $R \times S$ convolution filter moving over an $M \times N$ input image, which will be used in the following sections.

A. Convolution Filter Implementation Strategy

The $R \times S$ convolution of an image is defined by equation 1:

$$Y'_{m,n} = \sum_{i=-(R-1)/2}^{(R-1)/2} \sum_{j=-(S-1)/2}^{(S-1)/2} Y_{m+i,n+j} \cdot W_{i,j} \quad (1)$$

where $Y'_{m,n}$ is the convolved pixel on the output image, $Y_{m,n}$ is the pixel value from the input image, and $W_{i,j}$ is the convolution kernel weight. To calculate the convolution $Y'_{m,n}$, each pixel $Y_{m,n}$ from a $R \times S$ window of input image centered on (m, n) is multiplied by the corresponding convolution kernel of weights, and then the products are accumulated to produce the output value. Because the two-dimensional convolution $Y'_{m,n}$ of each pixel $Y_{m,n}$ requires the values of its $R \times S - 1$ immediate neighbors before being able to process that pixel, more columns than needed will be read within the same transaction. Each output pixel requires $R \times S$ multiply-accumulations, all of which can be performed in parallel. To accelerate the computation of convolution filter, multiple data in a convolution window need to be accessed simultaneously, so the calculations can be performed in parallel.

B. Multiple Dataflow Single Convolution Architecture (MDSCA)

In order to eliminate the shift register arrays in [7], multiple dataflow single convolution architectures are adopted in [8][10]. In these architectures, small portion of image pixels are loaded to the convolution filter. However, with fewer shift register arrays, the pixels can no longer be loaded to the convolution window in zigzag order. Instead of that, pixels belonging to consecutive rows are read into the shift register simultaneously. Groups of FIFOs are included to feed the pixels to the shift registers. After one column of pixels are fed into the convolution filter, the convolution window moves to a next position.

Fig. 2 shows a multiple dataflow single convolution architecture using an input/output bus, which can completely eliminate the shift register arrays in [7]. The convolution window pixel registers receive the pixels belonging to consecutive rows of the original image through R stacks. Multiple dataflow single convolution architecture requires much larger bandwidth than the single dataflow architecture. The shift register arrays are completely eliminated. Extra memory bandwidth is used to reduce the number of shift registers. To compute a single cycle $R \times S$ convolution, one new pixel per row is needed at every cycle. The total of R pixels transferred and one result produced means that a bandwidth of $R + 1$ bytes per cycle is needed.

C. Single Dataflow Complete Convolution Architecture (SDCCA)

To avoid directly access to external memories, FPGA on-chip memories are used as intermediate data buffers [7]. In Fig. 3, a single dataflow complete convolution architecture, makes use of on-chip shift register arrays to move a $R \times S$ window over the input image. To extract pixels from input image, a single dataflow strategy has been adopted. Pixels are fed from external memories in a zigzag order, until $R - 1$ complete lines and the first S pixels in the next line are contained within a series of linear shift registers. From that moment on, all the pixels belonging to the first $R \times S$ convolution window are available for the processing element. Each time a new pixel is loaded, the convolution window moves to a new position until the entire image has been visited. The throughput of this architecture is one clock per pixel. In [7], $R - 1$ sets of shift registers with a length of $N - S$, are employed to keep data before moving them to the convolution filter, and R sets of registers, each with S shift registers, are used for the convolution filter. These shift registers, which enable arbitrary size convolution filter to work with a single data stream, require no more than one pixel per clock external memory bandwidth. Pixels in the input image need to be read only once. The side-effect of this architecture is that in order to make this single data stream architecture work, $R - 1$ complete rows must be read from external memory first, therefore storing these data within a set of shift registers would be very expensive in FPGA implementation when the size of input image or the size of convolution filter is large.

D. Rotation-based Multiple dataflow Buffering Architecture (RMDBA)

In order to reuse data that are already stored in on-chip buffers as many times as possible, we proposed a rotation-based data buffering architecture. Fig.4 illustrates R continuous convolution filter in a row-wise direction, where the two adjacent filter windows share $S - 1$ columns. The architecture of these sliding windows includes R contiguous convolution filter windows, which share $S - 1$ columns in the row-wise direction. If the calculations of these R convolution kernels are performed at the same time, a much higher level of data reusing will be

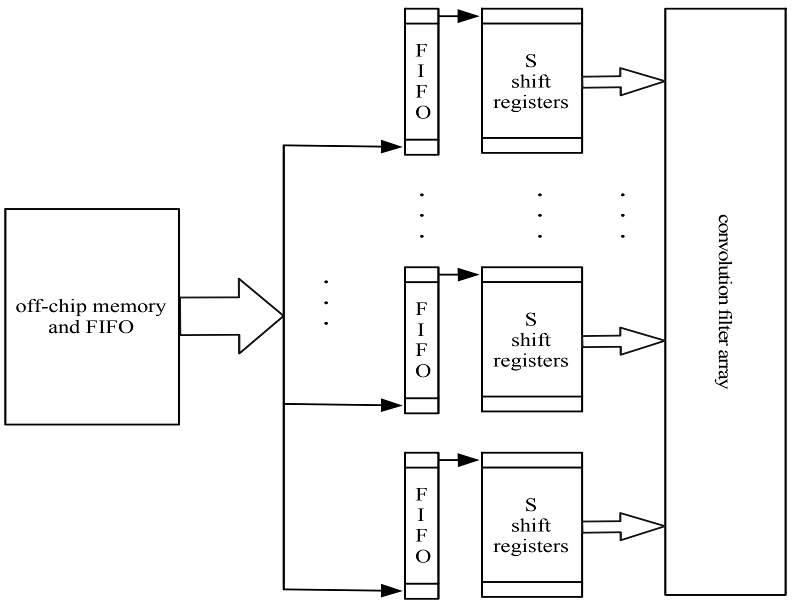

Figure 2. Multiple dataflow single convolution architecture

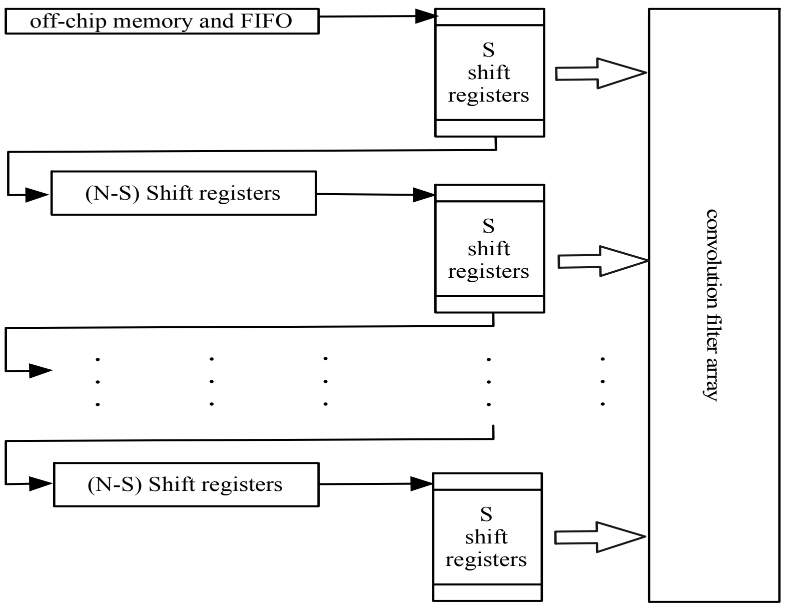

Figure 3. Single dataflow complete convolution architecture

achieved compared with the multiple dataflow single convolution architecture. Fig. 5 illustrates the rotation-based multiple dataflow architecture we proposed. The number of shift register arrays is extended to Y to hold all the pixels in the $R \times Y$ area as depicted in Fig. 4. Unlike the multiple dataflow single convolution architecture and the single dataflow complete convolution architecture, the R pixel data in each set of shift register array are not simultaneously fed to the convolution filter window, but in a serial type instead. One register in the shift register group is useable in each cycle, and a rotationally self-incrementing counter is used to address the register in the output. Consequently, Y pixels in all of a same row in the input, belonging to R adjacent windows in the row-wise direction, are available to the convolution filter in each cycle. After R cycles, all the data in the $R \times Y$ place have been sent to the convolution filter, and then shift register arrays will be updated. A new row of data will be moved in from the FIFO and moves the $R \times Y$ area to next position effectively. The architecture for the convolution filter using rotation-based data buffering architecture is not the same as the aforementioned architectures. For each $R \times S$ convolution window, input pixels are fed column-by-column, therefore one-column convolution line can be calculated, and it will take R cycles to complete all the calculation for each convolution window. When R neighboring windows are available, entire R one-column convolution can be processed simultaneously.

In order to achieve the throughput rate of 1 cycle/pixel, multiple dataflow must be loaded to update the convolution window. Compared with the multiple dataflow single

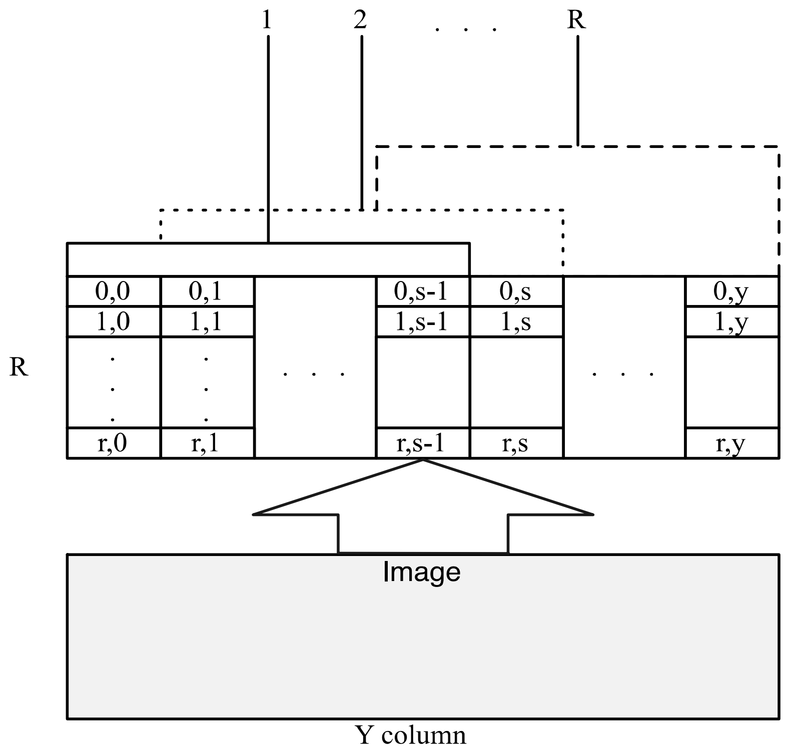

Figure 4. R simultaneous convolution windows in a $R \times Y$ area

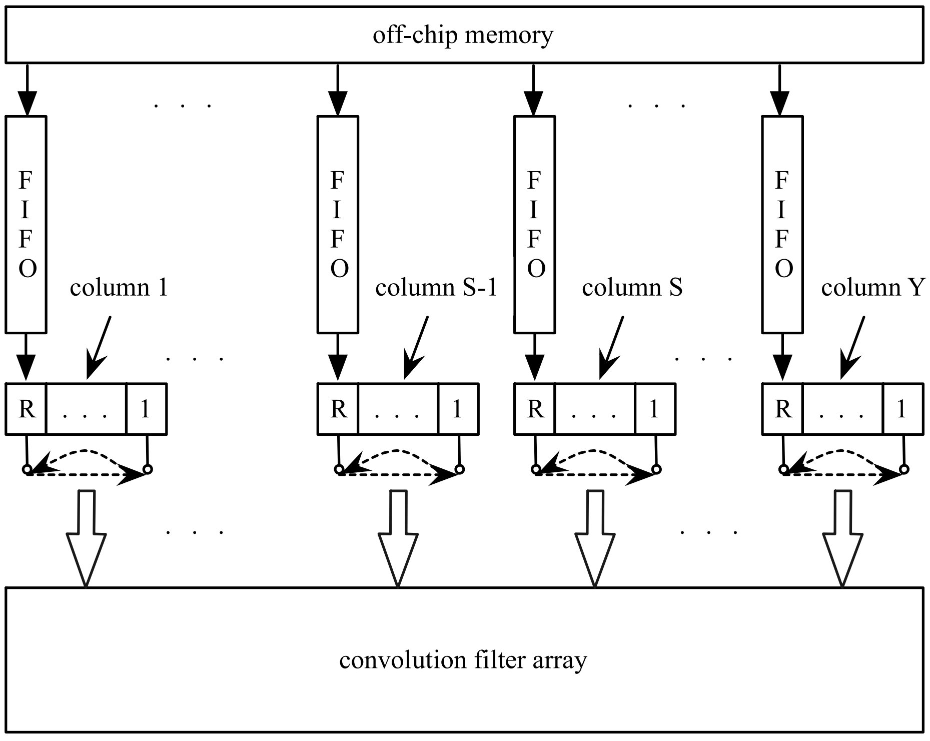

Figure 5. Rotation-based data buffering architecture

convolution architecture the window in the rotation-based architecture is updated every R cycle. In this case, shift registers can move every R cycles. Y pixels in all will be loaded from off-chip memories every R cycles. So the external memory bandwidth is Y/R pixels/clock. This means that for most convolution filter applications approximately twice of the external memory bandwidth requirement is needed.

III. Architecture Selection

In this section, we will consider an input image size of

1280×720 with 8 bits/pixel and a convolution kernel size of 7×7 as a case study. The operation will fetch image pixels from external memories, and store back to external memories after the convolution operation. In addition to this we will use a memory bus word length of 256-bits and a burst length (BL) of 8 words (i.e. 16 pixels). In Table I, we have summarized the main features of the two and the proposed architectures: area-utilization measured in terms of register pixels and memory pixels. Flip-flop count was obtained by multiplying the number of shift registers and memory pixels by bit per pixel; throughput, given in terms of cycles/pixel; and external memory bandwidth requirements, given in terms of pixels/cycle. We used different FPGA resources to implement FIFOs and shift registers depending on specific FPGA devices.

TABLE 1.
FEATURES OF DIFFERENT CONVOLUTION FILTER FOR A $R \times S$ WINDOW

architecture	register pixels	memory pixels	throughput (cycles/pixel)	ff count	bandwidth (pixels/cycle)
MDSCA	$R \times S$	$R \times BL \times ppw$	1	5496	7
SDCCA	$R \times S$	$(R-1) \times (N-S) + BL \times ppw$	1	49336	1
RMDBA	$R \times Y$	$Y \times BL \times ppw$	1	2392	1.9

TABLE 2.
AREA UTILIZATION OF DIFFERENT ARCHITECTURES FOR VARIOUS CONVOLUTION FILTER WINDOW SIZE

filter size	MDSCA	SDCCA	RMDBA
	flip-flop count	flip-flop count	flip-flop count
3 × 3	456	16536	760
5 × 5	840	32936	1512
7 × 7	5496	49336	2392
9 × 9	1800	65736	3400
11 × 11	2376	82136	4536
13 × 13	3016	98536	5800
15 × 15	3720	114936	7192
17 × 17	4488	131336	8712
19 × 19	5320	147736	10360

For comparison, the area-utilization will be evaluated in terms of flip-flops. The last two columns of Table I show the results of flip-flop count and external memory bandwidth requirement for the case study. The SCPB architecture shows the most area-efficient feature at the cost of much more requirement of the external memory bandwidth.

In order to choose the optimum architecture for a particular design point, a suitable metric that consists in maximizing the throughput with respect to the amount of resources will be used. The evaluation metric was proposed in [10] that the product throughput in terms of cycles/pixel times flip-flop number is the metric. For a particular design point, the architecture will minimize the metric value and maximize the degree of area efficiency. We used the same concept in our architecture. Table 2 shows the corresponding product of flip-flop count and throughput for convolution window size from 3 to 19 for the three architectures. We assumed a same output memory bandwidth of 1 pixel/cycle. In Fig. 6, we show the aforementioned metric comparisons and the remaining variable are the same described for the case study. In the bar diagram in Fig. 6, we can observe that RMDBA architecture is superior to the rest of the architecture for window size 7, and for the other window size MDSCA is superior. Window size 5 and 7 are the most frequently used convolution window in practical applications. As the size of input image gets larger, tradeoffs must be made, depending on different FPGA resources and available off-chip memory bandwidth.

IV. CONCLUSIONS

In this paper, we proposed a rotation-based data buffering architecture for convolution filtering in FPGA. Compared with the direct implementation of the prior-arts, the new technique requires less FPGA resources and lowers off-chip memory bandwidth and retains the optimum throughput for a particular design point, therefore it is suitable for low-cost FPGA implementation.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China No. 61003036 and the Natural Science Foundation of Heilongjiang Province of China under Grant No. QC2010049 and Fundamental Research Funds for the Central Universities (No. HEUCFT1202, No. HEUCF100606).

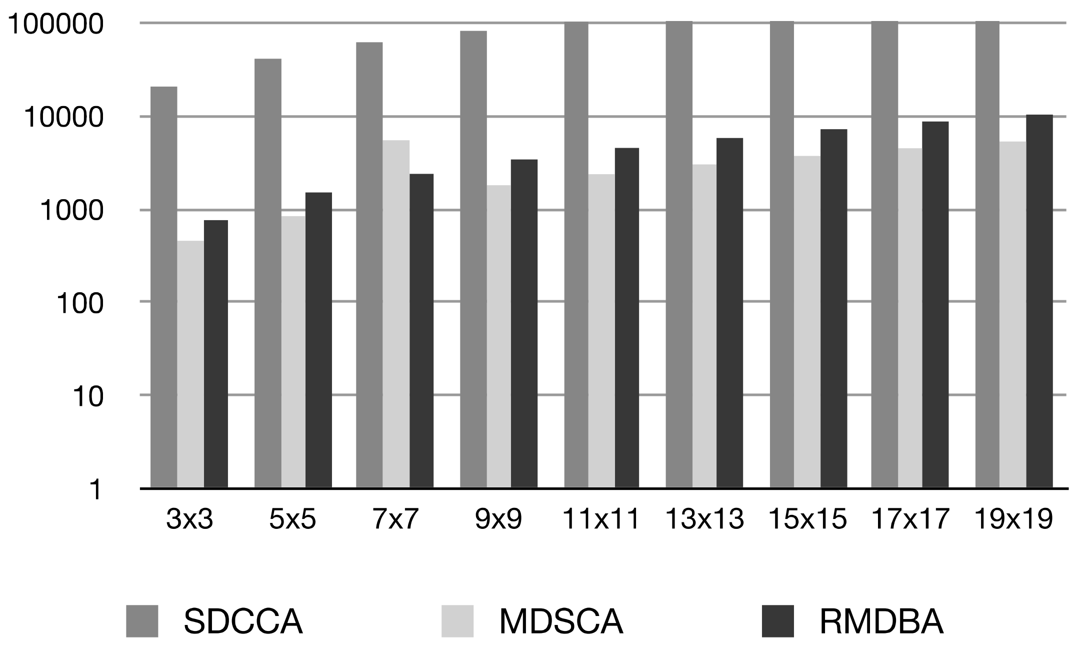

Figure 6. Bar diagram comparing the area efficiency metric for different architectures and for window sizes from 3x3 to 19x19 using the parameters of the case study. The lower the bar, the more efficient.

REFERENCE

[1] Gonzalez, R.C. and R.E. Woods, "Digital Image Processing," *Prentice Hall Press*, 2002

[2] B. S. Wu, C. C. Hsieh and C. C. Lee, "A Distance Computer Vision Assisted Yoga Learning System," *Journal of Computers*, 11(6): pp.2382-2388, 2011

[3] Z. Wang and X. Sun, "Orthogonal Maximum Margin Projection for Face Recognition," *Journal of Computers*, 2(7): pp.377-383, 2012

[4] B. Zhu and W. Jin, "Radar Emitter Signal Recognition Based on EMD and Neural Network," *Journal of Computers*, 6(7): pp.1413-1420, 2012

[5] Hecht, V. and K. Ronner, "An Advanced Programmable 2D-convolution Chip for Real Time Image Processing," *IEEE International Sympoisum on Circuits and Systems,* pp.1897-1900, 1991

[6] Leblebici, Y., et al., "A Fully Pipelined Programmable Real-time (3×3) Image Filter Based on Capacitive Threshold-logic gates," *Proceedings of IEEE International Symposium on Circuits and Systems*, vol.3, pp. 2072-2075, 1997

[7] Bosi, B., G. Bois, and Y. Savaria, "Reconfigurable Pipelined 2-D Convolvers for Fast Digital Signal Processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* 7(3): pp. 299-308, 1999

[8] Liang, X., J. Jean, and K. Tomko, "Data Buffering and Allocation in Mapping Generalized Template Matching on Reconfigurable Systems," *The Journal of Supercomputing*, 19(1): pp. 77-91, 2001

[9] Nakajima, M., et al., "A 40GOPS 250mw Massively Parallel Processor Based on Matrix Architecture," *IEEE International Solid-State Circuits Conference*, pp.1616-1625, 2006

[10] Cardells-Tormo, F. and P.L. Molinet, "Area-efficient 2-D Shift-variant Convolvers for FPGA-based Digital Image Processing," *IEEE Workshop on Signal Processing Systems Design and Implementation,* pp. 209-213, 2005

Zhijian Lu is a Ph.D. student in College of Computer Science and Technology of Harbin Engineering University, Harbin, China. His current research interest includes neural network, reconfigurable computing and image processing.

Yanxia Wu is Associate Professor in College of Computer Science and Technology of Harbin Engineering University, Harbin, China. Her current research interests include safe compiler, reconfigurable compiler and computer architecture.

Zhenhua Guo is a Ph.D. student in College of Computer Science and Technology of Harbin Engineering University, Harbin, China. His current research interest includes reconfigurable computing and embedded system.

Guochang Gu is Professor in College of Computer Science and Technology of Harbin Engineering University, Harbin, China. His main research interests include embedded systems and safe compiler.