A Fine-Grained Pipelined 2-D Convolver for High-Performance Applications

Mahdi Kalbasi and Hooman Nikmehr

***Abstract*—2-D convolution has been used as a subsystem for filtering and enhancement in a wide range of signal and image processing applications. It is not only a memory intensive operation but also a compute intensive process. The previous efforts to improve the performance of 2-D convolution have primarily focused on the memory access challenges. However, the convolver's performance is highly dependent on the efficient design of the computation units, which can be enhanced significantly by employing some techniques such as pipelining. In this brief, a pipelined with low pixel access rate architecture for implementation of 2-D convolution is presented. Compared to the conventional convolvers, the proposed design involves a fixed (independent to the problem/kernel) size, and a significantly shorter critical path specifically for large kernel sizes where the proposed convolver works with 283-MHz clock frequency, on a Xilinx Virtex-7 (XC7V2000t) field-programmable gate array, for a 3 × 3 kernel. Additionally, the required pixel access rate of the new scheme is less than that of the state-of-the-art methods, which is only 849 Mb/s for a 3 × 3 kernel and 8-bit pixels. The improvements in the critical path delay and the required pixel access rate are obtained without significant increase in the resource utilization.**

***Index Terms*—2-D convolution, pipelining, real-time image processing, large kernel size.**

I. Introduction

Two-dimensional (2-D) convolution is one of the most widely used operators exploited in machine vision and signal/image processing applications [1], specially as a preprocessing operator. For example, in edge detection processing, the convolution operator is used in two successive stages [2]. As another example, parallel convolutional operators are widely exploited in today's convolutional neural networks used in applications such as image and video recognition, recommender systems and natural language processing [3], [4]. Therefore, the implementation of a convolver which is not only fast but also easy to use and low resource consumption in today's systems is a necessary requirement.

It is well known that 2-D convolution is a memory intensive operation. For a $k \times k$ convolution kernel, k^2 image pixel values must be simultaneously available and delivered to calculate the convolution of just one output pixel. This can be a very demanding task especially for large kernel sizes. To access this amount of input data for real-time applications, in previous works a buffering module was employed in advance of the convolution module. For example, in the full buffering scheme proposed in [5], the pixel access rate of one pixel/cycle was achieved using large buffers. However, this buffering scheme consumes a huge amount of on-chip resources, especially when the convolution kernel size is large. To solve this problem, some partial buffering schemes have been proposed that require fewer on-chip resources [5], [6]. As expected, the pixel access rate of such methods is greater than those of the full buffering counterparts. As a whole, in buffer-based design approaches, there is a trade-off between the pixel access rate and the resource consumption.

Computational complexity, as another challenge, has also been investigated in the literature. For a $k \times k$ convolution kernel, k^2 multiplications and $k^2 - 1$ additions are required to calculate the convolution of a single pixel. For example, to support a 30 frames-per-second (fps) full-HD video with a 7×7 kernel, more than 6 Giga-Operations-per-Second (GOpS) are required.

Several attempts have been made to reduce the computational complexity of 2-D convolution. One class of methods attempts to reduce the critical path as well as the resources utilization using multiplier-less constant multiplication methods for fixed coefficient kernels [7]–[10]. In these approaches, the performance is improved and the resource utilization is reduced at the expense of having fixed and kernel dependent convolvers. However, these convolvers have a limited number of kernels; thus, they can only be used in specific (sets of) applications.

Another set of ideas for improving the performance of 2-D convolution aims to reduce the computational complexity by dividing 2-D convolution into smaller parts. For example, in [5], the 2-D convolver is partitioned into a number of one-dimensional (1-D) convolution modules. In [11], based on the recurrently decomposable (RD) filter design method, a 2-D convolver is proposed that separates the convolution mask into a series of smaller masks. The reduction of resource utilization is the main achievement of such methods; however, this approach leads to increase in critical path delay, and, in some cases, reduction of the throughput.

In another class of methods, various pipelining techniques, mostly with coarse stages, are employed to enhance the throughput of the 2-D convolver. Having expressed convolution as a sum-of-products operation between the

Manuscript received January 13, 2018; revised March 13, 2018; accepted April 26, 2018. Date of publication May 1, 2018; date of current version December 20, 2018. This brief was recommended by Associate Editor M. Alioto. *(Corresponding author: Hooman Nikmehr.)*

The authors are with the Department of Computer Architecture, Faculty of Computer Engineering, University of Isfahan, Isfahan 81746-73441, Iran (e-mail: nikmehr@eng.ui.ac.ir).

Digital Object Identifier 10.1109/TCSII.2018.2832064

image's pixels and the elements of the kernel, the pipelined convolver usually includes separate stages to accommodate a buffering module, a multiplication module, and an adder tree [8], [12], [13]. Although the pipelined convolvers demonstrate high throughput results, increasing the operation clock frequency is confined due to the high computational load of each pipeline stage.

In this brief, a heuristic design for a pipelined convolver is proposed that combines the buffering, multiplication, and addition modules to organize a new pipelined convolver with finer stages. The proposed design offers three main advantages:

1) Due to its fine-grained configuration, the critical path of the proposed pipelined convolver is shorter than that of the conventional coarse-grained counterparts; consequently, it can operate at a higher frequency.
2) In the proposed architecture, the pipeline registers temporarily hold some of the data necessary for the next pixel computation. Thus, the pipeline registers and buffering registers are merged to reduce the resource utilization.
3) Since a portion of the data used in the calculation of the current pixel convolution is already stored in the pipeline registers, the presented convolver requires a lower pixel access rate.

This brief is structured as follows. In Section II, the computational analysis and mathematical formulation of the new pipelined convolver are presented. The proposed 2-D convolver is explained in Section III. Comparisons of field-programmable gate array (FPGA) implementations are provided in Section IV. Finally, Section V presents the study's conclusions.

II. Computational Analysis and Mathematical Formulation

2-D convolution with a $k \times k$ kernel can be expressed as Equation (1)

$$Y(m,n) = \sum_{j=1}^{k}\sum_{i=1}^{k} h(i,j)X(m-\left\lceil\frac{k}{2}\right\rceil+i, n-\left\lceil\frac{k}{2}\right\rceil+j) \quad (1)$$

where h, X, and Y represent the convolution kernel, the input image, and the output image, respectively. The unrolled representation of Equation (1) for a 3×3 kernel is expressed in Equations (2) and (3), respectively, for two successive outputs: $Y(m,n)$ and $Y(m,n+1)$

$$\begin{aligned} Y(m,n) = {} & h(1,1)X(m-1,n-1) + h(1,2)X(m-1,n) \\ & + h(1,3)X(m-1,n+1) + h(2,1)X(m,n-1) \\ & + h(2,2)X(m,n) + h(2,3)X(m,n+1) \\ & + h(3,1)X(m+1,n-1) + h(3,2)X(m+1,n) \\ & + h(3,3)X(m+1,n+1) \quad (2) \end{aligned}$$

$$\begin{aligned} Y(m,n+1) = {} & h(1,1)X(m-1,n) + h(1,2)X(m-1,n+1) \\ & + h(1,3)X(m-1,n+2) + h(2,1)X(m,n) \\ & + h(2,2)X(m,n+1) + h(2,3)X(m,n+2) \\ & + h(3,1)X(m+1,n) + h(3,2)X(m+1,n+1) \\ & + h(3,3)X(m+1,n+2) \quad (3) \end{aligned}$$

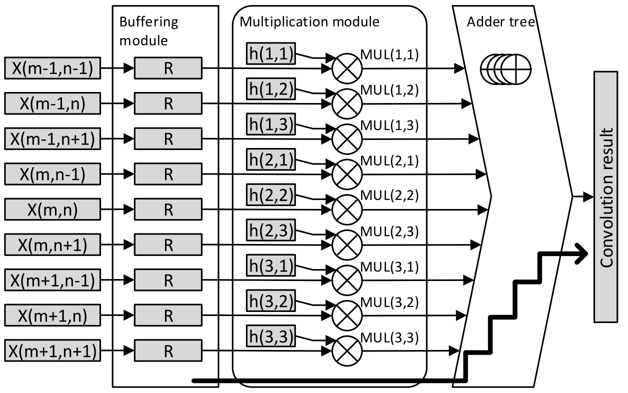

Fig. 1. Non-pipelined convolver (NPC) for a 3×3 kernel. Critical path is shown in bold line.

From these equations, it can be found that the six input values used to calculate $Y(m,n)$, including $X(m-1,n)$, $X(m-1,n+1)$, $X(m,n)$, $X(m,n+1)$, $X(m+1,n)$, and $X(m+1,n+1)$, will be used again for calculating $Y(m,n+1)$. For instance, input $X(m-1,n+1)$, which is multiplied by the kernel coefficient $h(1,3)$ in Equation (2), is multiplied this time by coefficient $h(1,2)$ in Equation (3); it is also multiplied by coefficient $h(1,1)$ to calculate the convolution of the next pixel, $Y(m,n+2)$.

This observation is generalized based on the mathematical formulation of 2-D convolution. Consequently, Equation (1) for pixels (m,n_1) and (m,n_2) can be rewritten as Equations (4) and (5), respectively:

$$\begin{aligned} Y(m,n_1) = {} & \sum_{i=1}^{k} h(i,1)X(m-\left\lceil\frac{k}{2}\right\rceil+i, n_1-\left\lceil\frac{k}{2}\right\rceil+1) \\ & + \sum_{j=2}^{k}\sum_{i=1}^{k} h(i,j)X(m-\left\lceil\frac{k}{2}\right\rceil+i, n_1-\left\lceil\frac{k}{2}\right\rceil+j) \end{aligned} \quad (4)$$

$$\begin{aligned} Y(m,n_2) = {} & \sum_{j=1}^{k-1}\sum_{i=1}^{k} h(i,j)X(m-\left\lceil\frac{k}{2}\right\rceil+i, n_2-\left\lceil\frac{k}{2}\right\rceil+j) \\ & + \sum_{i=1}^{k} h(i,k)X(m-\left\lceil\frac{k}{2}\right\rceil+i, n_2-\left\lceil\frac{k}{2}\right\rceil+k) \quad (5) \end{aligned}$$

Given that (m,n_1) and (m,n_2) are two successive image elements, substituting $n_2 = n_1+1$ and $j = j'-1$ in Equation (5) gives Equation (6):

$$\begin{aligned} Y(m,n_2) = {} & \sum_{j'=2}^{k}\sum_{i=1}^{k} h(i,j'-1)X(m-\left\lceil\frac{k}{2}\right\rceil+i, n_1-\left\lceil\frac{k}{2}\right\rceil+j') \\ & + \sum_{i=1}^{k} h(i,k)X(m-\left\lceil\frac{k}{2}\right\rceil+i, n_1+\left\lceil\frac{k}{2}\right\rceil) \quad (6) \end{aligned}$$

Studying Equations (4) and (6) reveals that they accommodate some redundant terms that can be temporarily held in registers to reduce the number of pixels required in 2-D convolution calculations.

Regarding convolution calculation that corresponds to the first pixel of the next row, a specific padding strategy is taken

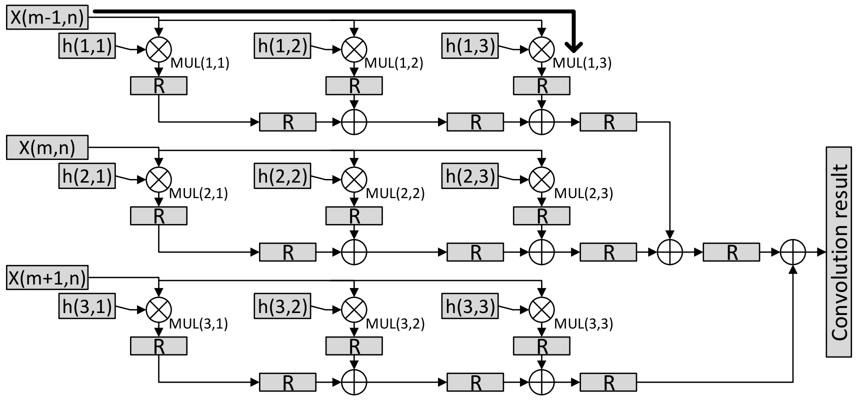

Fig. 2. Proposed reduced-access pipelined convolver (RAPC) for a 3 × 3 kernel.

TABLE I
PARTIAL PRODUCTS OF RAPC PIPELINE DATA-FLOW

Cycle	MUL(1,1)	MUL(1,2)	MUL(1,3)	MUL(2,1)	MUL(2,2)	MUL(2,3)	MUL(3,1)	MUL(3,2)	MUL(3,3)
1	$X(m-1,n-3)$	$X(m-1,n-3)$	$X(m-1,n-3)$	$X(m,n-3)$	$X(m,n-3)$	$X(m,n-3)$	$X(m+1,n-3)$	$X(m+1,n-3)$	$X(m+1,n-3)$
2	$X(m-1,n-2)$	$X(m-1,n-2)$	$X(m-1,n-2)$	$X(m,n-2)$	$X(m,n-2)$	$X(m,n-2)$	$X(m+1,n-2)$	$X(m+1,n-2)$	$X(m+1,n-2)$
3	$X(m-1,n-1)$	$X(m-1,n-1)$	$X(m-1,n-1)$	$X(m,n-1)$	$X(m,n-1)$	$X(m,n-1)$	$X(m+1,n-1)$	$X(m+1,n-1)$	$X(m+1,n-1)$
4	$X(m-1,n)$	$X(m-1,n)$	$X(m-1,n)$	$X(m,n)$	$X(m,n)$	$X(m,n)$	$X(m+1,n)$	$X(m+1,n)$	$X(m+1,n)$
5	$X(m-1,n+1)$	$X(m-1,n+1)$	$X(m-1,n+1)$	$X(m,n+1)$	$X(m,n+1)$	$X(m,n+1)$	$X(m+1,n+1)$	$X(m+1,n+1)$	$X(m+1,n+1)$

in the proposed convolution method to maintain the redundancy of the image pixels as described by Equations (4) to (6). Traditionally, in the convolution implementation, to keep the output image's size the same as for the input image, zero-padding is used that expands the input matrix into a larger array by filling the new entries with zero. In the proposed implementation, for the last pixel of each row, the first $(k-1)/2$ pixels of the next row, and for the first pixel of each row, the last $(k-1)/2$ pixels of the previous row are used as the pad, respectively.

On the other hand, by examining Equations (2) and (3) more carefully, it can be found that, since the multiply-add modules can operate independently from each other, the convolver can be pipelined by placing stage registers at appropriate locations on the data-path to reduce the critical path delay and increase the clock frequency. However, these improvements (less pixel access rate and critical path delay) are achieved at the expense of increased resource utilization, such as more registers.

In this brief, a pipeline structure for 2-D convolution is proposed, which is organized in such a way that the added registers can act not only as the stage registers, but also as the temporary memories holding reusable terms. This approach minimizes the amount of additional resources; more details about this will be presented in the next section.

III. Proposed Pipelined 2-D Convolver

Traditionally, 2-D convolution, mathematically expressed as Equation (1), can be realized in hardware similar to the structure constructed for a 3 × 3 kernel, in Figure 1. This non-pipelined convolver (NPC) includes k^2 multipliers, k^2-1 adders, and k^2+1 input/output registers for a $k \times k$ kernel. NPC's critical path, highlighted in bold line in the figure, includes one multiplier and $\lceil \log_2 k^2 \rceil$ adders for a $k \times k$ kernel. Clearly, by increasing the kernel size, the depth of the adder tree, which indicates the number of adders in the critical path, also increases. On the other hand, in the NPC, achieving the quadratic pixel access rate of k^2 pixel/cycle for large kernels can be practically difficult.

To reduce the NPC's critical path delay and pixel access rate, the proposed reduced-access pipelined 2-D convolver (RAPC) is developed and presented in Figure 2 for $k = 3$. The partial products, produced by the RAPC's multipliers for five consecutive clock cycles, are shown as data-flow in Table I, where those needed to be stored and reused for successive outputs $Y(m,n)$, $Y(m,n-1)$ and $Y(m,n-2)$ are highlighted in three different shades. As appeared in Table I, after the pipeline's fill-up time, the convolution result is generated uninterruptedly at the rate of 1 pixel/cycle.

As presented in Figure 2, in one cycle, the RAPC architecture proposed in this brief, reads k pixels for a $k \times k$ kernel, while the pixel access rate of the conventional NPC implementations, equals k^2 pixel/cycle as can be seen in Figure 1.

Because it employs extra registers, the resource utilization of the RAPC is more than that of the NPC. However, it should be noted that while most previous hardware implementations of 2-D convolution take advantage of large buffering modules with high resource utilization, the proposed RAPC does not require any additional storage element for buffering.

As the final note in this section, because of the fine stages in its pipeline, the critical path of the proposed RAPC includes only one multiplier, regardless of the kernel size. However, as shown in Figure 1, the depth of NPC's tree adder increases for larger kernel sizes, making it's critical path longer.

IV. Evaluation Results

Both the conventional NPC and proposed RAPC designs, coded in Verilog for different kernel sizes, are implemented on the Xilinx Artix-7 (XC7A200T) FPGA device [14] using Xilinx Integrated Software Environment (ISE v14.7) with byte-sized input pixels and kernel coefficients.

A. RAPC Versus NPC

Having realized NPC and RAPC on hardware for $k = 3$, the critical path delay, hardware resources occupied and pixel access rate for each design are extracted and listed in Table II. As shown in the table, both RAPC and NPC utilize the same number of LUT slices and DSP modules; nevertheless, the proposed RAPC implementation requires extra flip-flops due to its pipelined nature. It should be noted that, compared to the total flip-flops available in today's FPGA chips, the reported extra flip-flop utilization (297 flip-flops for a 3×3 kernel) is negligible (about 0.11% for the selected device).

Table II also demonstrates that the critical path delay of NPC is almost 2.5 times larger than that of RAPC, for $k = 3$. This becomes increasingly challenging when dealing with large kernel sizes. In studying the fine-grained pipelined structure of the proposed RAPC it is revealed that its critical path proceeds through only one 8-bit multiplier unit for all kernel sizes. However, in NPC, the critical path depth increases with respect to the kernel size.

In order to have a more practical insight into the impact of the kernel size on the critical path, RAPC and NPC are implemented on the FPGA for various kernel sizes from 3 to 22; the results are presented in Figure 3. As shown in this figure, RAPC's critical path delay is almost a fixed number in all cases while, for NPC, when the kernel size increases, the critical path delay grows accordingly. The very slight growth in the RAPC's critical path delay is mainly due to more required routings in the FPGA device.

As evident from Figure 2, the RAPC design accesses the memory more efficiently than the conventional NPC schemes, especially for larger kernel sizes. For a 3×3 kernel, the proposed pipelined convolver only requires three pixels to carry out convolution of an input pixel while NPC performs the same operation on one pixel, with nine pixel access per cycle. Figure 4 compares the required pixel access rate of RAPC and NPC methods for different kernel sizes. As demonstrated in the figure, the pixel access rate of NPC increases quadratically with respect to k while, for RAPC, the increasing pace of the pixel access rate is almost linear. For example, NPC's pixel access rate equals 484 pixel/cycle, for $k = 22$, which is equivalent to 17.3 Giga-pixel/second when the convolver works at 35.7 MHz frequency. While providing such pixel access rate is a huge challenge specifically in FPGA-based designs, the proposed RAPC convolver needs a pixel access rate of only 22 pixel/cycle for $k = 22$ which equals 5.9 Giga-pixel/second when RAPC design operates at 269.3 MHz frequency.

B. RAPC Versus State-of-the-Art Convolvers

Table III compares a number of state-of-the-art 2-D convolvers [8], [12], [15], [16], implemented on different Xilinx FPGA devices, with the proposed RAPC design in terms of working clock frequency, pixel access rate and resource utilization. For the sake of fair comparison, in each case, RAPC

TABLE II
RAPC AND NPC EVALUATION RESULTS FOR $k = 3$

Design	Critical path delay (ns)	clock frequency (MHz)	Resource utilization (LUT+FF+DSP)	Pixel access rate (pixel/cycle)
RAPC	3.627	276	264 + 409 + 9	3
NPC	9.186	109	264 + 49 + 9	9

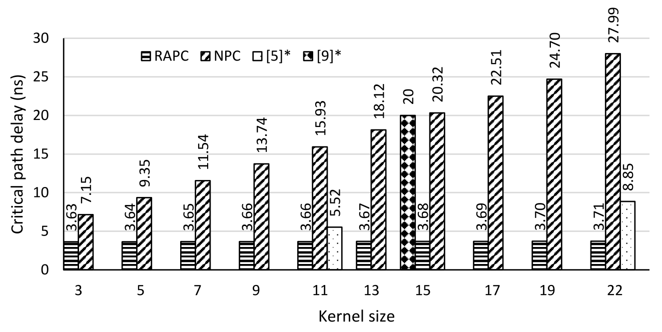

Fig. 3. Critical path delay vs kernel size for different convolution implementations (*: only reported values are presented).

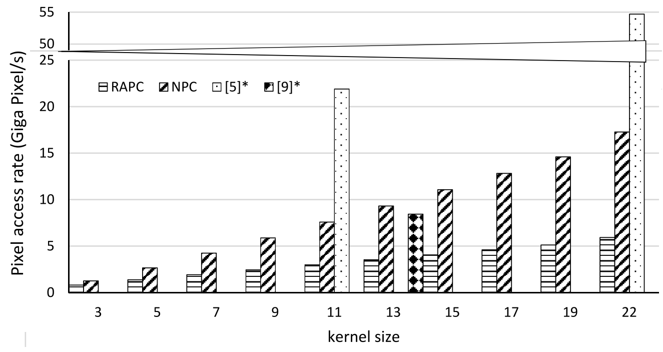

Fig. 4. Pixel access rate vs kernel size for different convolution implementations (*: only reported values are presented).

is implemented under the same condition (kernel size and FPGA device) as of the counterpart methods. Compared to the works presented in [15] and [16], in where the convolvers are optimized for the Gaussian filter for visual search applications, the proposed RAPC approach is 34% faster than hardware realization presented in [15] and nearly twice as fast as the implementation reported in [16]. Furthermore, RAPC's pixel access rate is about 45% and 65% of the pixel access rate of the FPGA implementations discussed in [15] and [16], respectively. It is worth mentioning that in contrast with [15] and [16], RAPC is not tailored for a specific application and demonstrates a great flexibility for using under different circumstances. In particular, it does not depend on the kernel characteristics such as its symmetry condition, size or coefficients value.

When the proposed RAPC is implemented on Xilinx Virtex-4 (XCV4LX25) FPGA, Table III indicates that the RAPC convolver works with four times faster clock and one-fifth pixel access rate with respect to the design presented in [12]. The last row of the table also shows the implementation results, for a large kernel with $k = 22$, extracted from

TABLE III
COMPARISON OF RESOURCE UTILIZATION, BANDWIDTH AND CRITICAL PATH OF DIFFERENT CONVOLUTION STRUCTURES (*: COMPARING [15] AND [16], RESPECTIVELY, **: N/R DENOTES "NOT REPORTED" IN THE CORRESPONDING REFERENCE)

Design	Kernel size	Device	Clock frequency (MHz)	Improvement	pixel access rate (pixel/cycle)	Resource utilization (LUT+FF+DSP)
[15]	3	XC7V2000t	212		9	4750 + N/R** + 0
proposed	3	XC7V2000t	283	33%	3	1101 + 409 + 0
[15]	3	XC6SLX16	145		9	2395 + N/R + 0
[16]	3	XC6SLX16	100		9	209 + N/R + 0
proposed	3	XC6SLX16	194	34% and 94%*	3	264 + 409 + 0
[12]	5	XCV4LX25	50		25	4612 + 4978 + 20
proposed	5	XCV4LX25	229	358%	5	696 + 1137 + 25
[8]	7	XCV4LX160	175		49	N/R
proposed	7	XCV4LX160	228	30%	7	1320 + 2161 + 49
[8]	22	XCV4LX160	177		484	18622 + 13089 + N/R
proposed	22	XCV4LX160	217	23%	22	20871 + 8888 + 0

RAPC implementation and the solution presented in [8]. In addition to faster clock frequency, another even more important advantage of RAPC that it requires much less pixel access rate (22 pixel/cycle for RAPC versus 484 pixel/cycle for the convolver introduced in [8], both for a 22×22 kernel).

V. CONCLUSION

This brief presents a high-performance 2-D convolution technique with a low pixel access rate, which makes it suitable for real-time signal and image processing applications. Various computational analyses and experimental evaluations are carried out on the proposed pipelined convolver and its hardware realization based on FPGA is explored. In contrast to the conventional pipelined counterparts with separate buffering, multiplication, and addition modules, by using the finer stages and merging the buffering and pipelining registers, the throughput and pixel access rate of the proposed convolver are improved, with a negligible increase in resource utilization. Several comparisons between the new pipelined design and the other implementations reported in the literature are made, based on their critical path delay, pixel access rate, and resource utilization. The evaluation results show that the critical path of the convolver developed in this brief is fixed for all kernel sizes and its working frequency is independent of the size or other features of the convolution kernel. The FPGA synthesis results also reveal that compared to the best available convolvers with the same kernel size, the presented pipelined design works with up to twice clock frequency and one-third less pixel access rate.

REFERENCES

[1] B. R. Masters, R. C. Gonzalez, and R. Woods, "Digital image processing," *J. Biomed. Opt.*, vol. 14, no. 2, 2009, Art. no. 029901.

[2] J. Lee, H. Tang, and J. Park, "Energy efficient canny edge detector for advanced mobile vision applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 4, pp. 1037–1046, Apr. 2018.

[3] Y. Zhao, M. Wang, G. Yang, and J. C.-W. Chan, "FOV expansion of bioinspired multiband polarimetric imagers with convolutional neural networks," *IEEE Photon. J.*, vol. 10, no. 1, pp. 1–14, Feb. 2018.

[4] Q. Wang *et al.*, "Multiscale rotation-invariant convolutional neural networks for lung texture classification," *IEEE J. Biomed. Health Inform.*, vol. 22, no. 1, pp. 184–195, Jan. 2018.

[5] B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-D convolvers for fast digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 3, pp. 299–308, Sep. 1999.

[6] H. Zhang, M. Xia, and G. Hu, "A multiwindow partial buffering scheme for FPGA-based 2-D convolvers," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 2, pp. 200–204, Feb. 2007.

[7] M. H. Sunwoo and S. K. Oh, "A multiplierless 2-D convolver chip for real-time image processing," *J. VLSI Signal Process. Syst. Signal Image Video Technol.*, vol. 38, no. 1, pp. 63–71, Aug. 2004, doi: 10.1023/B:VLSI.0000028534.35761.a8.

[8] F. J. Toledo-Moreo, J. J. Martínez-Alvarez, J. Garrigós-Guerrero, and J. M. Ferrández-Vicente, "FPGA-based architecture for the real-time computation of 2-D convolution with large kernel size," *J. Syst. Archit.*, vol. 58, no. 8, pp. 277–285, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762112000562

[9] M. Z. Zhang, H. T. Ngo, and V. K. Asari, "Multiplier-less VLSI architecture for real-time computation of multi-dimensional convolution," *Microprocessors Microsyst.*, vol. 31, no. 1, pp. 25–37, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933106001104

[10] M. Z. Zhang and V. K. Asari, "An efficient multiplier-less architecture for 2-D convolution with quadrant symmetric kernels," *Integr. VLSI J.*, vol. 40, no. 4, pp. 490–502, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926006000666

[11] Z.-B. Ma, Y. Yang, Y.-X. Liu, and A. A. Bharath, "Recurrently decomposable 2-D convolvers for FPGA-based digital image processing," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 10, pp. 979–983, Oct. 2016.

[12] F. Fons, M. Fons, and E. Cantó, "Run-time self-reconfigurable 2D convolver for adaptive image processing," *Microelectron. J.*, vol. 42, no. 1, pp. 204–217, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002626921000162X

[13] D. K. Yadav, A. K. Gupta, and A. K. Mishra, "A fast and area efficient 2-D convolver for real time image processing," in *Proc. TENCON IEEE Region 10 Conf.*, Nov. 2008, pp. 1–6.

[14] *Artix-7 FPGAs Data Sheet, v1.22*, Xilinx Inc., San Jose, CA, USA, 2017.

[15] G. D. Licciardo, C. Cappetta, and L. D. Benedetto, "FPGA optimization of convolution-based 2D filtering processor for image processing," in *Proc. 8th Comput. Sci. Electron. Eng. (CEEC)*, Sep. 2016, pp. 180–185.

[16] F. Cabello, J. León, Y. Iano, and R. Arthur, "Implementation of a fixed-point 2D Gaussian filter for image processing based on FPGA," in *Proc. Signal Process. Algorithms Archit. Arrangements Appl. (SPA)*, Sep. 2015, pp. 28–33.