Real-Time Approximate and Combined 2D Convolvers for FPGA-based Image Processing

Ali Ramezanzad
University of Isfahan
Mehran Rezaei (m.rezaei@eng.ui.ac.ir)
University of Isfahan
Hooman Nikmehr
University of Isfahan
Mahdi Kalbasi
University of Isfahan

Research Article

Keywords: 2D convolution, Approximate convolver, Real-time image processing, Power consumption, Resource utilization, FPGA

Posted Date: December 29th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2396811/v1

Additional Declarations: No competing interests reported.

Real-Time Approximate and Combined 2D Convolvers for FPGA-based Image Processing

Ali Ramezanzad [1], Mehran Rezaei [1], Hooman Nikmehr [1,2], Mahdi Kalbasi [1]*

[1] *Department of Computer Engineering, University of Isfahan, Isfahan, Iran*

[2] *UniSA STEM, University of South Australia, Mawson Lakes, South Australia, Australia*

* *Corresponding author: m.rezaei@eng.ui.ac.ir*

Abstract

Convolution widely has been used as the main part of the improvement in digital image processing applications. In convolutional computations, a large number of memory accesses and a huge amount of computations challenge its performance. Many of the related proposed convolvers are based on exact computations. Although exact convolvers keep the accuracy of the convolution operation at the top level, sometimes by missing a negligible amount of accuracy, the performance can be improved. Approximate computing is a new technique for solving computation overhead problems. In this paper, approximate 2D convolvers are presented which minimize the memory access rate and computations by a special factor of Multiply-and-Accumulate (MAC) terms. On the other hand, to preserve the flexibility for supporting different required accuracy, the proposed approximate convolvers are combined with the exact designs with real-time pre-processing stages by exploiting innovative methods which manage the hardware overhead. In comparison to conventional convolvers, the proposed designs improve the number of active resources which causes a significant reduction in power consumption. For 3×3 kernel size, the evaluation results on the Xilinx Virtex-7 (XC7V2000t) FPGA device show 34% and 20% power optimization of the proposed approximate and combined convolvers respectively, in comparison to Exact Convolver (EC). Also, this improvement grows by increasing the kernel size. Finally, a comparison based on RMSE and PSNR for different sample images and filters reveals that the error rate and image quality reduction are acceptable for many real-time image processing applications.

Keywords

2D convolution, Approximate convolver, Real-time image processing, Power consumption, Resource utilization, FPGA

Acknowledgment

This publication was supported by grant No. RD-51-9911-0039 from the R&D Center of Mobile Telecommunication Company of Iran (MCI) for advancing information and communications technologies.

1. INTRODUCTION

Two-dimensional (2D) convolution is a widespread operator used in computer vision and digital image processing applications. As an example, convolution operators are widely exploited for edge detection in advanced mobile vision applications. Also, high-pass filtering (sharpening) and low-pass filtering (blurring) are performed by 2D convolution filters [1]. For another example, in image recognition with convolutional neural networks (CNNs), there are a large number of parallel convolutional computations to extract features from the input image [2]. Therefore, the implementation of a convolver working with a low pixel access rate and optimized performance is necessary.

It is well known that the computational complexity of 2D convolution challenges its performance. To compute the convolution result between an image and a filter with $k \times k$ kernel size, $k^2 - 1$ addition and k^2 multiplications must be performed. For another challenge, 2D convolution also requires high memory bandwidth. For $k \times k$ kernel size k^2 image pixels must be read from memory for calculation of only one output [4]. These problems will be critical when the size of the kernel grows. Therefore, the performance of the convolver is significantly dependent on the design of computational units and memory bandwidth.

In image processing applications, like edge detection, the focus is on developing the most effective convolutional computation with little attention to the computational complexity and the hardware requirement. Although the implementation on personal or supercomputer platforms may not challenge the performance, in embedded applications the hardware power consumption must be considered. Therefore, a heuristic high-performance computing paradigm with improved hardware utilization is mandatory. As one of the most energy-efficient computing strategies, approximate computing has drawn research attention in the past years. In [3] different approximate computing techniques are presented. By exploiting approximate techniques, the metrics such as power consumption, critical path delay, and computational complexity can be improved at the expense of reducing a negligible amount of accuracy.

A large number of convolution-based applications are tolerant to degradation in accuracy. Many of the conventional convolvers are proposed based on the exact computation. Exact designs are suitable when the applications need high computation accuracy. But, it's at the expense of missing a huge amount of performance. In this paper, new approximate convolvers are presented to address the aforementioned limitations. In the proposed designs, by conducting pre-processing on a sample image and exploiting a power-efficient approximate computing strategy, convolutional computations for repetitive or narrow-range pixels are performed in a single time. Finally, the resource utilization of the intended hardware platform is improved significantly.

Different applications may need various levels of accuracy. Therefore, the proposed approximate convolvers are combined with the exact designs to prepare more flexibility for supporting different required accuracy. The proposed design can switch between approximate and exact convolver based on the accuracy required by applications by setting a special threshold. The selection between two convolvers for processing current convolution computation is decided by a pre-processing stage. This stage can be designed by real-time processing. However, designing a real-time decision system for the pre-processing stage increases the resource utilization, power consumption, and critical path delay of the proposed combined convolvers. Thus, it is crucial to propose an optimized real-time decision system with less hardware overhead. In addition, the approximate convolver must be structured in such a way that compensates for the real-time pre-processing stage design cost.

Convolution operation can be executed on different hardware platforms such as Application-Specific Integrated Circuit (ASIC), Graphical Processing Unit (GPU), and Field Programmable Gate Array (FPGA). Between these platforms, for hardware implementation of 2D convolution, the FPGA-based devices are suitable for three main reasons. First, it can be easily reconfigured for different architectures. Second, for exploiting parallel processing opportunities in 2D convolution, the FPGA-based platforms are the best option because of their fine-grained parallelism architecture. Finally, it can be developed faster

than other hardware platforms [4, 12-13]. For these reasons, we have selected FPGA-based platforms to implement our proposed designs.

In this paper, an innovative design for an approximate convolver is proposed that exploits narrow-range pixels to minimize the convolution computation complexity and pixel access rate. The proposed design offers five main advantages:

1) In the proposed architecture, because the Multiply-and-Accumulate (MAC) operation for narrow-range pixels is performed by a special factor of reusable terms, the required multipliers are reduced to improve resource utilization and power consumption.
2) The narrow-range pixels are considered to be in a window, row, and column of the input image for different kernel sizes to investigate various amount of error rates and performance.
3) For managing the degradation in accuracy, the proposed approximate convolvers are combined with the exact design, and new dual-purpose convolver architectures are presented which provide power optimization with negligible on-chip overhead in resource utilization.
4) New real-time pre-processing stages are designed with exploiting pipeline processing and an innovative decision system for combined convolvers with minimum hardware overhead.
5) Since the convolutional calculation provides the reusable terms, the presented convolvers require a lower pixel access rate.

The rest of the paper is organized as follows. In Section 2, the related works are mentioned. Motivation in image processing and computational analysis are presented in Section 3. The proposed 2D approximate and combined convolvers are explained in Section 4. The FPGA implementation and filtered image accuracy evaluation are provided in Section 5. Finally, Section 6 concludes the research.

2. *RELATED WORKS*

As mentioned earlier there are two main challenges for proposing an intended convolver which are computational complexity and memory bandwidth. Several approaches are considered to reduce the computational complexity of 2D convolution. In [4] a fine-grained 2D pipelined convolver is proposed. The basic idea is that in convolution computation there are some reusable computations; and, by using pipelining, the delay of computations will be decreased. At last, the critical path delay, pixel access rate, power consumption, and resource utilization of the proposed Reduced-Access Pipelined Convolver (RAPC) are compared by a conventional Non-Pipelined Convolver (NPC). Also, the authors of [5] have optimized the FPGA implementation of a convolution-based 2D filtering processor for image processing applications. The proposed filter swaps the multiplication unit with floating-point adders and also exploits a set of pre-computed coefficients to design a 32-bit multiplier module.

The second part of the literature emphasizes the use of approximate methods to improve key design metrics. In [6] by using the approximate bit-width selection strategy in the fractional part, the FPGA implementation of a fixed-point 2D Gaussian filter for image processing is proposed. As floating-point computation needs a huge amount of power consumption, designing a new fixed-point 2D Gaussian filter is essential which causes performance improvement in the processing and decreases computational costs. In [7] a two-dimensional (2D) convolver is presented in which both approximate circuit- and algorithm-level techniques are utilized. Truncation is exploited as a circuit technique while bit-width reduction is used at the algorithm level. Authors of [8] have implemented Reduced Precision Redundancy (RPR) Multiply-and-Accumulate (MAC). RPR utilizes approximated reduced precision copies instead of replicating the whole circuit which highly reduces the hardware overhead while still the largest errors can be modified. The authors of [9] have presented an FPGA-based accelerator for the Gaussian Filter by exploiting approximate computing. In this paper, based on approximate techniques the hardware architecture of the

2D convolver is modified to improve the on-chip resource utilization. For example, in the Gaussian filter, some coefficients exploited to multiply in the input pixels are repetitive. So, the input pixels multiplied by the repeated values will be added before multiplication. In [10], the authors have proposed a low-power hardware accelerator for Sobel edge detection by using an approximate gradient magnitude. Accordingly, separate gradient components are obtained for vertical and horizontal orientations in which approximate method will cause reducing the complexity of gradient computations. The authors of [11] also have applied new approximate techniques to compute gradient orientation and magnitude.

The third part of the related papers focuses on dividing 2D convolution into several smaller sections to decrease the computational complexity. For example in [12], a multi-window partial buffering approach for 2D convolvers is proposed by using FPGA platforms. In this paper, the authors focus on balancing performance and cost in using FPGA resources. Finally, the new approach causes a suitable trade-off between resource utilization and on-chip memory bus bandwidth. Also, the authors of [13] have partitioned the 2D convolver into several one-dimensional convolution sections. Other approaches try to enhance the clock frequency as well as minimize the power consumption by exploiting multiplier-less constant multiplication units for fixed elements of the kernel [14-17]. In these methods, the performance is optimized and the power consumption is improved by proposing kernel-dependent convolvers. But, the proposed convolvers can just be exploited in specific applications because the kernel sizes are limited to a special amount.

In [18], by exploiting the recurrently decomposable (RD) filter, a 2D convolver is designed where the convolution mask will be separated into a set of smaller masks. In this paper, the resource utilization is improved; but, the critical path delay is increased. In the other methods, different pipelining techniques, are exploited to increase the throughput of the proposed design. For example in [15, 19-20] the convolution is expressed as the sum-of-products among the image's pixels and the coefficients of the kernel while the ordinary pipelined convolver exploits separate pipeline stages for buffering, multiplication, and adder modules. Also, the proposed design works with high clock frequency, but it is at the expense of a huge computational overhead in each pipeline stage.

In [22], the 2D convolvers are compared. The comparison of the convolvers is based on four methods. The methods are named Non-Pipelined, Reduced-Bandwidth Pipelined, Multiplier-Less Pipelined, and Time-Shared convolver. Finally, the critical path delay, memory bandwidth, and resource utilization are analyzed for various convolution kernel sizes. In this paper, different convolver types for executing 2D convolution are explained. In non-pipelined convolver besides huge resource utilization, the critical path delay is large. Also, in pipelined convolver, the critical path delay is highly reduced but this reduction results in a huge amount of FPGA resources. In the multiplier-less pipelined convolver, a special constant is prepared which could multiply by a constant multiplication module; so, the flexibility is reduced. Finally, in Time-Shared convolver, the FPGA resources are significantly reduced but the computation time grows unexpectedly.

Authors of [23] have proposed an area-efficient FPGA-based reconfigurable 2D convolver for image processing. In this paper, the adjustment of logical block arrangement for the latest convolvers is analyzed. At last, the throughput and convolution computation time are compared with pre-proposed convolvers. However, this paper optimizes the computation time of different intended kernel sizes but this optimization is earned by utilizing a large number of FPGA resources.

As mentioned in related works, different innovative methods are considered to minimize the computational complexity and memory access of convolutional computation. Approximate computing has a special place among these techniques. Using approximate computing in such a way that makes a trade-off between accuracy and performance is essential. On the other hand, proposing a particular architecture that supports various levels of accuracy with switching among approximate and exact convolvers seems necessary. In this paper, the required design is proposed with power optimization by minimizing the number of active hardware resources.

3. MOTIVATION AND PRE-ANALYSIS

In this section, to motivate our intended architecture in image processing applications an innovative scenario is considered. In addition, before presenting the design of approximate and combined 2D convolvers, a computational analysis is provided which investigates the proposed designs with theoretical analysis.

3.1 *Motivation In Image Processing*

Approximate computing is a novel opportunity for solving the computation overhead challenges in image processing. There are several options in this scope. As shown in Figure 1, there are places in an image where pixels are repetitive or in a narrow range. To decrease the memory access rate while reducing the computational overhead, one way will be averaging close-range input pixels in image processing and performing the computation in a single time. For example, spatial and temporal coding have been expressed in image processing applications. In the spatial coding example, instead of sending the same pixels, just the color value and the number of repeated pixels can be sent to the computation units. In addition, in the temporal coding example, only differences from frame (i) can be sent instead of sending the whole frame at (i+1) [24]. Therefore, by conducting the computation of repetitive or close-range pixels at once, the computational overhead will highly decrease. These are our primitive conditions for proposing approximate and combined 2D convolvers for image processing applications.

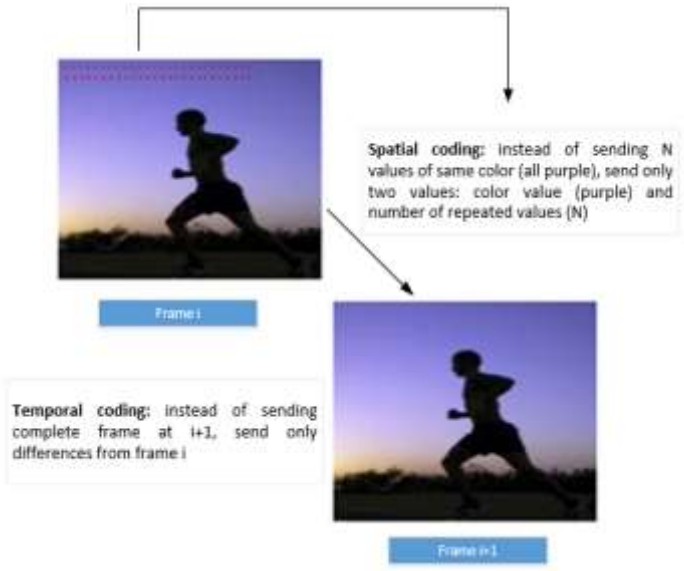

Figure 1. Spatial and temporal coding example.

3.2 *Computational Analysis*

In motivation, the existence of repetitive or close-range pixels is studied. To survey how the mentioned opportunity can be exploited in convolutional computations, the theoretical investigation helps for defining the problem. 2-D convolution with a k × k kernel can be shown as Equation (1):

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

In this Equation X, Y and h represent the input image, output image, and convolution kernel respectively. If the input pixels in a k × k window be fixed, the Equation with a k × k kernel will be as follows:

$$Y(m,n) = X(m,n)\sum_{i=1}^{k}\sum_{j=1}^{k} h(i,j) \quad (2)$$

As shown in Equation (2) if the input pixels in a k × k window be fixed, one access to the memory is required because all pixels are repetitive. Therefore, all coefficients in the k × k kernel are added, and the single input pixel is multiplied by the sum of coefficients.

Considering all pixels in a window as one single value is an ideal assumption. Since it is common that these numbers are narrow-range [24], only their average is sent to the processing element from off-chip memory. As the result, the average of all close-range pixels will be shown as Equation (3) and finally, Equation (2) will be changed to Equation (4):

$$Avg(X) = \frac{1}{k \times k}\sum_{i=1}^{k}\sum_{j=1}^{k} X\left(m - \left\lceil\frac{k}{2}\right\rceil + i, n - \left\lceil\frac{k}{2}\right\rceil + j\right) \quad (3)$$

$$Y(m,n) = Avg(X)\sum_{i=1}^{k}\sum_{j=1}^{k} h(i,j) \quad (4)$$

Where $Avg(X)$ is the average of all input pixels in the current window. Since the computed average is in a window of pixels, Equation (4) is named Window-based convolution.

In the aforementioned scenario, we have considered that all pixels in the selected input window are in a narrow range. Another scenario will be considering pixels in a row or column of input pixels in a narrow range. Let's consider 'i' as the index of the row and 'j' as the index of the column. Therefore, if we suppose that pixels in a row are in a narrow range (not fixed) then Equation (3) will be modified to Equation (5):

$$Avg(X_m) = Avg_m = \frac{1}{k}\sum_{j=1}^{k} X\left(m, n - \left\lceil\frac{k}{2}\right\rceil + j\right) \quad (5)$$

Where X_m denotes the m^{th} row of X. For simplification, the average of X_m has been shown by Avg_m. Then, based on Equations (4) and (5), Equation (6) will be as follows, which is named Row-based convolution. In other words, Equation (6) computes $Y(m,n)$ by multiplying the average of each row of X by the coefficients of that row.

$$Y(m,n) = \sum_{i=1}^{k} Avg_m\left(m - \left\lceil\frac{k}{2}\right\rceil + i, n\right)\sum_{j=1}^{k} h(i,j) \quad (6)$$

For another scenario, consider that pixels in a column are in a narrow range (not fixed); then, Equations (5) and (6), will be changed to Equations (7) and (8):

$$Avg(X_n) = Avg_n = \frac{1}{k}\sum_{i=1}^{k} X(m - \left\lceil \frac{k}{2} \right\rceil + i, n) \quad (7)$$

$$Y(m,n) = \sum_{j=1}^{k} Avg_n\left(m, n - \left\lceil \frac{k}{2} \right\rceil + j\right) \sum_{i=1}^{k} h(i,j) \quad (8)$$

Where Avg_n is the average of all input pixels in n^{th} column of X and Equation (8) is named Column-based convolution.

The main intuition behind the presented Window-, Row-, and Column-based methods is that in many real-world images close-range pixels exist, and the MAC operation can be performed by a simple factor of reusable terms. Finally, this approach leads to minimizing resource utilization by using fewer multipliers. More details will be discussed in section 4.

4. PROPOSED REAL-TIME APPROXIMATE AND COMBINED 2D CONVOLVERS

In this section, the design of the proposed real-time approximate and combined 2D convolvers is explained. First, the architecture of the exact convolver is presented. Figure 2 shows the exact 2D convolver for a 3 × 3 kernel size based on Equation (1). In the exact design, nine input pixels are registered and multiplied by nine kernel coefficients. After that, the partial products are added with parallel adders. For a k × k kernel size, the Exact Convolver (EC) contains k^2-1 adders, k^2 multipliers, and k^2 + 1 I/O registers (pixels/kernel coefficients/result). The critical path delay includes $\lceil \log_2 k^2 \rceil$ adders and one multiplier.

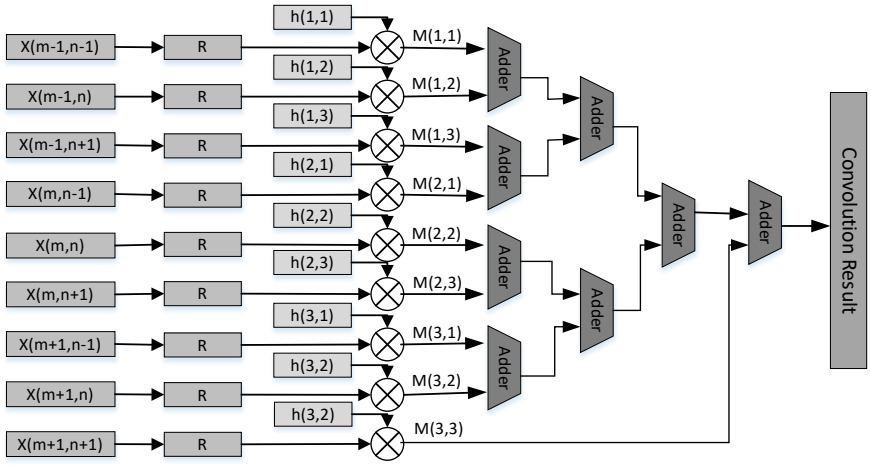

Figure 2. Exact 2D convolver for a 3 × 3 kernel.

4.1 *Real-Time Approximate Convolver*

Figure 3 shows the general architecture of the proposed real-time approximate convolver. In this architecture, a row or column of the input pixels is registered. Then, based on Equations (3), (5), and (7) the average of all input pixels in a window, row, or column is computed respectively. To implement the pipeline processing, the average value is stored before sending it to the proposed approximate convolver. At last, the final convolution result is registered to transform to the output port. It's worth noting, since in

convolution computation in image processing the sliding window equals one (stride = 1), and after the pipeline's fill-up, the approximate convolver in each cycle will produce one output result. The Binary Average (BA) is a new hardware-efficient averaging method. In this module, the operands are classified with equivalent binary values to perform the division only with the shift operator. The next sub-sections explain the BA and AWC/ARC modules in more detail.

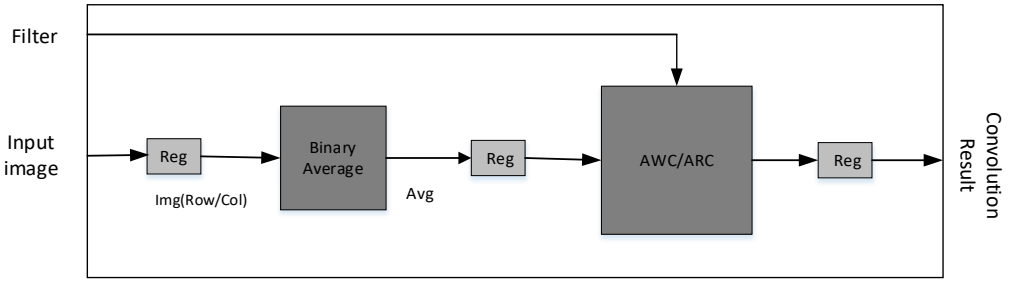

Figure 3. The general architecture of the proposed real-time approximate convolver.

4.1.1 *Approximate Window-based Convolver (AWC)*

Based on Equation (4), when the closed-range input pixels are in a window, the exact convolver is changed as shown in Figure 4. First, the averaged pixel is registered. Then, instead of adding all of the partial products in EC, the kernel coefficients are added with parallel adders. Finally, the sum of the coefficients is multiplied by the average registered pixel. For a k × k kernel size, the AWC includes k^2-1 adders, one multiplier, and two I/O registers. The critical path delay is $\lceil \log_2 k^2 \rceil$ adders and one multiplier.

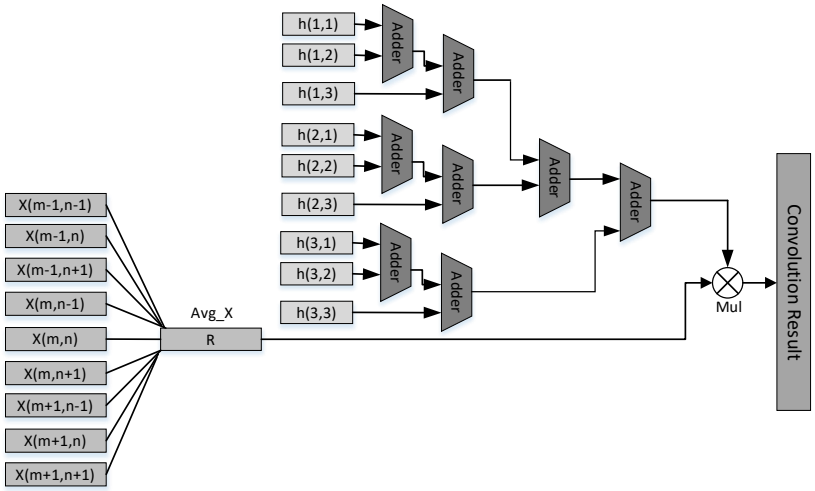

Figure 4. AWC for a 3 × 3 kernel.

4.1.2 *Approximate Row-based Convolver (ARC)*

The proposed Approximate Row-based Convolver (ARC) is presented for a 3 × 3 kernel in Figure 5. First, three input pixels are stored in three registers. As mentioned before, because three pixels in a row are narrow-range, based on Equation (5) the average pixel is transferred to the appropriate buffering module. Second, based on Equation (6), in the multiplication module, all coefficients for the equivalent row are added and the addition result is multiplied by the appropriate average term of the input pixels. Third, these product terms are added by a two-operand adder. Finally, based on the two-operand adder output, the final convolution result is computed. ARC includes k multipliers, $k^2 - 1$ adder, and k + 1 input/output registers for a k × k kernel while these amount for EC is k^2, $k^2 - 1$, and $k^2 + 1$, respectively. The Critical path of the proposed non-pipelined convolver includes $\lceil \log_2 k^2 \rceil$ adders and one multiplier for a k × k kernel.

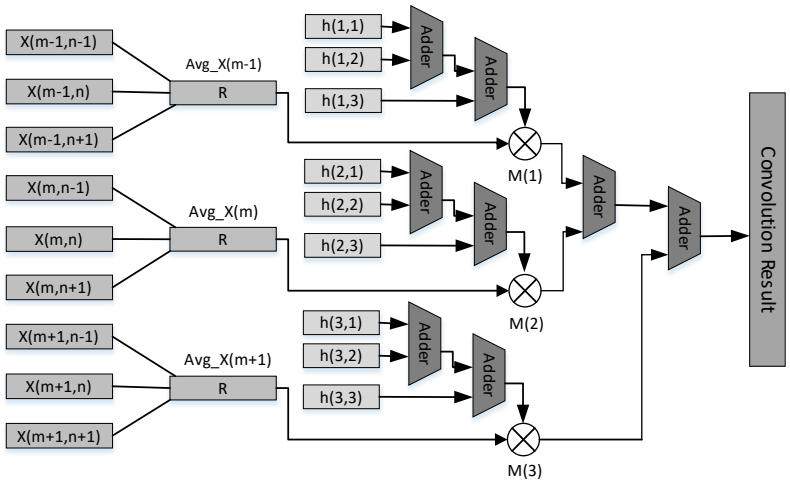

Figure 5. ARC for a 3 × 3 kernel.

4.1.3 *Binary Average (BA)*

Figure 6 shows the Binary Average for ARC with a 3 × 3 kernel. As mentioned earlier, in this module, the operands are classified in a power-of-two method which is equivalent to the binary representation of the number of operands. For example, in Figure 6 three operands must be averaged. Therefore, instead of adding all three numbers and dividing by three, we have classified them into 2 + 1 operands which is equivalent to $(11)_b$. For another example, when k =5, the binary representation of 5 is $(101)_b$. So, the operands are classified in the 4 + 1 terms. The proposed method improves the hardware overhead since the division is performed by shift operators; in addition, the number of utilized shift operators is optimized. On the other hand, because we have supposed that the pixel values are close-range, the error rate for averaging will be negligible. Based on Figure 6, for averaging nine input pixels, three pixels in the first row are selected. After that, with add and shift operators, the average of these three pixels is computed and stored in a shift register. The same process will be performed for the second and third rows. Finally, with a Serial to Parallel Shift Register (STP_SHR), all averaged values are sent to the approximate convolver. For a k × k kernel size, the BA for ARC uses k-1 adders, $\lfloor \log_2 k \rfloor$ shifters, and k registers (serial to k parallel output shift register).

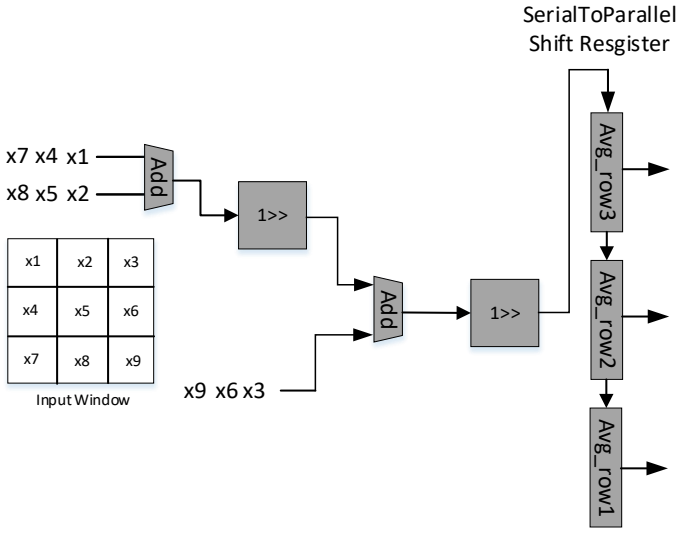

Figure 6. BA for ARC with 3 × 3 kernel size.

Similar to the previous scenario, Figure 7 shows the BA for AWC with a 3 × 3 kernel. The only difference is that a new BA module is needed for averaging all three rows to compute the average value in a window. The BA for AWC exploits 2(k-1) adders, $2\lfloor \log_2 k \rfloor$ shifters, and k registers.

It's worth mentioning, the Approximate Column-based Convolver (ACC) architecture is similar to the Row-based method in the hardware realization. In the evaluation section, two presented methods are named Real-time Approximate Window-based Convolver (RAWC) and Real-time Approximate Row-based Convolver (RARC) for the general architecture as demonstrated in Figure 3.

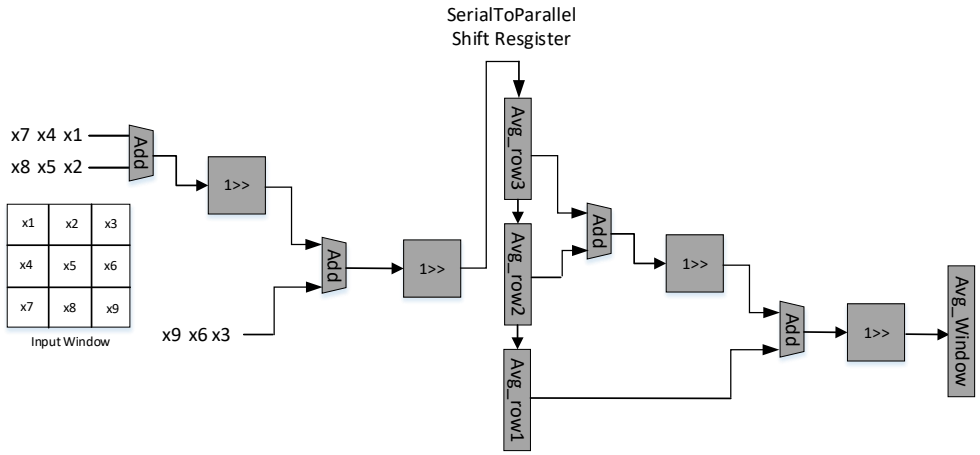

Figure 7. The BA for AWC with 3 × 3 kernel size.

4.2 *Real-Time Combined Convolver*

In the previous section, real-time approximate convolvers with two Window-based and Row-based methods are proposed. As mentioned earlier, different applications need various amounts of accuracy. So, for managing the error rate of the proposed approximate convolvers, the real-time combined 2d convolvers are presented in this section. In the new method, the proposed approximate convolvers are combined with the exact one. The general architecture of the real-time combined convolver is shown in Figure 8. Similar to previous real-time approximate convolvers, a row or column of input pixels is registered and with the BA module, the required computation is performed. Next, a Decision module is prepared to decide for selecting approximate or exact convolver based on a special relation and threshold. Also, because of the pipeline architecture of the proposed combined convolver, the internal results are stored. Finally, the input pixels are sent to CWC/CRC with three STP_SHR (one serial row to a parallel window). Similar to real-time approximate convolver, after the pipeline's fill-up time, the convolution result will be generated in each cycle uninterruptedly.

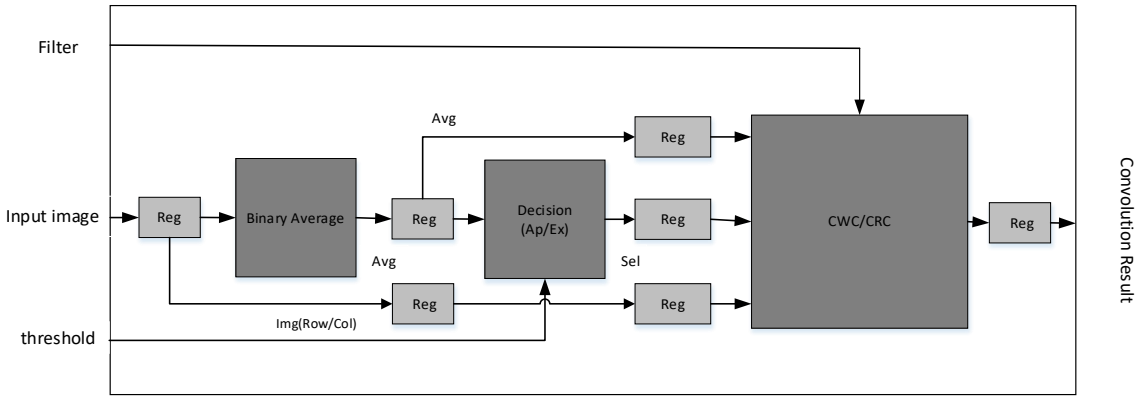

Figure 8. The general architecture of the real-time combined convolver.

4.2.1 *Combined Window-based Convolver (CWC)*

Figure 9 demonstrates the architecture of the proposed Combined Window-based Convolver (CWC) for 3 × 3 kernel size. In this convolver, the close-range pixels are considered in a window. In the exact convolver process, the select bit is equal to 1; so, the enable bit is set to 1, and nine input pixels in a window are registered. Also, the nine multiplication modules are enabled and the kernel coefficients are multiplied by the input pixels. Next, the multiplexer transforms the partial products into the output with a select bit equal to 1. The parallel adders compute the sum of partial products and guide the result to the next level of the multiplexer. In the approximate process, the nine input registers are turned off because the select and enable bits are equal to 0. So, the kernel coefficients are sent through path 0 of the multiplexer and the sum of coefficients is computed. Also, all nine multiplication units are turned off. Instead of that, one multiplier is enabled to multiply the sum of coefficients by the average of the input pixels. For a k × k kernel size, CWC uses $k^2 + 1$ multipliers, $k^2 + 1$ multiplexers, $k^2 - 1$ adders, and $k^2 + 2$ I/O registers. It's worth mentioning that in the approximate process, only $k^2 + 1$ multiplexers, $k^2 - 1$ adders, 2 registers, and one multiplier are turned on. Finally, the critical path delay of CWC is similar to RAWC and EC designs in each process respectively plus two levels of the multiplexer.

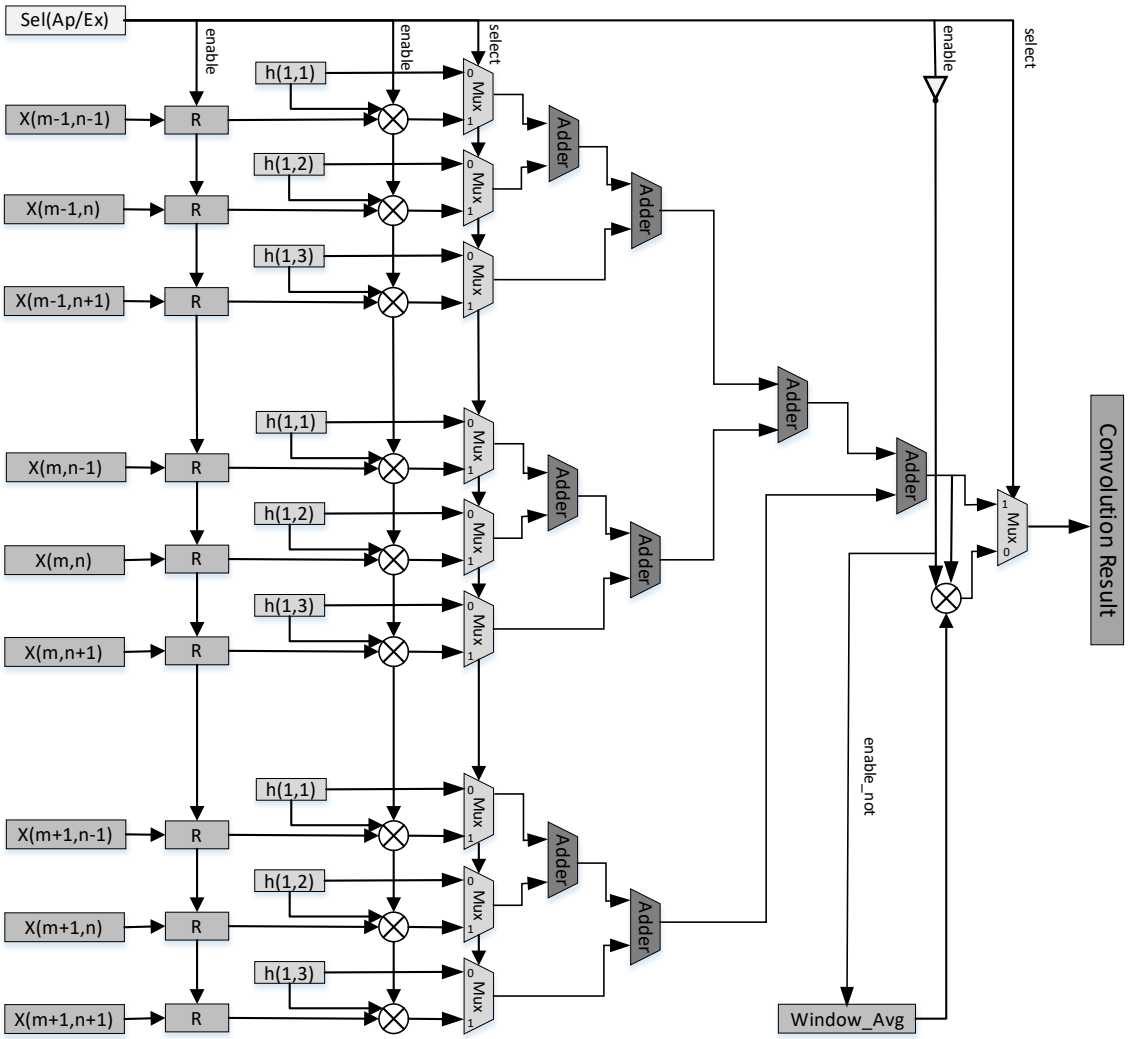

Figure 9. CWC for a 3 × 3 kernel.

4.2.2 *Combined Row-based Convolver (CRC)*

Figure 10 shows the architecture of the proposed Combined Row-based Convolver (CRC) for a 3 × 3 kernel. First, nine input pixels are registered in the buffering modules. After that, the input pixels are multiplied by the kernel coefficients for the exact convolver. Because in approximate convolver, the sum of kernel coefficients in a row is required, by using multiplexers, the coefficients and partial product are guided to two-operand adders input. With two-level of parallel adders, based on the select bit of the multiplexers, the sum of coefficients or partial products is computed. In the next level, in the approximate convolver, the sum of coefficients is multiplied by the average of each row. Next, with the new levels of multiplexers and two-operand adders, it should be decided whether to add the sum of partial products of the previous level in the exact convolver or partial products of the current level of approximate convolver. It's worth mentioning that in the proposed design we have exploited multipliers with enable pin in which the select bit decides to turn off the nine multipliers of exact computation or three multipliers of approximate convolver. For a k × k kernel size, CRC uses $k^2 + k$ multipliers, $k^2 + k$ multiplexers, $k^2 - 1$ adders, $k^2 + k + 1$ I/O registers. But, in the approximate process, $k^2 + k$ multiplexers, $k^2 - 1$ adders, $k + 1$ I/O registers, and k multipliers are activated. At last, the critical path delay is similar to RARC and EC in each process respectively plus two levels of the multiplexer.

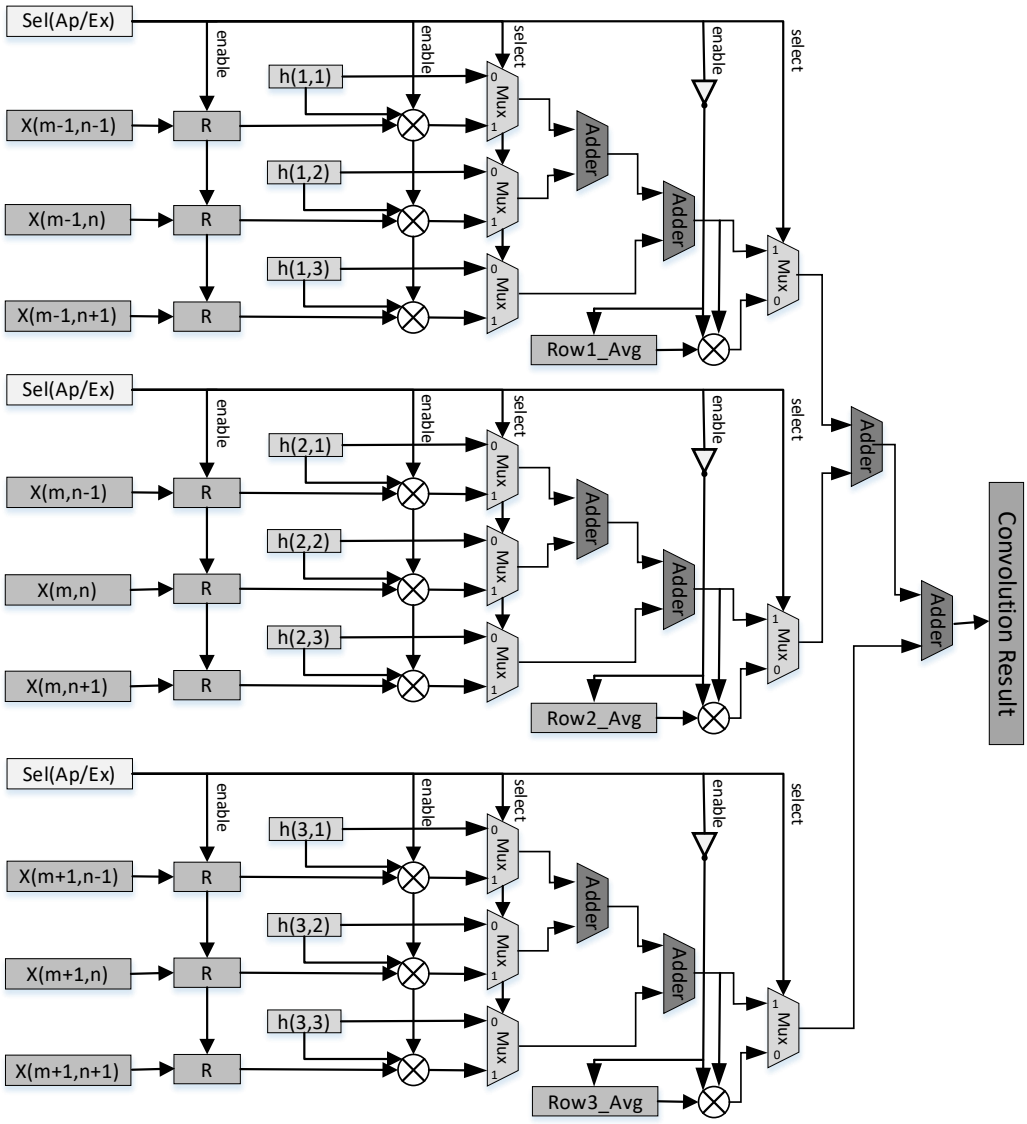

Figure 10. CRC for a 3 × 3 kernel.

It's worth noting, in sub-sections 4.2.1 and 4.2.2, CWC and CRC are demonstrated. In the evaluation section, two presented designs are named Real-time Combined Window-based Convolver (RCWC) and Real-time Combined Row-based Convolver (RCRC) for the general architecture shown in Figure 8.

4.2.3 *Decision (ApEx)*

To balance the accuracy of the filtered image and the performance of the proposed convolver, an innovative selection mechanism between approximate and exact processes is required. Therefore, a new sub-module named Decision is structured in Figure 8. Since the decision module works with real-time processing, it must be designed in such a way that not only prevents high hardware overhead but also make a reliable decision to manage the degradation in accuracy. On the other hand, since the workflow of Figure 8 is based on pipeline processing, the execution time of the Decision module must be less than the proposed combined convolver to ensure that the critical path delay will not increase.

The selection strategy of the Decision module is based on Equation (11). This relation makes a comparison between the amount of Mean Absolute Error (MAE) and a special threshold value. If MAE is less than a threshold value; then, the error rate is acceptable and the approximate process is selected and vice versa. The Decision sub-module receives the input pixels, the average value (based on Figures 6 and 7), and the threshold value as the inputs, and generates the select bit as output. It's worth mentioning, in contrast to Mean Square Error (MSE) and Root Mean Square Error (RMSE), MAE doesn't consist of square root and power operations which minimize the pre-processing overhead. The MAE formula is shown in Equation (12), where x_i is the input pixel, x_{avg} is the average of all input pixels, and n is the number of input pixels.

$$\mathrm{MAE}(original\ pixels,\ Averaged\ pixels\ ) < threshold \tag{11}$$

$$MAE = \frac{\sum_{i=1}^{n} |x_i - x_{avg}|}{n} \tag{12}$$

Figure 11 shows the architecture of the Decision module for CRC based on Equations (11) and (12). First, the input pixels are subtracted from the average of each row which was computed in Figure 6. After computing the average of absolute subtractions, with a Comparator module, the previous level result is compared with the threshold value. The output of the Comparator module specifies using approximate or exact convolver with 0 or 1 select bit respectively. Finally, with the STP_SHR module, all three prepared select bits are sent to the combined convolver. For k × k kernel size, the Decision module for CRC utilizes k subtractor, k absolute module, k-1 adders, $\lfloor \log_2 k \rfloor$ shifters, one comparator, and k registers (serial to k parallel output shift register).

The architecture of the Decision module for CWC is demonstrated in Figure 12. In this module, because the decision is made based on a window of input pixels, all pixels are subtracted from the average of the whole window which was computed in Figure 7. After computing the average of the absolute subtraction results in each row, with the STP_SHR module, the partial results are sent to the BA module. Finally, with a Comparator module, the select bit for the whole window is prepared. Finally, for a k × k kernel size, the Decision module for CWC uses k subtractors, k absolute modules, 2(k-1) adders, $2\lfloor \log_2 k \rfloor$ shifters, one comparator, and k registers (serial to k parallel output shift register).

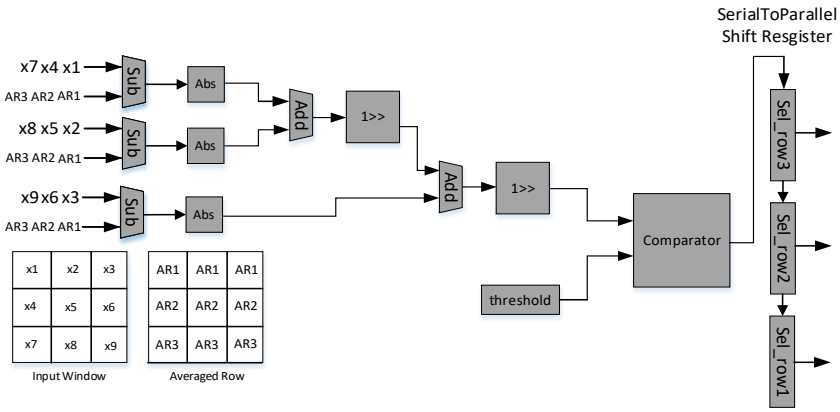

Figure 11. The architecture of the Decision module for CRC.

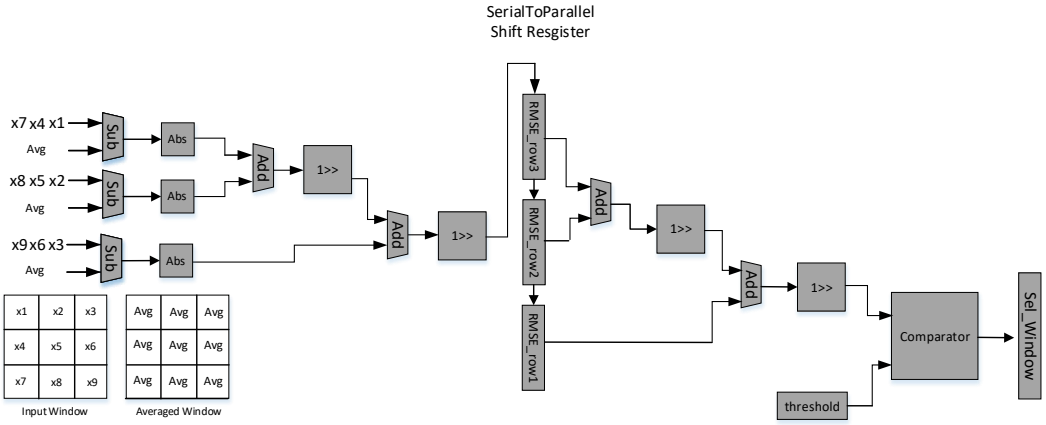

Figure 12. The architecture of the Decision module for CWC.

5 . EVALUATION RESULTS

5.1 *Hardware Evaluation*

All of the proposed approximate convolvers are coded in VHDL for various kernel sizes. We have synthesized the designs with Xilinx Virtex-7 (XC7V2000t) FPGA device by exploiting Xilinx Vivado (v2018.3) with 8-bits kernel coefficient and input pixels. To make a fair comparison, one of the related designs proposed in [4] is implemented to compare critical path delay, pixel access rate, power consumption, and resource utilization for the approximate and combined convolvers.

5.1.1 *Critical Path Delay*

In Figure 13, the critical path delay for the proposed approximate and combined convolvers is compared with EC and the proposed design in [4] which is named RAPC (Reduced Access Pipelined Convolver). As shown in this figure, RAWC has less critical path delay in comparison to EC since the 8-bit coefficients add before multiplication, and RARC's delay is close to EC approximately. But, the combined convolvers have a negligible overhead because of using two levels of multiplexers in the critical path of the circuit. For combined designs, the critical path delays for both approximate and exact paths are shown in Figure 13; but, the largest one will be selected as the critical path. Except for RAPC [4], the delay of all convolvers goes up by increasing the size of the kernel, since the number of operands for the multiply and accumulation operator will be increased. In RAPC, because it's a pipelined design, only one multiplier is in the critical path of the circuit, and the slight growth respecting the kernel size is due to the more routing

required in the FPGA device. It's worth mentioning that for future work, the proposed AWC, ARC, CWC, and CRC can be pipelined to improve the critical path delay.

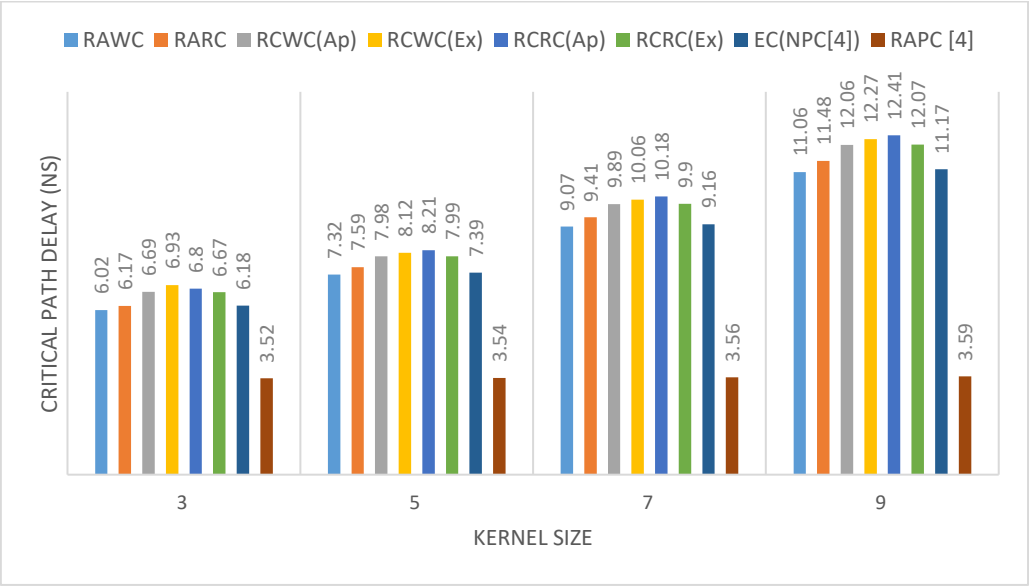

Figure 13. Critical path delay comparison among various kernel sizes.

5.1.2 *Pixel Access Rate*

Figure 14 shows the pixel access rate of the proposed designs in comparison to EC and RAPC [4]. The pixel access rate of the RAWC, RARC, RCWC, and RCRC is equal to k and grows linearly. But, EC's pixel access rate is equal to k^2 and increases quadratically with kernel size. Also, in RAPC because of using pipeline convolver, this parameter goes up linearly respecting the kernel size. Therefore, the proposed design outperforms according to the low access to the memory in comparison to EC. As mentioned before, convolutional computation needs a huge amount of input data. On the other hand, memory accesses are one of the most significant challenges for designing convolvers since off-chip DRAM access consumes the main part of chip power. So, the proposed convolver could answer these problems by minimizing the off-chip pixel access. Finally, by decreasing the pixel access rate the computational complexity of MAC operation in convolution unit will also diminish.

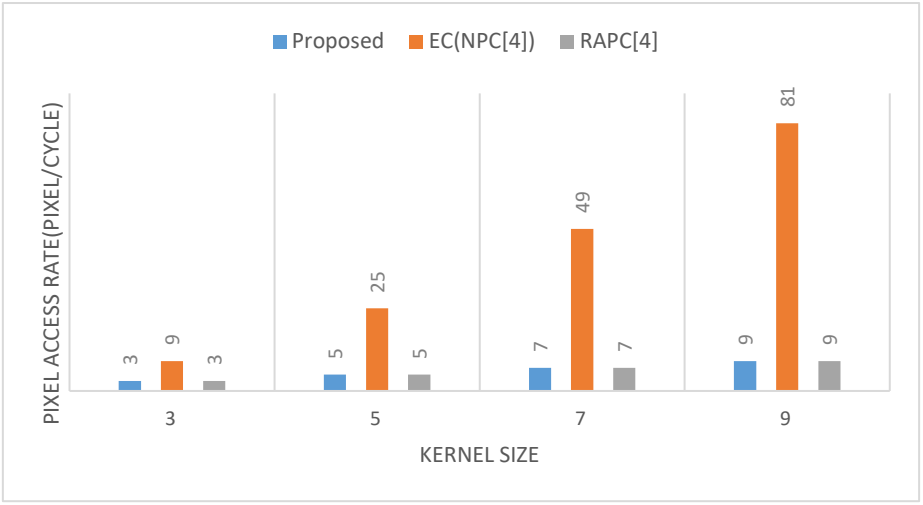

Figure 14. Pixel access rate comparison among various kernel sizes.

5.1.3 *Power Consumption*

Figure 15 demonstrates the power consumption of the approximate and combined convolvers in comparison to EC and RAPC [4] among various kernel sizes which shows the proposed design optimizes the power consumption. For instance, when kernel size is 3, RAWC and RARC consume 17.52 and 23.05 watts for on-chip power but this amount for EC grows to 26.55 watts which shows 34% and 13% improvement for mentioned designs respectively. In addition, by increasing the kernel size, the difference in power consumption goes up quadratically since the number of Look-up Tables (LUTs), Flip-Flops (FFs), and I/Os is decreased in approximate convolvers. On the other hand, in combined designs, using the approximate path (process) significantly improves the dynamic power consumption and the total on-chip power. For example, when k is 3, RCWC (Ap) consumes 21.17 watts of on-chip power which indicates 20% improvement in comparison to EC since only one multiplier and input register are turned on instead of nine in the EC module. It's worth mentioning that the reduction in power consumption goes up with increasing kernel size. Therefore, using a large number of approximate execution has a negligible effect on accuracy degradation; nevertheless, it will cause a significant reduction in dynamic power consumption.

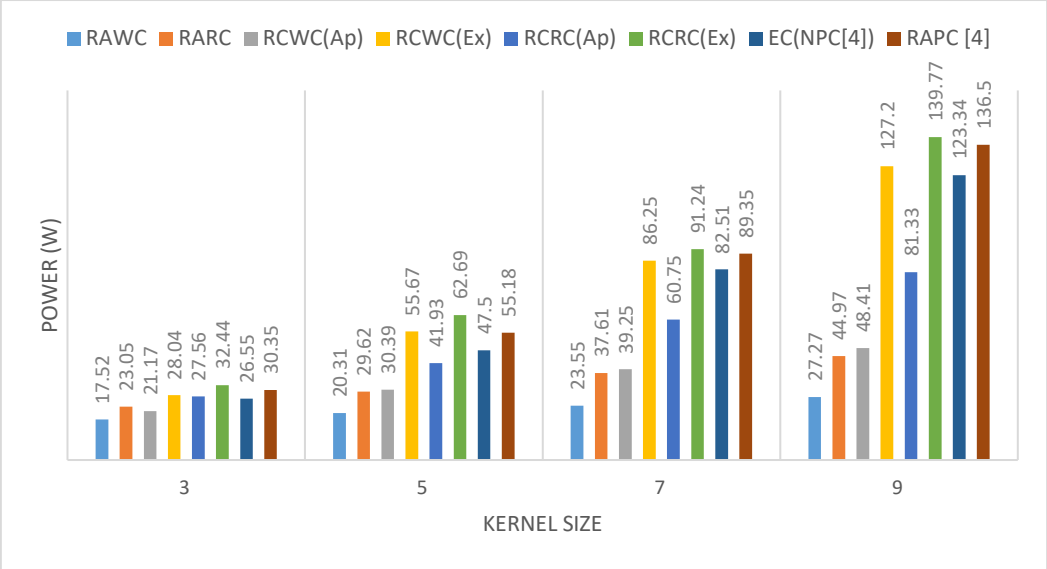

Figure 15. Power consumption comparison among various kernel sizes.

5.1.4 *Resource Utilization*

In Figure 16, the number of LUT slices is compared. By increasing the kernel size, the number of LUTs grows. The resource utilization of the proposed approximate convolver is significantly improved in comparison to EC. For instance, when the kernel size is 3, RAWC and RARC use 155 and 281 LUTs. But, this amount for EC and RAPC [4] are 767 and 771 respectively. It's worth mentioning that the total number of utilized LUTs for RCWC and RCRC is more than EC since both approximate and exact designs have been implemented; but, extra resources in each process are turned off. For example, in RCWC (Ap), the number of active LUTs is 292 from 1160 total LUTs. For this reason, in combined designs, the number of active LUTs in the approximate and exact processes also is mentioned in Figure 16.

Figure 17 shows the number of registers used for different convolvers in comparison to EC and other related works. For k = 5, RAWC and RARC use 98 and 126 FFs respectively while EC and RAPC exploit 216 and 977 of them. In combined designs, the total number of registers is more than EC (NPC [4]); but, the active FFs are highly reduced. For instance, when k = 5, the RCRC total register is equal to 343 while RCRC uses 143 FFs in the approximate process. Also, the number of FFs will increase respecting the kernel size. As can be seen in this figure, RAPC [4] utilizes more registers since exploits pipeline

processing to hold temporary internal results which is a drawback for the proposed design; however, it happened with a significant reduction in critical path delay. In combined designs, the total and active number of FFs are mentioned in each scenario to show the superiority of the approximate path when a large number of registers are turned off. For example when k = 3, the total number of FFs in the RCWC method is equal to 186 while the approximate path will turn on 114 of them which shows up to 39% improvement. Finally, when the approximate path is selected, by reducing the resource utilization, the power consumption will be reduced.

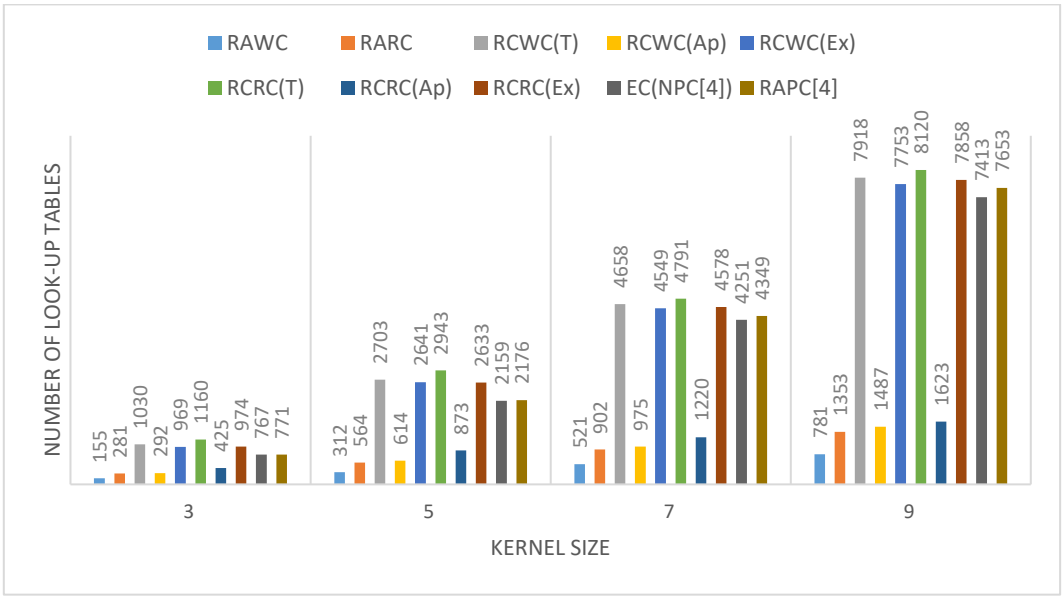

Figure 16. Number of LUTs among various kernel sizes.

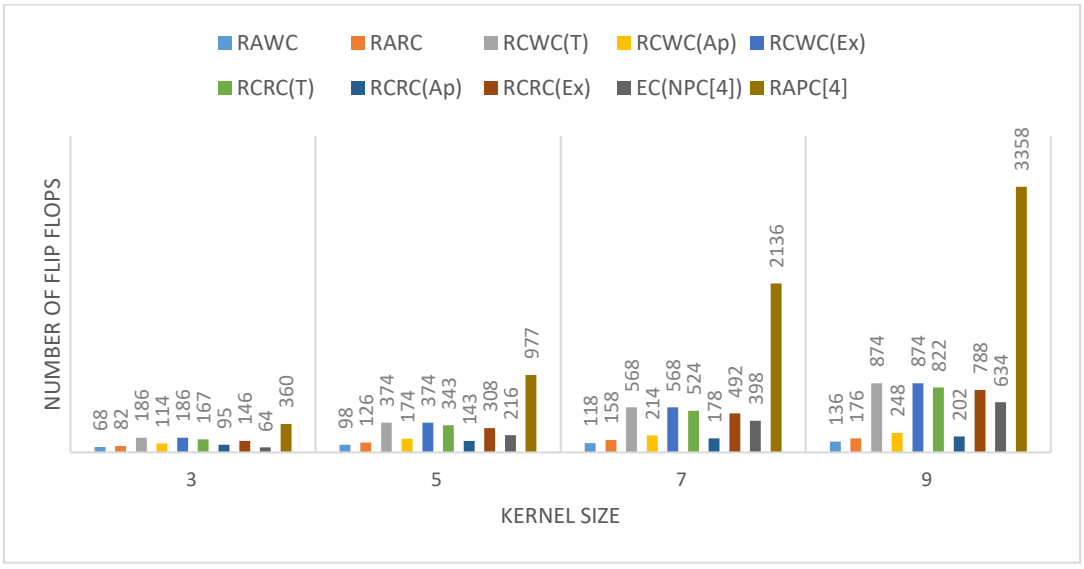

Figure 17. Number of FFs among various kernel sizes.

To show the superiority of the proposed design, we have also compared it with other related 2D convolvers. As shown in Table 1, the pixel access rate, and resource utilization of the convolvers proposed in [5],[6], and [19] are compared with approximate and combined convolvers. In this discussion, the FPGA devices are altered to make a fair comparison with other designs. Based on this table, the resource utilization of the approximate convolvers is improved significantly as shown in the total resource utilization column of Table 1. In the proposed combined convolvers, the total number of LUTs, FFs, and DPSs is less than the convolvers proposed in [5]. Also, in Table 1, the number of active resources is

mentioned in the approximate and exact processes which shows a significant reduction in the approximate execution for RCWC and RCRC. In comparison to [6], the proposed approximate convolvers utilize fewer LUTs and FFs; but, the combined designs use more LUTs while the number of FFs is reduced in RCRC. Finally, the pixel access rate and resource utilization of the proposed approximate and combined designs outperform in comparison to the proposed convolver in [19]. It's worth noting, the main characteristic of compared related works is that they conclude comparable key parameters in the same scenario; in addition, the proposed designs are implemented on FPGA-based hardware platforms.

Table 1: Comparison of pixel access rate, and resource utilization of different convolution structures (N/R denotes "not reported" in the corresponding reference)

Design	Kernel size	Device	Pixel Access rate(Pixel/cycle)	Total Resource Utilization (LUT+FF+DSP)	Active Resource Utilization (Ap) (LUT+FF+DSP)	Active Resource Utilization (Ex) (LUT+FF+DSP)
[5]	3	XC7V2000t	9	4750 + N/R + 0	All Active	All Active
RAWC	3	XC7V2000t	3	158 + 68 + 0	All Active	All Active
RARC	3	XC7V2000t	3	281 + 82 + 0	All Active	All Active
RCWC	3	XC7V2000t	3	1030 + 186 + 0	292 + 114 + 0	969 + 186 + 0
RCRC	3	XC7V2000t	3	1160 + 167 + 0	425 + 95 + 0	974 + 146 + 0
[6]	3	XC6SLX16	9	209 + 135 + N/R	All Active	All Active
RAWC	3	XC6SLX16	3	93 + 52 + 1	All Active	All Active
RARC	3	XC6SLX16	3	63 + 38 + 3	All Active	All Active
RCWC	3	XC6SLX16	3	384 + 138 + 10	310 + 117 + 1	383 + 138 + 9
RCRC	3	XC6SLX16	3	404 + 105 + 12	325 + 87 + 3	402 + 98 + 9
[19]	5	XCV4LX25	25	4612 + 4978 + 20	All Active	All Active
RAWC	5	XCV4LX25	5	253 + 75 + 1	All Active	All Active
RARC	5	XCV4LX25	5	179 + 47 + 6	All Active	All Active
RCWC	5	XCV4LX25	5	885 + 294 + 26	532 + 176 + 1	1048 + 295 + 25
RCRC	5	XCV4LX25	5	847 + 228 + 30	533 + 123 + 5	849 + 228 + 25

5.2 *Error Rate Evaluation*

The final part of the evaluation is conducting convolution operation on a sample image to compare the degradation in accuracy while using the proposed approximate and combined convolvers. All of the experimental analyses in this section are conducted in Matlab which is a popular mathematical and scientific software. In this evaluation, two popular benchmarks in image processing named Cameraman and Lena are selected. For the comparison of the error rate and filtered image quality of exact and approximate convolvers in computing convolution results, the parameters such as Root Mean Square Error (RMSE), Peak Signal to Noise Ratio (PSNR), and the number of approximate and exact convolvers in each scenario are measured. RMSE shows the error rate by checking the similarity of two images and when is closer to zero, the higher similarity is the result. On the other hand, PSNR compares the image quality between the reconstructed image and the original one. As it's obvious, an increase in RMSE is equivalent to a decrease in PSNR [24].

In this section two filters, i.e. Sobel, and Gaussian are selected to evaluate the proposed convolvers. In image processing applications, the Sobel filter in directions x and y will be used for detecting a wide range of edges in a sample image while the Gaussian filter is a method for blurring images by decreasing the amount of intensity variation between neighboring pixels [1]. In convolution computation, we have used both exact and approximate convolvers for the trade-off between accuracy and performance. As mentioned earlier, a pre-determined threshold is considered to decide the amount of using approximate convolver which prevents high degradation in accuracy. In the evaluation scenarios, the threshold is set from 0 to 10. Based on Equations (11) and (12), when the threshold value grows, the number of approximate convolvers will be increased and vice versa.

5.2.1 *Error rate evaluation with Sobel filter*

Figure 18 shows the Sobel filter coefficients in directions x and y. To evaluate the proposed designs, two general scenarios are selected. First, the experimental analyses are conducted on the approximate convolver by using Window-, Row-, and Column-based convolutional computations. Second, the evaluation is performed on the combined convolvers under the same methods of the convolutional computations.

-1	0	1
-2	0	2
-1	0	1

(a) Sobel filter in direction x

-1	-2	-1
0	0	0
1	2	1

(b) Sobel filter in direction y

Figure 18. Sobel filter in directions x and y.

Figures 19 and 20 demonstrate different filtered images with the Sobel in directions x and y kernel by using approximate convolver under the Cameraman benchmark. As shown in these figures, the Column- and Row-based strategies are superior for Sobel kernels in directions x and y respectively. Since the Sobel filter in direction x finds vertical edges and the Column-based method averages the column of input pixels, the error rate is bounded. In contrast to the Sobel filter in direction x, for y direction, the Row-based method is superior because the horizontal edges are found. Finally, since the Window-based method computes the average of input pixels in a window, the error rate for both filters is increased.

The RMSE, PSNR, and the number of approximate and exact convolvers for the Sobel filter in direction x under different threshold values are demonstrated in Figure 21. When the threshold is set to 0, because all exploited convolvers are exact, the RMSE value is zero and PSNR is ∞ accordingly. As the threshold value goes up, in all approximate convolutional computation techniques, the number of approximate convolvers increases, and accordingly the number of exact convolvers decreases. Because in this scenario the Sobel filter in direction x is used, the Column-based method is superior among all other strategies which is proven by the PSNR and RMSE values. On the other hand, the Window-based method has negligible superiority in comparison to the Row-based strategy by using fewer approximate convolvers. In Figure 22, to compare two strategies in a fair situation, the number of approximate convolvers is equalized. After that, the PSNR value of the Row-based method is more than the Window-based strategy and the RMSE value is decreased accordingly. To show the effect of the proposed strategies on a sample image, Figures 23, 24, and 25 are depicted under 0, 5, and 10 threshold values respectively. As can be seen, by increasing the threshold value, the Column-based method has minimum degradation in accuracy in edge detection. But, the error rate of Row-based and Window-based strategies is increased due to decreasing the PSNR under 30 dB based on Figure 22.

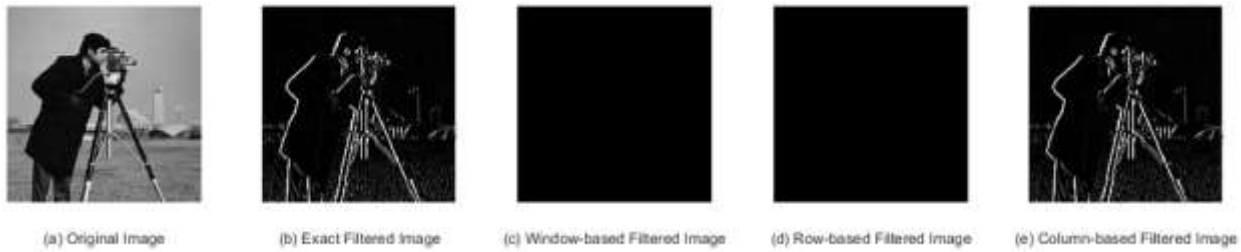

Figure 19. Different filtered images with the Sobel in direction x kernel by using approximate convolver under Cameraman Benchmark.

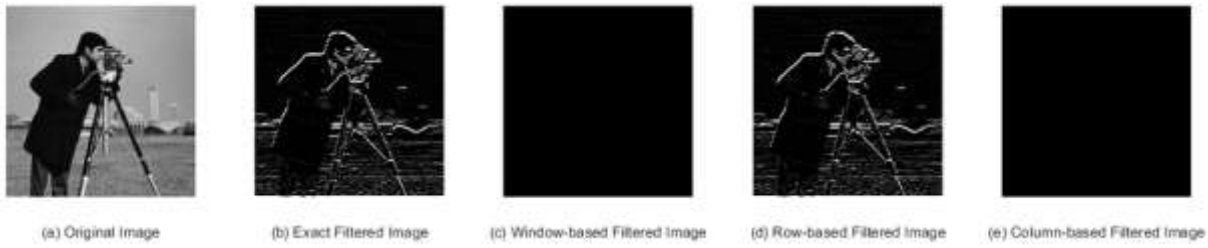

Figure 20. Different filtered images with the Sobel in direction y kernel by using approximate convolver under Cameraman Benchmark.

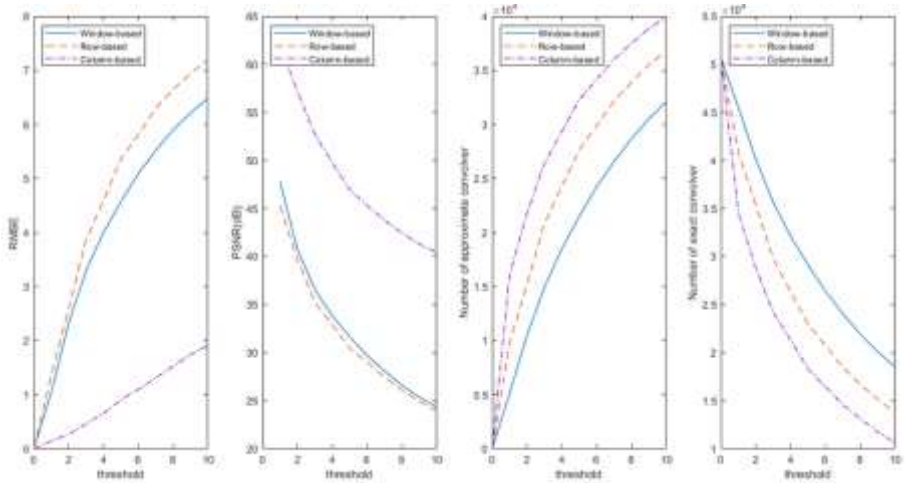

Figure 21. Comparison between RMSE PSNR, and the number of approximate and exact convolvers with the Sobel in direction x kernel by using combined approximate and exact convolvers under Lena Benchmark.

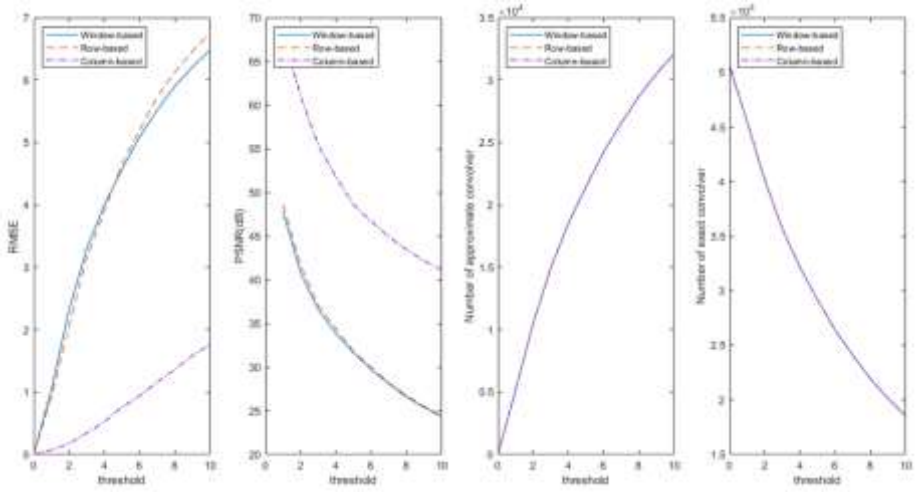

Figure 22. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction x kernel using combined approximate and exact convolvers under Lena Benchmark.

Figure 23. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction x kernel using combined approximate and exact convolvers under Lena Benchmark with threshold = 0.

Figure 24. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction x kernel using combined approximate and exact convolvers under Lena Benchmark with threshold = 5.

Figure 25. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction x kernel using combined approximate and exact convolvers under Lena Benchmark with threshold = 10.

For the comparison of two convolvers under the Sobel filter in direction y, Figures 26 and 27 are shown. Based on the above-mentioned reasons, it's predicted that the Row-based method outperforms in comparison to other strategies. In this scenario, the PSNR for the Row-based method is from 65 to 40 under different threshold values. Also, the RMSE value is significantly less than other related methods. In this scenario, the Window-based method is superior in comparison to the Column-based method by using a fewer number of approximate convolvers. In Figure 27, we have also equalized the number of approximate convolvers among different strategies which the PSNR and RMSE for the Column-based method are improved. For the visual comparison of the proposed convolvers on a sample image, Figures 28, 29, and 30 are drawn under 0, 5, and 10 threshold values respectively which show the Row-based method misses fewer edges in comparison to other methods.

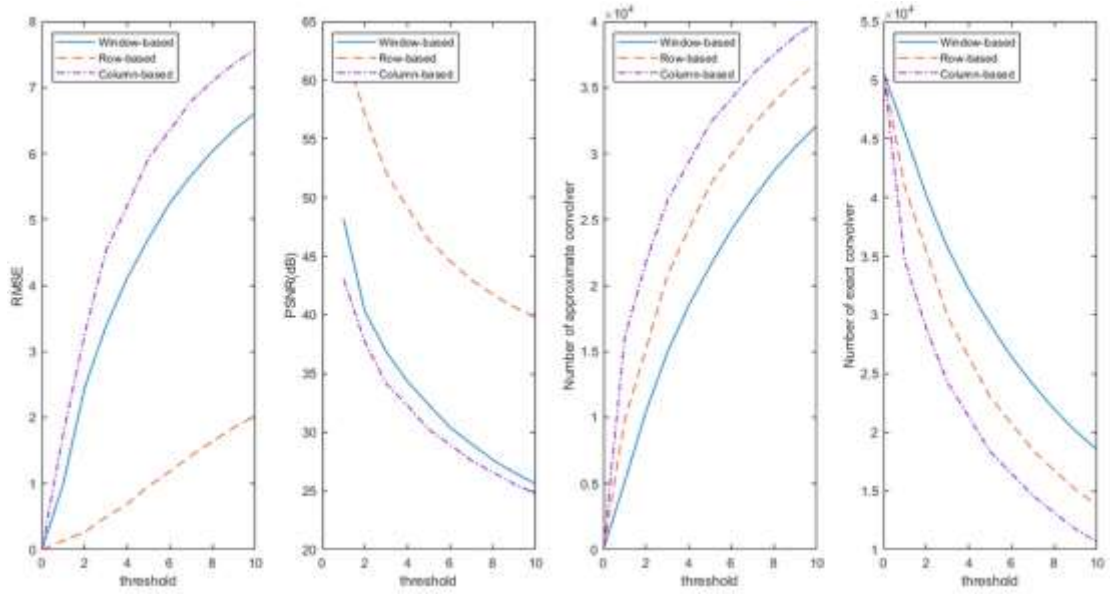

Figure 26. Comparison between RMSE, PSNR, and the number of approximate and exact convolvers with the Sobel in direction y kernel by using combined approximate and exact convolvers under Lena Benchmark.

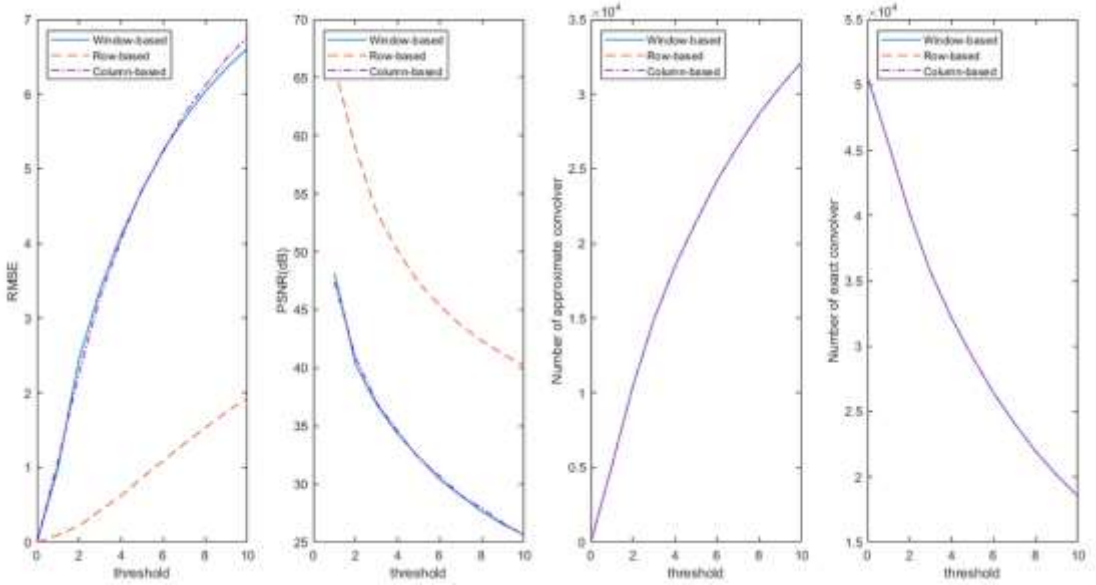

Figure 27. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction y kernel using combined approximate and exact convolvers under Lena Benchmark.

Figure 28. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction y kernel using combined approximate and exact convolvers under Lena Benchmark with threshold = 0.

Figure 29. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction y kernel using combined approximate and exact convolvers under Lena Benchmark with threshold = 5.

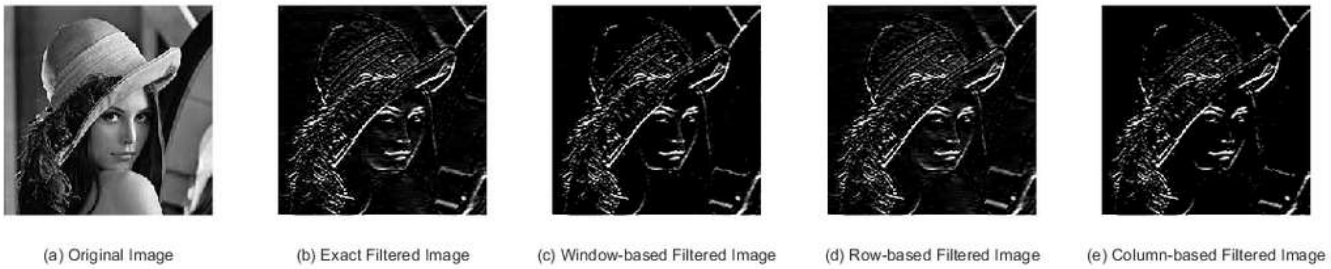

Figure 30. Comparison between RMSE and PSNR with the same number of approximate and exact convolvers by the Sobel in direction y kernel using combined approximate and exact convolvers under Lena Benchmark with threshold = 10.

5.2.2 *Error rate evaluation with Gaussian filter*

To show the error rate of the proposed approximate and combined convolvers for various kernel sizes, the Gaussian filter is selected. Figure 31 shows the Gaussian filter coefficients for 3 × 3, 5 × 5, and 7 × 7 kernel sizes. The Gaussian filter performs a blurring operation on a sample image.

1/16

1	2	1
2	4	2
1	2	1

(a)

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

(b)

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

(c)

Figure 31. Gaussian filters with (a) 3 × 3 (b) 5 × 5 (c) 7 × 7 kernel sizes.

In the Gaussian filter, as predicted, the error rate is less than the Sobel filter in the three methods generally. As can be seen in Figure 32, all three methods have PSNR more than 30 dB even with large usage of the approximate convolvers. Accordingly, the Window-based method has the largest error rate among others because of averaging in a window of pixels. In the Column- and Row-based methods, the error rate is dependent on the input image, and based on the selected benchmark one of them will be superior. To show the effect of increasing the kernel size on RMSE and PSNR value, Figures 33 and 34 are drawn with 5 × 5 and 7 × 7 kernel sizes respectively. As evident from these figures, the error rate is increased; but, the PSNR value is more than 30 dB which is acceptable for many applications. To make a fair comparison, we have also equalized the number of approximate and exact convolvers in Figures 35, 36, 37, and 38. In addition, because the top amount of error rate occurs when the number of exact

convolvers is close the zero, the threshold value is increased to 100. Also, this scenario is for testing the error rate when the approximate convolver is used and the kernel sizes are similar to the previous scenario. As can be seen, the PSNR value for 3×3 and 5×5 for all three methods is still more than 30 dB; but, for 7×7 kernel size is decreased to less than 30 dB. In Figure 36, the input image is changed to the Cameraman benchmark to show the superiority of the Row-based method in comparison to the Column-based strategy in a real example. As can be seen in this figure, by changing the input image the PSNR value for the Row-based method is more than other related methods.

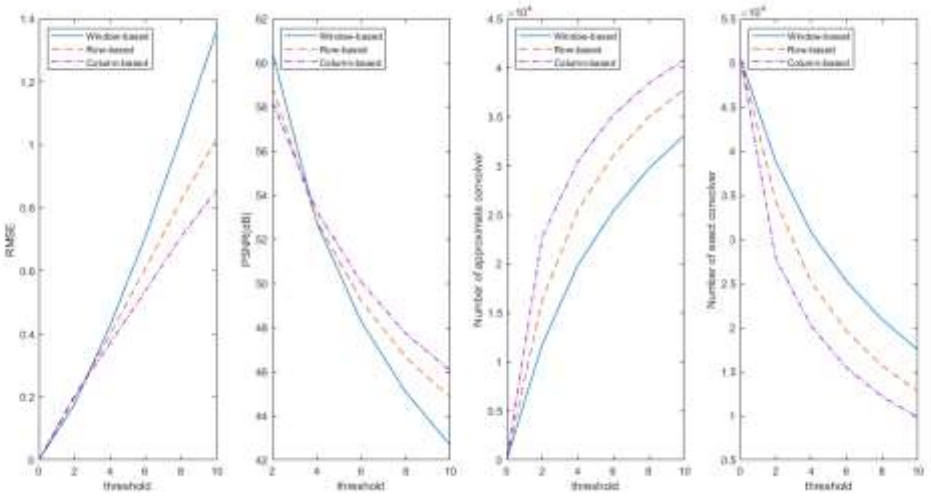

Figure 32. Comparison between RMSE, PSNR, and the number of approximate and exact convolvers with the Gaussian kernel by using combined approximate and exact convolvers under Lena Benchmark for 3×3 kernel size.

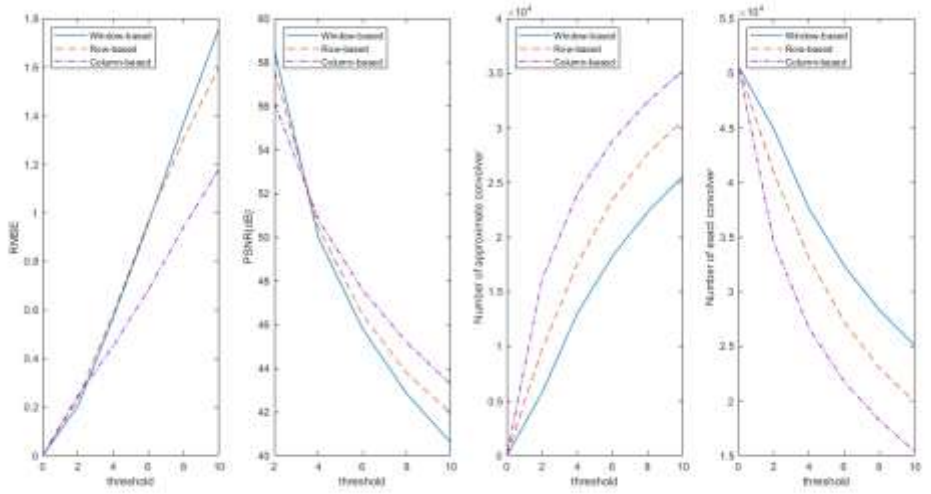

Figure 33. Comparison between RMSE, PSNR, and the number of approximate and exact convolvers with the Gaussian kernel by using combined approximate and exact convolvers under Lena Benchmark for 5×5 kernel size.

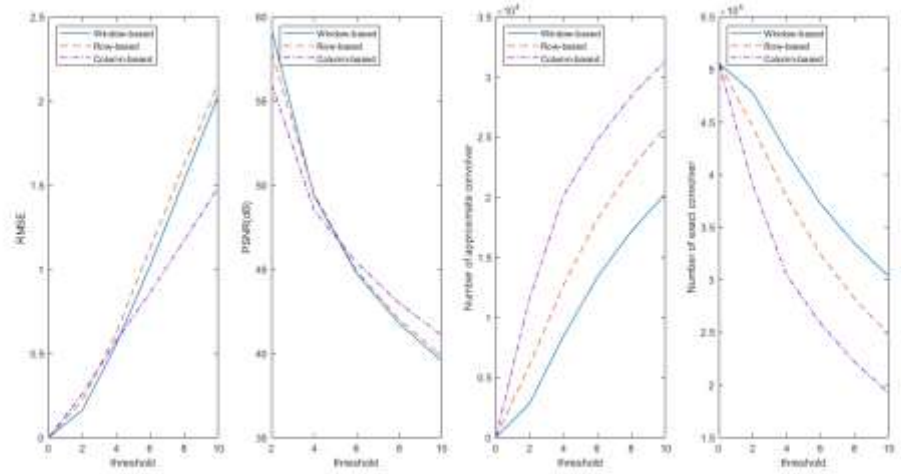

Figure 34. Comparison between RMSE, PSNR, and the number of approximate and exact convolvers with the Gaussian kernel by using combined approximate and exact convolvers under Lena Benchmark for 7 × 7 kernel size.

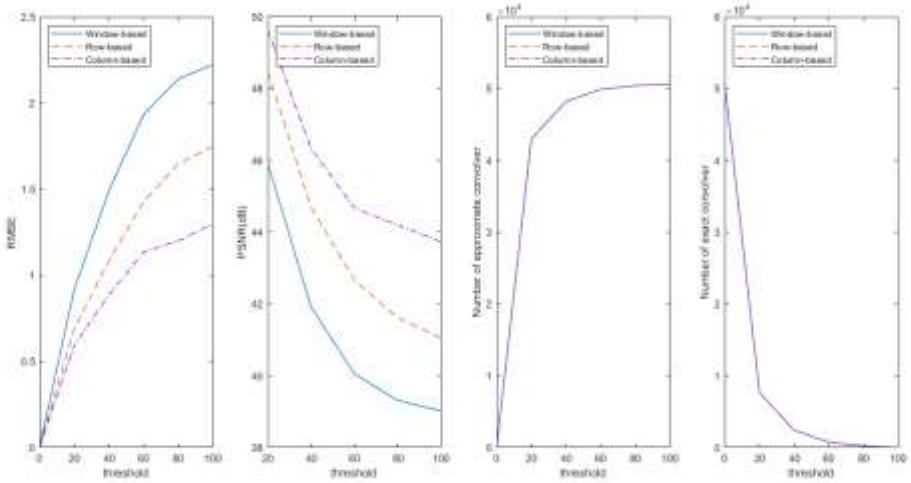

Figure 35. Comparison between RMSE, PSNR, and with the same number of approximate and exact convolvers with the Gaussian kernel by using combined approximate and exact convolvers under Lena Benchmark for 3 × 3 kernel size.

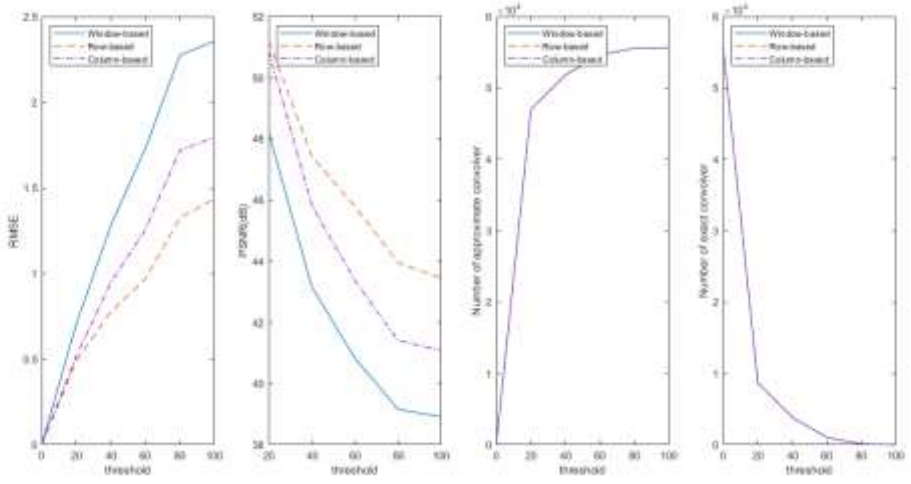

Figure 36. Comparison between RMSE, PSNR, and with the same number of approximate and exact convolvers with the Gaussian kernel by using combined approximate and exact convolvers under Cameraman Benchmark for 3 × 3 kernel size.

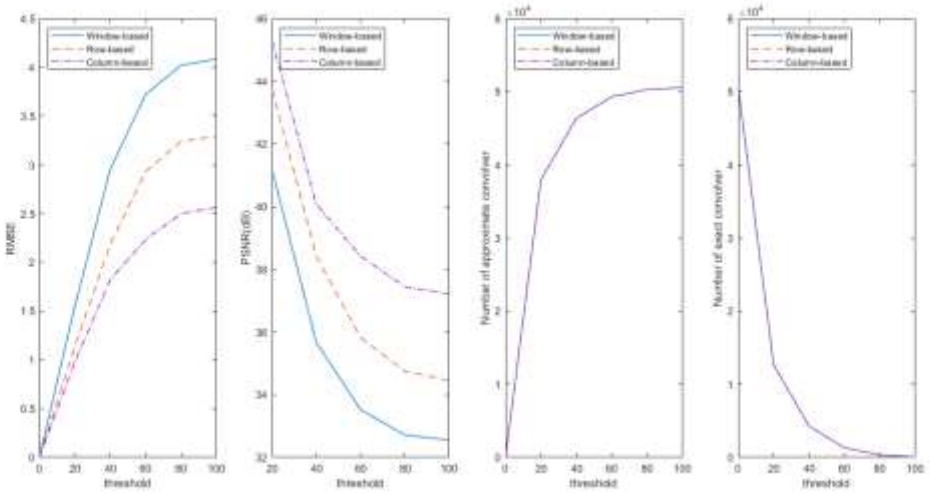

Figure 37. Comparison between RMSE, PSNR, and the same number of approximate and exact convolvers with the Gaussian kernel by using combined approximate and exact convolvers under Lena Benchmark for 5 × 5 kernel size.

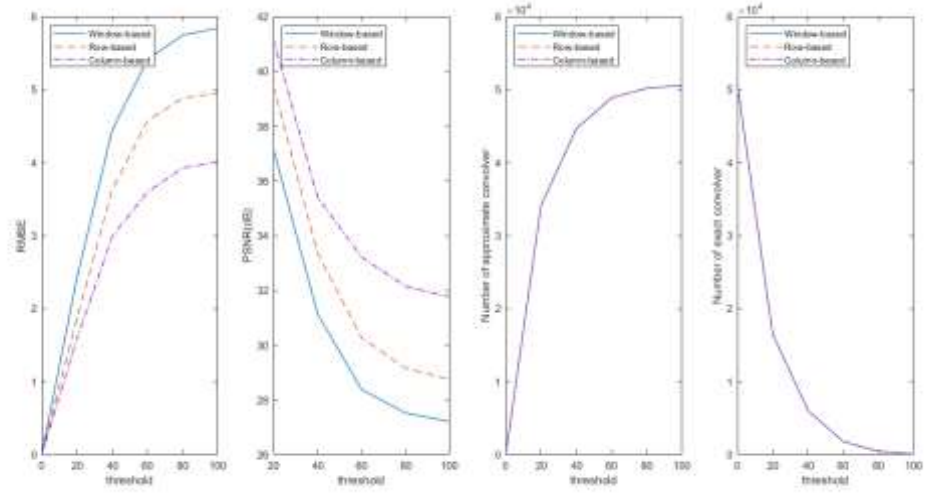

Figure 38. Comparison between RMSE, PSNR, and the same number of approximate and exact convolvers with the Gaussian kernel by using combined approximate and exact convolvers under Lena Benchmark for 7 × 7 kernel size.

As a final note in this section, the convolved image evaluation shows that exploiting approximate convolver in special places of the image, not only didn't reduce the filtered image accuracy significantly but also the resource utilization and power consumption can be improved. Also, we have evaluated our proposed combined convolver under different scenarios by using a special relation to decide whether exact or approximate convolver is suitable for current computation which can be extended to other related scenarios. Finally, it's up to the requirement for the applications to decide between the accuracy and performance of the convolvers.

6. CONCLUSION

In this paper, approximate and combined 2D convolvers with optimized power consumption and low pixel access rate are introduced. The presented designs can be exploited in real-time image processing applications. In the approximate convolvers, the resource utilization and power consumption outperform their counterparts with a competitive critical path delay in comparison to EC. In the combined convolvers,

the number of active resources in the approximate path is improved with a negligible reduction in clock frequency. In addition, we have taken the opportunity to the applications to choose different levels of accuracy. Different computational analyses and experimental evaluations are performed on the approximate and combined convolvers using various FPGA-based hardware platforms. The evaluated results on hardware and error rate reveal that with a negligible missing of accuracy, the power consumption is improved up to 34% and 20% in approximate and combined convolvers respectively for 3 × 3 kernel size. For future work, CNN needs a high-performance processing engine. The basic module of the CNNs is the convolution unit. By exploiting the proposed convolvers and developing a CNN accelerator the on-chip power consumption can be reduced significantly.

Declarations

Ethical Approval

Not applicable

Competing interests

Not applicable

Authors' contributions

All authors have contributed to the writing and reviewing of the manuscript text and the level of contribution is according to the name list order maintained in the paper.

Funding

This research and publication was supported by grant No. RD-51-9911-0039 from the R&D Center of Mobile Telecommunication Company of Iran (MCI) for advancing information and communications technologies.

Availability of data and materials

All the materials including VHDL codes, experimental road map, data collected are available and ready to be provided as requested by the reviewers.

7. REFERENCES

[1] B. R. Masters, R. C. Gonzalez, and R. Woods, “Digital image processing,” Journal of biomedical optics, vol. 14, no. 2, p. 029901, 2009.

[2]Y. Zhao, M. Wang, G. Yang and J. C. W. Chan, “FOV Expansion of Bioinspired Multiband Polarimetric Imagers With Convolutional Neural Networks,” IEEE Photonics Journal, vol. 10, no. 1, pp. 1–14, Feb 2018.

[3] Q. Xu, T. Mytkowicz and N. S. Kim, "Approximate Computing: A Survey," in *IEEE Design & Test*, vol. 33, no. 1, pp. 8-22, Feb. 2016.

[4] M. Kalbasi and H. Nikmehr, "A Fine-Grained Pipelined 2-D Convolver for High-Performance Applications," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 1, pp. 146-150, Jan. 2019.

[5] G. D. Licciardo, C. Cappetta, and L. D. Benedetto, “FPGA optimization of convolution-based 2d filtering processor for image processing,” in 2016 8th Computer Science and Electronic Engineering (CEEC), Sept 2016, pp. 180–185.

[6] F. Cabello, J. Len, Y. Iano, and R. Arthur, “Implementation of a fixed-point 2d Gaussian filter for image processing based on FPGA,” in 2015 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Sept 2015, pp. 28–33.

[7] Chen, Ke, Fabrizio Lombardi, and Jie Han. "Design and analysis of an approximate 2D convolver." *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2016.

[8] Chen, Ke, et al. "Efficient implementations of reduced precision redundancy (RPR) multiply and accumulate (MAC)." *IEEE Transactions on Computers* 68.5 (2018): 784-790.
[9] Sborz, Guilherme, Felipe Viel, and Cesar Zeferino. "Architectural Exploration of an FPGA-based Hardware Accelerator for the Gaussian Filter using Approximate Computing." *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*. SBC, 2020.
[10] R. Menaka, S. Janarthanan, and K. Deeba, "FPGA implementation of low power and high speed image edge detection algorithm," Microprocess Microsyst, pp. 103 053–1–103 053–7, 2020.
[11] D. Sangeetha and P. Deepa, "Fpga implementation of cost-effective robust Canny edge detection algorithm," J Real Time Image Process, vol. 16, no. 4, pp. 957–970, 2019.
[12] H. Zhang, M. Xia, and G. Hu, "A multi-window partial buffering scheme for FPGA-based 2-D convolvers," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 54, no. 2, pp. 200–204, Feb 2007.
[13] B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-D convolvers for fast digital signal processing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 7, no. 3, pp. 299–308, Sept 1999.
[14] M. H. Sunwoo and S. K. Oh, "A multiplier-less 2-D convolver chip for real-time image processing," Journal of VLSI signal processing systems for signal, image and video technology, vol. 38, no. 1, pp. 63–71, Aug 2004. [Online]. Available: https://doi.org/10.1023/B:VLSI.0000028534.35761.a8
[15] F. J. Toledo-Moreo, J. J. Martnez-Alvarez, J. Garrigs-Guerrero, and J. M. Ferrndez-Vicente, "FPGA-based architecture for the real-time computation of 2-d convolution with large kernel size," Journal of Systems Architecture, vol. 58, no. 8, pp. 277 – 285, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762112000562
[16] M. Z. Zhang, H. T. Ngo, and V. K. Asari, "Multiplier-less VLSI architecture for real-time computation of multi-dimensional convolution," Microprocessors and Microsystems, vol. 31, no. 1, pp. 25 – 37, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933106001104
[17] M. Z. Zhang and V. K. Asari, "An efficient multiplier-less architecture for 2-D convolution with quadrant symmetric kernels," Integration, the VLSI Journal, vol. 40, no. 4, pp. 490 – 502, 2007, system-Level Interconnect Prediction. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926006000666
[18] Z. B. Ma, Y. Yang, Y. X. Liu, and A. A. Bharath, "Recurrently decomposable 2-D convolvers for FPGA-based digital image processing,"IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 63, no. 10, pp. 979–983, Oct 2016.
[19] F. Fons, M. Fons, and E. Cant, "Run-time self-reconfigurable 2D convolver for adaptive image processing," Microelectronics Journal, vol. 42, no. 1, pp. 204 – 217, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002626921000162X
[20] D. K. Yadav, A. K. Gupta, and A. K. Mishra, "A fast and area efficient 2-d convolver for real time image processing," in TENCON 2008 – 2008 IEEE Region 10 Conference, Nov 2008, pp. 1–6.
[21] Artix-7 FPGAs Data Sheet, Xilinx, Inc., 2017, v1.22.
[22] M. Kalbasi and H. Nikmehr, "A Classified and Comparative Study of 2-D Convolvers," *2020 International Conference on Machine Vision and Image Processing (MVIP)*, 2020, pp. 1-5, doi: 10.1109/MVIP49855.2020.9116874.
[23] Dehghani, A., Kavari, A., Kalbasi, M. *et al.* A new approach for design of an efficient FPGA-based reconfigurable convolver for image processing. *J Supercomput* (2021). https://doi.org/10.1007/s11227-021-03963-6.
[24] D. Salomon, Data compression: the complete reference. Springer, 2004.