The CSI Journal on
Computer Science and Engineering
Vol. 19, No. 1, 2025
Pages 17-24
doi.org/10.22034/jcse.2023.182036
Regular Paper

Approximate Carry Save Adders Lead to an Efficient Implementation of Low-latency Two-Dimensional Gaussian Smoothing Filter

Hadise Ramezani[1] Majid Mohammadi[2] Amir Sabbagh Molahosseini[1]

[1]Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran
[2]Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran

Abstract

One of the popular filters in image processing is the two-dimensional Gaussian smoothing filter (2D-GSF). An efficient implementation and design of 2D-GSF in real-time processing are significantly important since requiring high computing resources. A modern technique capable of reducing the 2D Gaussian filter design's latency and increasing the performance with insignificant computing overhead on field programmable gate arrays (FPGA) is called approximate computing. The present research's objective is presenting a low-latency Gaussian filter architecture on FPGA to be applied in real-time processing. Therefore, approximate and accurate carry-save adders (CSAs) were employed in Gaussian filters based on the adder tree. The implementation and simulation results prove the latency reduction in 3x3 2D-GSF architectures up to 45% and 22% by applying proposed approximate CSAs and accurate CSAs, respectively, compared to the available Gaussian filters with an adder tree structure. Output image quality decreases by an inconsiderable rate of 1.2% in approximate design.

Keywords: 2D Gaussian smoothing filter, Approximate computing, FPGA Implementation, Carry save adder**,** two-dimensional Gaussian smoothing filter (2D-GSF)

1. Introduction

Image smoothing is of applications of Gaussian filters. Image processing applications, including the detection of edge or line in an image, are of important functions of the smoothing filter. The function of an edge detection operator is determined as its capability of locating the very close edge to its true position in a noisy image [1]. A type of convolution filter is named 2D-GSFs, in which a convolution is performed between kernel and image matrices. The computation cost of convolution operations is considerably high. Hence, the design of low-latency 2D-GSD is of great importance in the majority of applications in the field of image processing that require the generation of results related to real-time circumstances. The font size must be 10 for the main text. The components of real-time digital image processing are implemented by applying FPGAs, which are proven to be twice faster and perform better than Digital Signal Processors (DSP) concerning the consumed resources [2].

A low-latency digital Gaussian smoothing filter is designed and implemented on an FPGA in the present article. For this purpose, a Gaussian filter architecture is proposed on the basis of approximate computing possessing better design matrices compared to the present filters while trying to keep the output quality matrices. Three tricks have been used to obtain a high-performance architecture as follows:

1- In SPAA Gaussian filter architecture [3], the techniques named rounding the filter coefficients to power-of-two and NPA Approximate Computing have been applied to enhance the latency and power consumption matrices. The conversion of the adder-tree structure into CSA has been done to reduce the latency.

2- The highest latency in CSA is strongly associated with its n-bit adder; an approximate adder in its structure has been applied for the purpose of more reduction of latency, and subsequently, in SPAA Gaussian filter architecture, this approximate CSA has been replaced with adder-tree.
3- Approximate adders, along with a new one, were applied at the time of designing the proposed approximate CSPA, which was the basis for designing the approximate adder that was modified for latency reduction.
The rest of the paper is organized as follows: In Section 2, the works in which approximate computing was applied in Gaussian filters and some of the key approximate adders employed in Gaussian filter architecture are examined. In Section 3, detailed information about presented CSA-based Gaussian filter architecture and also approximate CSA are proposed. Section 4 addresses the simulation and investigation process of suggested Gaussian filter architecture and reports the design criteria of Gaussian filter architecture such as latency, power consumption, area, and also the simulation results of output quality. Finally, the conclusion is presented in the last section.

2. Related Work

This filter is a particular weighted averaging filter applied in numerous applications in the field of image processing, including image segmentation, edge detection, and image blurring. The performance of such type of applications is considerably influenced by the Gaussian filter's performance. Approximate computing approaches applied to raise the Gaussian filter's performance are evaluated in this section. This approximate computing involves the application of simple types of the approximate kernel, approximate adders, and Gaussian filter architecture.

2.1. Approximate Gaussian Smoothing Filters

Extensive works were carried out on the two-dimensional approximate Gaussian filters design. In [4], a 2D-GSF architecture on the basis of a 5x5 window is proposed, in which the conversion of each filter kernel coefficient to an expression composed of the sum of power-of-two is performed; therefore, a slight modification of the standard deviation (σ) is observed. A presented algorithm was used to make the Power-of-two approximation.
The kernel coefficients of Gaussian filter are presented as floating point; in hardware implementation, raises its area and power consumption. In [5], rounding the filter coefficients and fixed-point data were applied for the purpose of reduction of runtime and simplification of computation. Although the area of implementation of this approach is still considerable.
A two-dimensional approximate Gaussian filter architecture has been suggested in [3], in which a couple of approximate approaches were applied to enhance Speed-Power-Area-Accuracy (SPAA) matrices: rounding of kernel coefficients and Nearest Pixel Approximation (NPA). Figure 1 illustrates the 5x5 as well as 3x3 architectures of this Gaussian filter. The coefficients on the kernel matrix's main diameter are only applied in the first approximation stage, instead of utilizing all kernel coefficients. As the image involves a high connection between nearby pixels, their values are nearly identical, except on the edges, with a maximum difference in the range of 1 to 4 percent. Those neighbour pixels can be almost identical, and as an alternative for N × N pixels, N pixels are processed accordingly. NPA approach displayed a discrepancy of 4 percent in 92 percent and 85 percent of its pixels, respectively in 5x5 and 3x3 2D-GSFs.

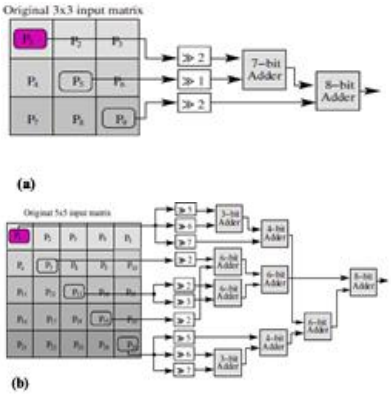

Figure1: NPA 2D-GSF Architecture; (a) 3x3 2D-GSF; (b) 5x5 2D-GSF [3]

The kernel coefficients of the Gaussian filter, presented as floating points, are converted to the nearest power of 2 fixed-point numbers in the second approximation stage. In this regard, multiplication operators may be executed by shift and sum operators in convolution computation, thus growing power usage and efficiency. An approximate Gaussian filter is presented in [6], approximate adders are applied to minimize the latency, area of the filter architecture in this filter, and power consumption. Approximate ETA-I adders with various levels of accuracy are used in the Gaussian filter presented in various sections of the filter architecture. Generally speaking, eight approximate adders are employed in the architecture presented, half of which are more precise. Correspondingly, both filter performance and output quality can be set at the same time.
The kernel coefficients of the Gaussian filter are not similar in terms of significance; from the center to the boundaries, their value decreases [7]. The importance of the filter coefficients and the boundaries is indicated in Figure 2. As an illustration, the maximum and minimum significance are related to boundaries B1 and B4, respectively. Therefore, a reconfigurable architecture has been given in [8] that is capable of being applied to produce output from one or more kernel boundaries. When the output quality is not important in one application, the kernel coefficients existed at boundary B1 are only applied for the purpose of energy saving, and in the case of maximum output quality, it is possible to use the existed coefficients at all boundaries.

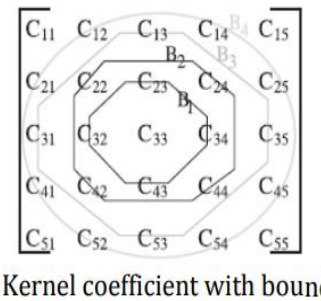

Figure2: 5x5 Gaussian Filter and Boundaries

2.2. Approximate Adders

For most cases, in an n-bit adder, a shorter length of carry chain compared to n is observed. Therefore, [9] proposed a speculative adder-based almost correct adder in [10]. An 8-bit almost correct adder with blocks of k = 4-bits is indicated in Figure 3; the latency is decreased to a 4-bit adder. On average, the adder's speed is 1.5 times higher compared to the speed of the Carry Looked Ahead Adder (CLA).

Error tolerant adders (ETA-II) are given in [11]. The n-bit error-tolerant adders, as seen in Figure 4, is separated into M blocks, each of which comprises two units (circuitry) - the Sum and Carry Generators. The input carry (carry-in) of the sum generator of every block is produced using the carry generator of the previous block. Hence, the critical path of the error-tolerant adders only comprises of carry and sum generators, every single of which is a bit of n/M. Other types of these adders are provided in [12], [13].

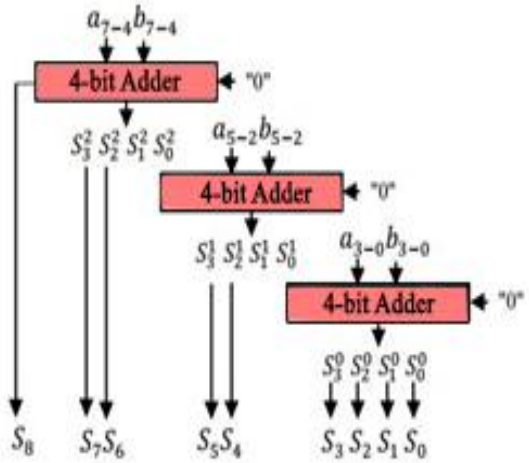

Figure3: Almost Correct Adder (ACA)

Researchers of [15] introduced an accuracy-configurable adder; this adder was defined as a GDA (a gracefully degrading accuracy-configurable adder). Using the control signals in the adder, the approximate or accurate carry-in is capable of being chosen, and the required accuracy may be adjusted accordingly. The latency of the GDA relies on the accuracy of the control signals and, thus, as seen in Figure 5, the mode of carry propagation.

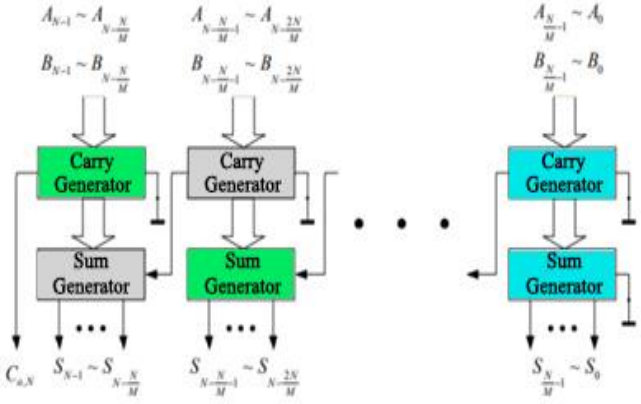

Figure4: Error Tolerant Adder (ETA-II)

The carry speculative adder (CSPA) is demonstrated in Figure 6 [16], which is analogous to the CSPA adders, except in each block, the CSPA is composed of two internal carry generators (carry with an input of 1 as well as carry with an input of 0), one carry predictor, and one sum generator. The output of the ith-block of carry predictor is applied to pick the output of one of the internal carry generators in the (i + 1) th block. As input to the sum generator of the ith block, the selected output is provided. K<x bit is only employed in the carry predictor, and thus the overhead of the hardware, compared to SCSA, is minimized. It is noteworthy that in K<x, x denotes the size of every single block, and the k-bit of the block is applied for predicting the next block's carry.

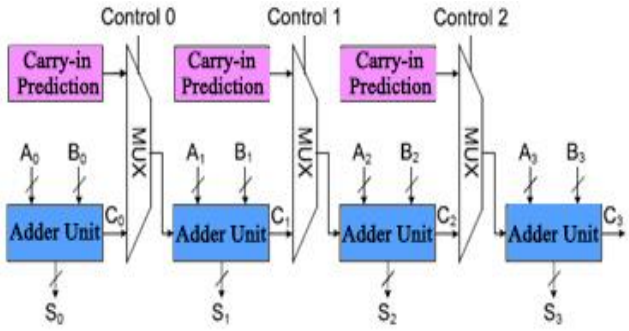

Figure5: Error Tolerant Adder (ETA-II)

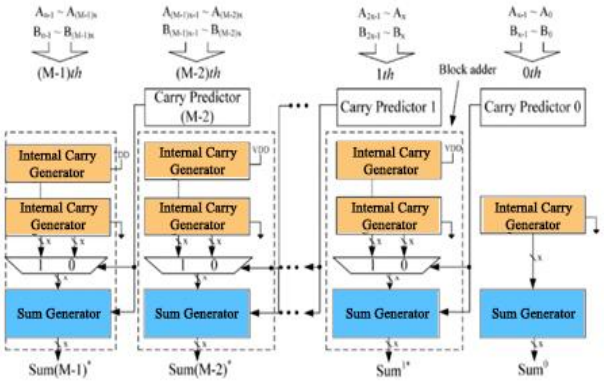

Figure6: Error Tolerant Adder (ETA-II)

The configuration of a GeAr (Generic Accuracy Configurable) adder [17] is performed by employing the parameters as follows: P: the number of projection bits in each sub-adder, k: the number of sub-adder units, R: the number of sub-adder bits, L: the sum of P and R for every single sub-adder. A GeAr (12, 4, 4, 2) is indicated in Figure 7. In GeAr, there are several configuration parameters that enable a considerably flexible architecture, and the balance between the quality of the output and the efficiency can be easily obtained.

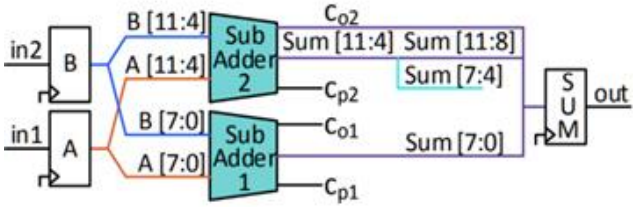

Figure7: Generic Accuracy Configurable Adder with k = 2, P = 4, R = 4, and N = 12 [17]

3.Approximate Gaussian Filter Architecture

Approximate carry-save adders are applied in an approximate Gaussian filter introduced in this section. A multi-operand adder called carry-save adder [18] can be applied to raise speed in circuits with structures as same as an adder tree structure.

According to Figure 1, the approximate Gaussian filter architecture demonstrates that SPAA matrices can be enhanced by a couple of methods in the design: rounding the kernel coefficients to the nearest power-of-two and also the NPA technique. It is indicated that a tree structure of adders is the components of two 3x3 and 5x5 2D-GSF architectures. The adders creating a critical path are in the longest branch in tree adders. As an instance, in the case that a branch includes two 8-bit adders, the branch's latency is equal to a 16-bit adder's latency.

For the reduction of the latency of adder-tree in two-dimensional Gaussian smoothing filter architecture, first, adder-tree was converted to carry-save adders, and then the approximate adders were replaced in carry-save adders' hardware.

The tree structure of two 3x3 two-dimensional Gaussian smoothing filter is converted into CSAs is shown in Figure 8. The nearest pixel approximation procedure has been applied in the first 2D-GSF, and the rounding of kernel coefficients to power-of-two has been employed in the second 2D-GSF.

The latencies of a 1-bit full adder plus an n-bit adder is equal to an n-bit CSA's latency. The latency of two-dimensional Gaussian smoothing filters indicated in modes (1), (2), (3), and (4) is calculated in Figure 8 as the following:

1-T= T (7-bit adder) + T (6-bit adder) + T (8-bit adder) + T (7-bit adder) = T (28-bit adder)

2- T= T (8-bit carry-save adders) + T (7-bit carry-save adders) = T (17-bit adder)

3- T= T (8-bit adder) + T (7-bit adder) = T (15-bit adder)

4- T= T (8-bit carry-save adders) = T (9-bit adder)

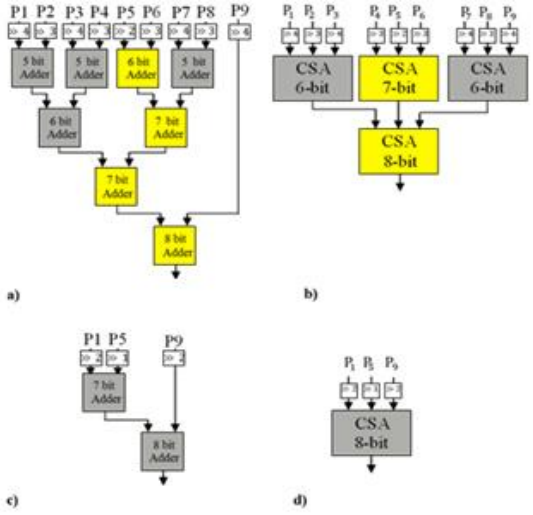

Figure8: Approximate and accurate 3x3 two-dimensional Gaussian smoothing filter conversion to CSA; (a) accurate 3x3 2D-GSF uses all nine kernel elements; (b) accurate 2D-GSF conversion to CSAs; (c) NPA 3x3 2D-GSF; (d) NPA 2D-GSF conversion to a CSA.

It is indicated in Figure 9 that the hardware is an n-bit CSA with an n-bit adder and an individual 1-bit n-adder. CSA adder's maximum latency is relevant to its n-bit adder. The replacement of an n-bit adder with a suggested approximate adder was done for reducing the latency of the CSA. The authors developed a CSPA-based approximate adder for performing this action, in which both accuracy and latency are enhanced.

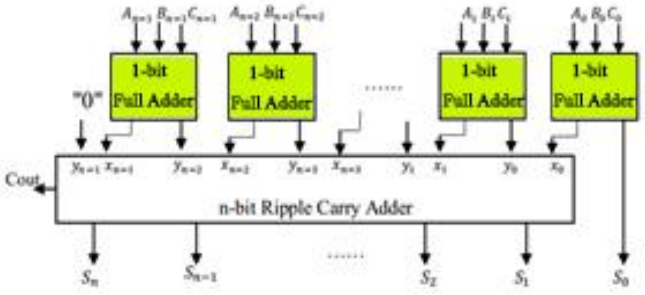

Figure9: An n-bit CSA Adder

An approximate adder, which was suggested, is indicated in Figure 10. The adder is identical to the CSPA adder, which has been changed to decrease latency and boost accuracy as the following:

1-The block size is considered as a 2-bit constant. For this purpose, the carry predictor circuits, error recovery of every single block, and carry generator are simplified, and their latency is minimized as a consequence.

In Figure 10, the group propagation outputs and group generate in every single block are produced by the carry speculator. Blocks' sizes are assumed to be 2-bit here; hence, the outputs, for instance, are achieved by the following relationships for block0:

$$P^0_{1-0} = (a_1 + b_1)(a_0 + b_0) \quad (1)$$

$$G^0_{1-0} = a_1 b_1 + a_0 b_0 (a_1 + b_1) \quad (2)$$

The group propagate (P) output is employed for detecting the correctness of the predicted carry. The group generate (G) output is applied as the predicted carry for the next block in every single block.

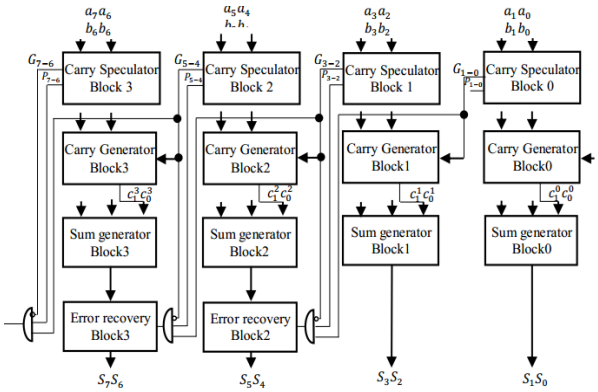

Figure10: Proposed MCSPA (Modified Carry Speculative Adder) Approximate Adder

2- The reduction of the blocks size is made to reduce the accuracy of the carry prediction, thus impacting the accuracy of the adder. For minimizing the severity of the errors, two modifications have been made for the CSPA adder error recovery. The error detection and recovery circuit in [16] is shown in Figure 11. In this circuit, for the detection of errors in every single block, the accurate and predicted carries are compared (by applying XOR gates); in the case of not being similar, that block's error signal (Err_block) is activated, and the next block's carry-in is defined as incorrect, which is predicted by this block. In the next block, the addition of one unit to the block's partial sum is performed by receiving the error signal. The modifications made involve the detection and recovery of errors. The overhead area is a part of the synchronized development of predicted and accurate carry; authors apply G and P group signals connected to every single block generated by the carry speculator for the purpose of detecting the error. If the group propagate (P^i) and carry-in (G^{i-1}) signal are equal to 1, carry-out (output carry) is equal to 0, and the carry predicted by the i block is considered as incorrect:

$$Err_{block_i} = G^{i-1}P^i\bar{G}^i \quad (3)$$

Hence, the implementation of the error detection circuit is made to be possible by applying an AND gate in every single block.

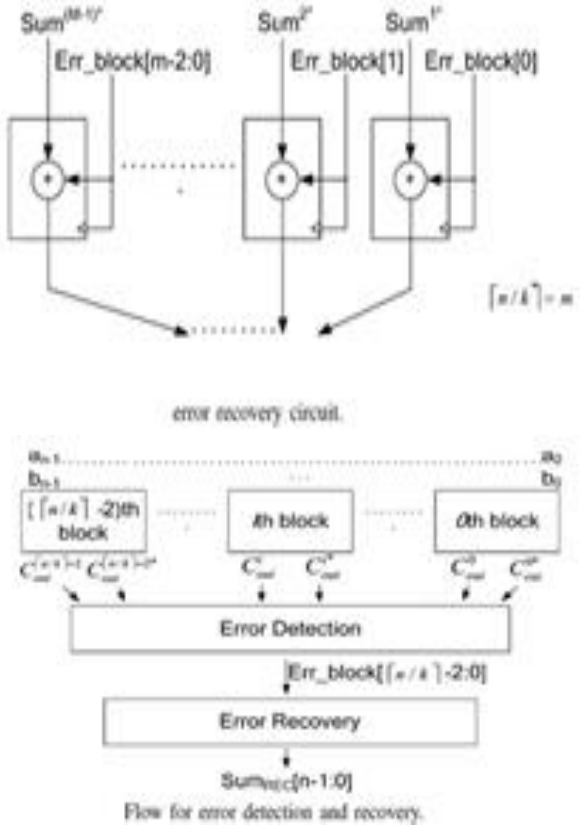

Figure11: The Stages of Error Detection and Recovery in [16]

In [16], the circuit related to error correction works as the following: the addition of one unit to the partial sum of the block is done if an error signal is received in the block. This approach results in correcting the error in most instances, but in some circumstances, that is shown in Figure 12, the error is not corrected, and even the sum overall error is raised. The error recovery circuit works appropriately in situations where the partial sum is among '00', '01', and '10' values. A problem arises whenever the partial sum is equivalent to "11" for the block in which the activation of the error signal occurs. In order to overcome the challenge, the consequence of the error propagation in all the next blocks must be removed, increasing latency and leading to a complexity in the error recovery circuit. If the error signal is enabled in our proposed method, the error correction is carried out solely in situations that the partial sum does not equal '11', as well as larger errors would be prevented.

i	4	3	2	1	0
P^i_{i-0}	0	1	1	1	1
G^i_{i-0}	0	0	0	1	1
A	01	10	11	01	10
B	00	01	00	11	11
C^i_{i-0}	00	00	11	11	00
Sum^i	01	11	00	01	01
Err	0	0	1	0	0
Sum^i_{cor}	01	00	00	01	01

Error recovery failed
Error magnitude without error recovery=64
Error magnitude with error recovery=256

i	4	3	2	1	0
P^i_{i-0}	0	0	1	1	1
G^i_{i-0}	0	0	0	1	1
A	01	01	11	01	10
B	00	01	00	11	11
C^i_{i-0}	00	10	11	11	00
Sum^i	01	10	00	01	01
Err	0	0	1	0	0
Sum^i_{cor}	01	11	00	01	01

Error recovery succeeded

Figure12: Succeeded, failed, or Error Recovery in [16].

4. Results and Discussion

The proposed Gaussian filters are examined in this section regarding the consumed resource, output quality, and latency. A Gaussian filter has been implemented by applying Xilinx ISE on the Virtex-7 family device 7VX330T for synthesizing and extracting the values of latency and used resources. By applying the MATLAB environment, the filter output quality analysis is performed. In Figure 13, stages of simulation and synthesis demonstrate the way that the proposed filter can be synthesized and simulated. The outcomes of existing approximate adders and Gaussian filter [3] were obtained in accordance with the Figure 13 procedure to achieve the same circumstances for comparison.

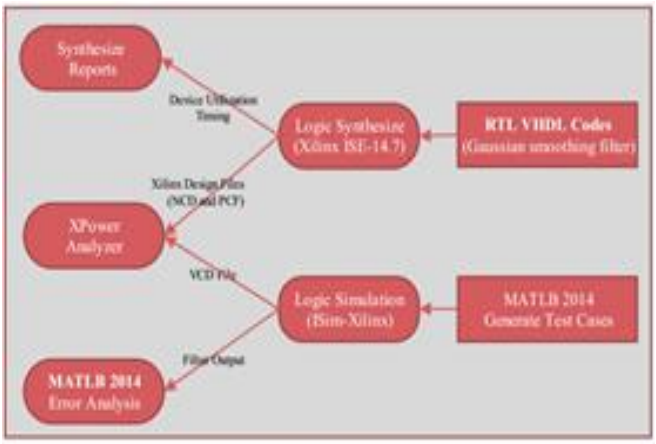

Figure13: The stages of synthesis and simulation.

4.1. Analysis of Gaussian Filter Output Quality

The noisy images of Boat and Lena are employed as inputs of Gaussian filters' inputs in [3], as well as the proposed 3x3 Gaussian filters are carried out in the simulations. Table 1 shows the simulation results.

Table 1: A comparison between the output quality of the images of the Gaussian filter architecture in [3] and the proposed Gaussian filter.

	SSIM		PSNR	
Noisy Image	0.391	0.315	20.68	20.32
SPAA-CSA-MCSPA-GSF	0.393	0.492	21.72	23.42
SPAA-CSA-CSPA-GSF	0.298	0.421	18.53	20.37
SPAA-CSA-ACAiN8Q5-GSF*	0.309	0.413	19.15	20.75
SPAA-CSA-ACAiiN8Q4-GSF	0.178	0.315	13.82	17.35
SPAA-CSA-GDAN8M4P4-GSF**	0.301	0.423	18.57	20.42
SPAA-CSA-GDAN8M8P4-GSF	0.312	0.396	19.21	20.71
SPAA-CSA-GDAN8M8P5-GSF	0.365	0.471	20.58	22.28
SPAA-CSA-GrAeN8R1P4-GSF***	0.312	0.398	19.20	20.78
SPAA-CSA-GrAeN8R2P2-GSF	0.179	0.308	13.81	17.35
SPAA-CSA-GrAeN8R2P4-GSF	0.305	0.418	18.57	20.40
SPAA-CSA-RCA	0.428	0.508	22.28	23.75
SPAA-RCA [3]	0.417	0.512	22.24	23.71

Table 2: Description of various parameters.

*N: adder size, Q: partition size
**N: adder size, M: number of partitions, P: Prediction bits
***N: adder size, R: Resultant, P: Prediction bits
Existing Approximate Adders (ACA, GDA, GrAe, CSPA)
MCSPA: Modified CSPA
CLA: Carry Looked Ahead
SPAA: Gaussian smoothing filter architecture in [3]
CSA: Carry Save Adder
RCA: Ripple Carry Adder

There are three kinds of proposed Gaussian filters following Table 1:

1-Speed-Power-Area-Accuracy Gaussian Filters: the conversion of the adder-tree structure to a carry-save adder is performed (SPAA-CSA-RCA).

2-Speed-Power-Area-Accuracy Gaussian Filters: the conversion of the adder-tree structure to an approximate carry-save adder is performed by employing existing approximate adders (GDA, ACA, CSPA, GeAr).

3-Speed-Power-Area-Accuracy Gaussian filters: the conversion of the adder-tree structure to an approximate carry-save adder is performed by applying the proposed adder (MCSPA).

Table 1 shows that among the suggested approximate Gaussian filters, the quality of the SPAA-CSA Gaussian filters, the accurate adder is applied in which, is identical to the SPAA method, and the image quality of the SPAA-CSA-MCSPA filter is almost equal to the image quality of the SPAA Gaussian filter. As illustrated in Figure 14, the quality of the SPAA Gaussian filter's output images is equivalent to the quality of the Gaussian filters proposed.

Figure14: a) Lena's Original Image; b) Lena's Noisy Image; c) Proposed Gaussian Filter Output by Applying CSA; d) Proposed Gaussian Filter Output by Employing Approximate Carry-Save Adder (by Applying the Proposed MCSPA Approximate Adder), e) Proposed Gaussian Filter Output by employing Approximate Carry-Save Adder (by Applying Carry Speculative Approximate Adder), f) Proposed Gaussian Filter Output using Approximate Carry-Save Adder (by Applying the Proposed Generic Accuracy Configurable Approximate Adder)

4.2. Synthesis Results

In accordance with the stages presented in Figure 13, the Xilinx ISE on the Virtex-7 family device 7VX330T are applied to design and synthesize the Gaussian filters in [3] and proposed Gaussian filters; Table 2 presents the results.

The findings indicate that the proposed 3x3 Gaussian filters' latency has been reduced to 44.8 percent and 22.6 percent, respectively, by applying approximate CSA (SPAA-CSA-MCSPA) and accurate CSA (SPAA-CSA-RCA), in comparison with the 3x3 Gaussian filter presented in [3]. The quantity of FPGA resources consumed in the suggested approaches has been reduced to 17 percent by employing approximate CSA, and 11 percent more resources were consumed by applying accurate CSA. Compared to existing

filters, the dynamic and static power usage of the proposed filter has also been moderately decreased. In Figure 15, the diagram demonstrates the power usage and various Gaussian filters' time latency. It is evident that all Gaussian filters' power consumption is constant; however, the proposed Gaussian filters' latency has been considerably reduced in comparison with Speed-Power-Area-Accuracy.

Table 3: Comparison of Power Consumption, Area, and Latency Matrices of Various Gaussian Filters.

	Power	Area (LUT)	Delay (ns)
Noisy Image	1.724	14	2.315
SPAA-CSA-MCSPA-GSF	1.718	23	3.221
SPAA-CSA-CSPA-GSF	1.718	20	2.684
SPAA-CSA-ACAiN8Q5-GSF*	1.718	18	2.638
SPAA-CSA-ACAiiN8Q4-GSF	1.718	22	3.181
SPAA-CSA-GDAN8M4P4-GSF**	1.718	27	3.491
SPAA-CSA-GDAN8M8P4-GSF	1.718	25	3.642
SPAA-CSA-GDAN8M8P5-GSF	1.718	20	2.658
SPAA-CSA-GrAeN8R1P4-GSF***	1.718	18	2.641
SPAA-CSA-GrAeN8R2P2-GSF	1.718	16	2.432
SPAA-CSA-GrAeN8R2P4-GSF	1.712	19	3.157
SPAA-CSA-RCA	1.712	12	4.151
SPAA-RCA [3]	1.738	16	4.121

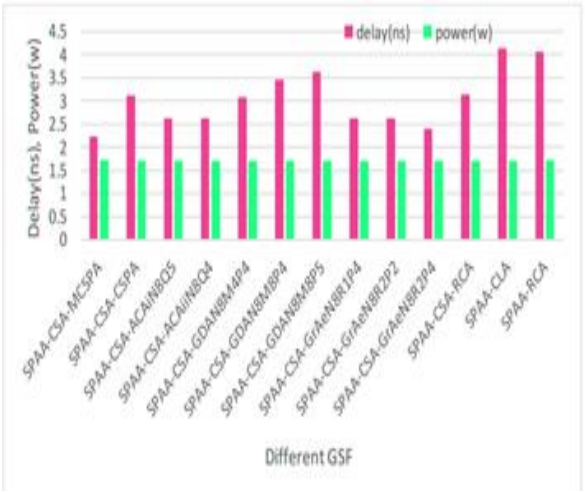

Figure15: Comparison between Various Gaussian Filters' Power Consumption and Latency

5. Summary and Conclusion

The authors provided a two-dimensional GSF architecture by applying approximate and accurate CSAs in the present study. The CSA adder, in the Gaussian Smoothing Filter structure, was replaced with the Adder-tree. Approximate adders were also applied in approximate CSAs, and a modern approximate adder was developed. The findings of the Gaussian filter's synthesis and simulation on field programmable gate arrays revealed that the proposed architecture has considerably lower latency in comparison with the previous Gaussian Smoothing Filters; hence, in the case of applying accurate and approximate CSAs, the latency reduction is around 22% and 45%, respectively. The output quality was decreased by 1.2% by applying an approximate CSA, which is insignificant in the majority of image processing applications.

References

[1] M. Basu, "Gaussian-based edge-detection methods—a survey," *IEEE Trans. Syst., Man, Cybern., Part C (Appl. Rev.)*, vol. 32, pp. 252–260, Apr. 2002.

[2] F. Krach, B. Frackelton, J. Carletta, and R. Veillette, "FPGA-based implementation of digital control for a magnetic bearing," in *Proc. Amer. Control Conf.*, vol. 2, pp. 1080–1085, 2003.

[3] A. Jaiswal, B. Garg, V. Kaushal, and G. Sharma, "SPAA-aware 2D Gaussian smoothing filter design using efficient approximation techniques," in *Proc. 28th Int. Conf. VLSI Design (VLSID)*, pp. 333–338, 2015.

[4] P.-Y. Hsiao, C.-H. Chen, S.-S. Chou, L.-T. Li, and S.-J. Chen, "A parameterizable digital-approximated 2D Gaussian smoothing filter for edge detection in noisy image," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 4, 2006.

[5] S. Khorbotly and F. Hassan, "A modified approximation of 2D Gaussian smoothing filters for fixed-point platforms," in *Proc. 43rd Southeastern Symp. Syst. Theory (SSST)*, pp. 151–159, 2011.

[6] 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. Arrang. Appl. (SPA)*, pp. 28–33, 2015.

[7] B. Garg and G. Sharma, "A quality-aware energy-scalable Gaussian smoothing filter for image processing applications," *Microprocess. Microsyst.*, vol. 35, pp. 1–9, 2016.
[8] M. Dubey and S. Agrawal, "An analysis of energy efficient Gaussian filter architectures," *Int. Res. J. Eng. Technol.*, vol. 4, pp. 1391–1397, 2017.
[9] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design Autom. Test Europe Conf.*, pp. 1250–1255, 2008.
[10] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, 2004.
[11] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error," 2009.
[12] N. Zhu, W. L. Goh, and K. S. Yeo, "Ultra low-power high-speed flexible probabilistic adder for error-tolerant applications," in *Proc. Int. SoC Design Conf. (ISOCC)*, pp. 393–396, 2011.
[13] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1225–1229, 2009.
[14] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *Proc. Int. SoC Design Conf.*, pp. 323–327, 2010.
[15] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, pp. 48–54, 2013.
[16] C. Lin, Y.-M. Yang, and C.-C. Lin, "High-performance low-power carry speculative addition with variable latency," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, pp. 1591–1603, 2015.
[17] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, pp. 1–6, 2015.
[18] N. H. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, STIA 85:47028, 1985.

Hadise Ramezani (Non-member) received a bachelor degree in software computer engineering from Shiraz Islamic Azad University, Shiraz, Iran in 2010 and M.Sc. degree in 2013. Currently she is pursuing the Ph.D. degree in Computer Systems Architecture from Kerman Branch, Islamic Azad University, Kerman, Iran. Her research interests are approximate computing, computer arithmetic and image Processing.

Email: h.ramezani@iauk.ac.ir

Majid Mohammadi (Non-member) was born in 16 September of 1966. He got B.Sc. of Electronic engineering in 1989 in Khajenasir university; M.Sc. of Control engineering in 1994 in Tehran university, and Ph.D. of Computer Architecture in 2009 in Shahid Beheshti university. He is lecturer in computer engineering department of Shahid Bahonar university since 1996. He interested in Analysis and design of quantum circuits, Reversible logic circuit design, multiple valued logic, Digital signal processing and its applications in sound, watermarking, steganography and cryptography.

Email: Mohammadi@uk.ac.ir

Amir Sabbagh Molahosseini (Non-member) received the B.Sc. degree from the Shahid Bahonar University of Kerman, Iran, in 2005, and the M.Sc. and Ph.D. degrees (Hons.) in computer engineering from Islamic Azad University, Science and Research Branch, Tehran, Iran, in 2007 and 2010, respectively. He was a Visiting Researcher with the Signal Processing Systems Group, Institute de Engenharia de Sistemas e Computadores, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal. He is currently an Assistant Professor with the Department of Computer Engineering, Islamic Azad University Kerman Branch, Kerman, Iran, and leads the High-Performance Computer Arithmetic Group, since 2010. His current research interests are computer arithmetic with special emphasis on residue number systems, and alternative computing systems with special emphasis on approximate computing.

Email: Sabbagh@iauk.ac.ir