

Optimizing gaussian filter implementation for canny edge detection using graph-based MCM algorithms

Lowkya Chandaka¹, Madhavi Dunna²

¹Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India

²Department of Electrical Electronics and Communication Engineering, GITAM School of Technology, Visakhapatnam, India

Article Info

Article history:

Received May 16, 2024

Revised Jul 21, 2025

Accepted Aug 8, 2025

Keywords:

Common sub-expression elimination

Filter implementation

Graph based algorithms

Multiplierless constant multiplications

Xilinx system generator

ABSTRACT

This study presents an optimized implementation of the gaussian filter in the Canny edge detection algorithm, focusing on reducing computational complexity while balancing power, timing, and resource utilization. Traditional implementations rely on the common subexpression elimination (CSE) algorithm for multiplierless constant multiplication, which results in high logic operations and resource consumption. To address this, we explore the constant array vector multiplication (CAVM) technique with two graph-based algorithms (exact GB and approximate GB). These algorithms offer a novel graph-structured approach to constant multiplication, differing from existing methods by modeling multiple paths to achieve optimal adder reuse. The architectures were implemented using Xilinx system generator (XSG) and evaluated in Vivado 2018.1. Experimental results reveal that both exact GB and approximate GB reduce logic operations and improve timing performance compared to CSE_csd. Among them, approximate GB achieves the fastest computation and lowest LUT utilization, making it the most hardware-efficient design. However, it exhibits the highest power consumption, whereas exact GB offers the best trade-off between speed and power efficiency. This optimization framework shows potential not only in image processing but also in embedded vision systems and low-power digital signal processing (DSP) applications. These findings demonstrate that GB Algorithms can effectively optimize gaussian filter design for real-time image processing applications.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Lowkya Chandaka

Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology
Hyderabad, India

Email: lowkyakrishna@gmail.com

1. INTRODUCTION

Multiple constant multiplications (MCMs) play a crucial role in various digital signal processing (DSP) applications, such as digital filter implementations, linear DSP transforms, and error-correction coding. The challenge in MCM-based designs lies in optimizing hardware area, delay, and power consumption, particularly when working with known constants. Conventional multiplier-based designs can be resource-intensive, making optimization essential. To avoid hardware-intensive multipliers, designers often rely on additions, subtractions, and shift operations to implement constant multiplications. While this forms the basis of traditional optimization methods such as common subexpression elimination (CSE), more recent approaches like graph-based (GB) algorithms offer a structured way to further minimize arithmetic complexity [1]-[4].

One key advantage of shift operations is that they do not require additional hardware, as shifts can be realized using simple wire connections. Consequently, the primary optimization challenge in MCM design is to determine the minimum number of adders and subtractors required to implement constant multiplications efficiently. Several algorithms have been proposed in the literature to minimize the number of add/sub operations through maximizing partial product sharing [5]-[11]. These optimization approaches can be broadly classified into two categories: i) CSE algorithms, which identify and reuse repeated subexpressions to minimize redundant computations and ii) GB methods, which represent multiplication as a directed acyclic graph and optimize computations by reducing the number of operations.

Researchers have developed innovative MCM optimization techniques leveraging these algorithms. For instance, Roy and Chandra [12] introduced triangular common subexpression elimination (TCSE), which combines horizontal and vertical CSE horizontal common subexpression elimination (HCSE) and vertical common subexpression elimination (VCSE) to minimize logic operators. Similarly, Bose *et al.* [13] proposed the matrix grouped CSE Algorithm, reducing computational complexity in finite impulse response (FIR) filters. Odugu *et al.* [14] optimized two-dimensional circular symmetric FIR filters using a modified Park–McClellan transformation method, achieving improved area, power, and speed performance through CSE-based coefficient encoding. Furthermore, Mert *et al.* [15] proposed a high-level synthesis method for implementing MCM operations using DSP blocks in Xilinx field programmable gate arrays (FPGAs), achieving a 35.8% reduction in DSP block usage for HEVC 2D DCT implementation. These approaches, while beneficial, do not exploit advanced MCM optimization techniques such as GB methods, leaving significant potential for improvement in terms of hardware efficiency and computational cost. Although these methods have been successfully applied to FIR filters, their effectiveness for Gaussian filters remains largely unexplored. Unlike FIR filters, Gaussian filters involve symmetric coefficients and are widely used in image smoothing and pre-processing. Yet little work has focused on minimizing their hardware cost using dedicated MCM optimization strategies. Existing methods for Gaussian filtering [16]-[20] focus primarily on adder optimization rather than optimizing the entire computational structure, which limits the potential for reducing hardware complexity and power consumption. This creates a gap in literature where efficient multiplierless architectures for Gaussian filtering remain unexplored. Our work builds on the method described in [21], an enhanced multiplierless constant array vector multiplication (CAVM) algorithm that combines exact and approximate strategies to minimize adders and delay under error constraints.

The novelty of this work lies in applying GB MCM algorithms to Gaussian filters—an area largely unexplored—and benchmarking them against the traditional CSE_csd approach in terms of logic depth, speed, power, and resource efficiency. Unlike previous methods, this work is the first to apply GB MCM optimization specifically to Gaussian filters, highlighting its effectiveness in reducing computational overhead while enhancing efficiency. The proposed approach minimizes the use of adders and subtractors while ensuring a well-structured and resource-efficient hardware design, making it highly suitable for real-time image processing applications.

2. METHOD AND IMPLEMENTATION

This section presents the proposed design approach for optimizing the Gaussian filter without multipliers and its integration into the Canny edge detection algorithm. The design flow is illustrated in Figure 1.

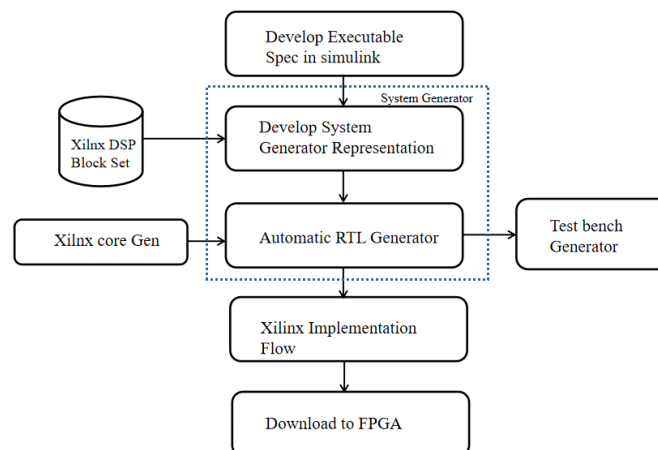


Figure 1. Design method

2.1. Gaussian filter architecture development

The first stage involves developing algorithms and constructing models using the Xilinx system generator (XSG) block set library in MATLAB/Simulink. The Gaussian filter is designed using adder and shift-based methods instead of multipliers, significantly reducing computational complexity. The filter architectures are implemented within the Canny edge detection algorithm using Xilinx block set components, incorporating pre-processing and post-processing blocks [22]-[25] as shown in Figure 2. The models are simulated in the MATLAB/Simulink environment with appropriate parameters, and results are viewed using a video viewer. These models are saved as .slx files. XSG enables automatic hardware description language (HDL) code generation, producing user constraint files (UCF), test benches, and test vectors for design verification. The generated HDL netlist is synthesized in Vivado 2018.2 to estimate hardware resource utilization, timing, and power consumption.

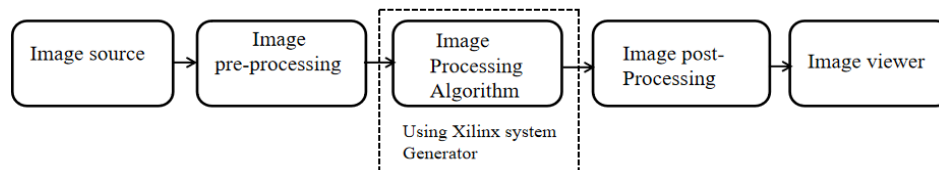


Figure 2. Hardware implementation of image processing in XSG

2.2. Constant multiplication optimization techniques

Three Gaussian filter architectures were developed using three different MCM algorithms, applied to a 3×3 Gaussian kernel Figure 3 [5], [21].

$$\frac{1}{256} * \begin{array}{|c|c|c|} \hline 21 & 31 & 21 \\ \hline 31 & 48 & 31 \\ \hline 21 & 31 & 21 \\ \hline \end{array}$$

Figure 3. Gaussian kernel for 3×3 matrix

The Gaussian filter is commonly used for image smoothing, reducing noise before edge detection. It employs a 2D Gaussian kernel in both x and y directions, mathematically as (1).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

where σ^2 represents the variance, a key parameter affecting the filter characteristics.

Since convolution with a 3×3 Gaussian kernel requires over 7.52 million multiplications and 6.29 million additions for a 512×512 colour image, hardware-based optimization is crucial. Instead of multipliers, adder-tree structures are employed to enhance computational performance while reducing area and power consumption.

2.3. Gaussian filter implementation using common subexpression elimination-canonical signed digit technique

One optimization method uses canonical signed digit (CSD) representation combined with CSE. The CSD format represents numbers using $\{-1, 0, 1\}$, reducing the number of addition operations compared to binary representation [5].

For example:

Binary value of 21: 10101 → CSE_csd representation: $x \ll 4 + x \ll 2 + x$

Binary value of 31: 11111 → CSE_csd representation: $x \ll 5 - 1$

Binary value of 48: 110000 → CSE_csd representation: $x \ll 6 - x \ll 4$

The optimized CSE_csd Gaussian filter architecture is depicted in Figure 4.

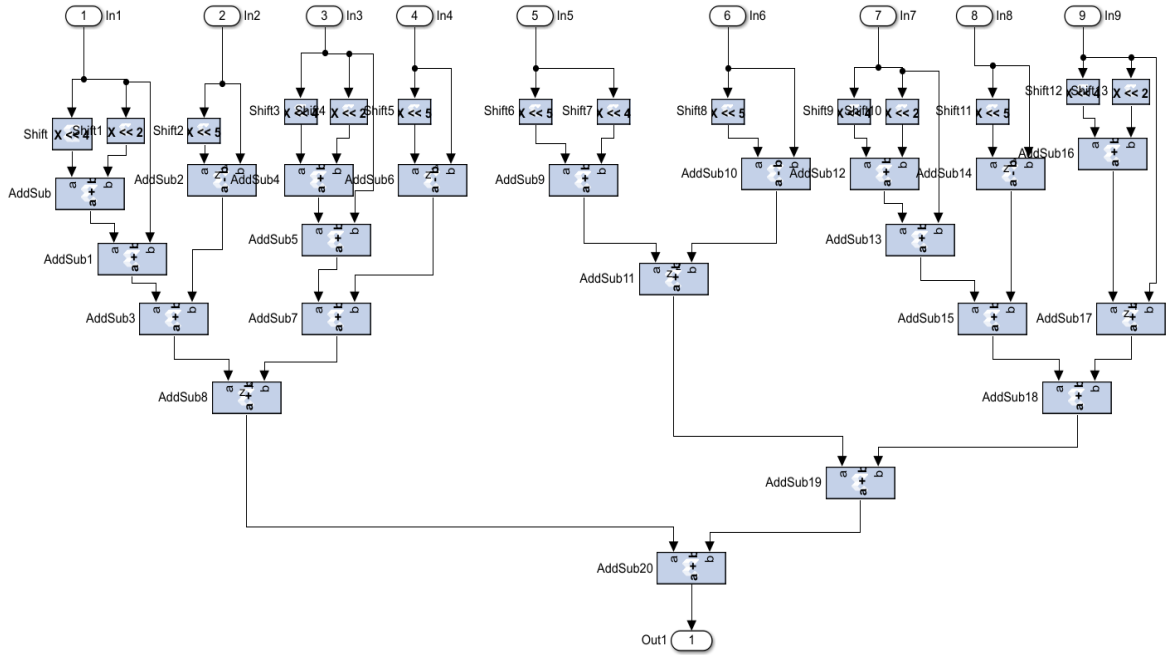


Figure 4. Gaussian image filter architecture using CSE_csd method (CSE_csd Gaussian filter)

2.4. Gaussian filter implementation using graph-based algorithms

A GB approach is used for multiplierless realization of 9 Gaussian coefficients, treating the 3×3 kernel as a 1×9 constant array:

$$Y = [21 \ 31 \ 21 \ 21 \ 31 \ 48 \ 31 \ 21 \ 31 \ 21]$$

$$= 21A_1 + 31A_2 + 21A_3 + 31A_4 + 48A_5 + 31A_6 + 21A_7 + 31A_8 + 21A_9$$

$$= 21(A_1 + A_3 + A_7 + A_9) + 31(A_2 + A_4 + A_6 + A_8) + 48A_5$$

where A1–A9 are inputs, grouped as:

$$a = (A_1 + A_3 + A_7 + A_9)$$

$$b = (A_2 + A_4 + A_6 + A_8)$$

$$c = A_5$$

Hence

$$Y = 21a + 31b + 48c$$

The resulting GB Gaussian filter architectures are shown in Figures 5(a) and (b) (see in Appendix), representing exact GB and Approx GB algorithms [16], [21].

2.5. Integration with Canny edge detection algorithm

The Canny edge detection algorithm Figure 6 consists of four main stages:

- Image smoothing—reducing noise (Gaussian filtering)
- Gradient calculation—using Sobel operators to compute edge intensity
- Non-maximum suppression—eliminating weak edges
- Hysteresis thresholding—finalizing edge detection

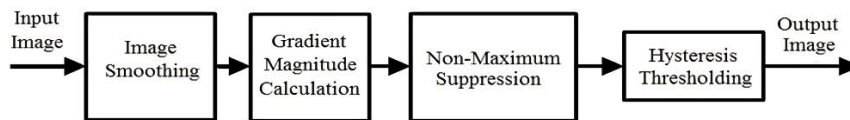


Figure 6. Block diagram for Canny edge detection algorithm

In our implementation, Gaussian filter blocks replace the traditional smoothing block, reducing computational cost while maintaining edge detection accuracy. The remaining Canny edge detection blocks are designed in XSG using Simulink.

2.6. Field programmable gate array-based implementation and performance analysis

To evaluate the efficiency of our designs, the optimized Gaussian filters integrated in Canny edge detection architectures were converted to Verilog code and implemented on an FPGA using Vivado 2018.2. The following metrics were analysed: i) hardware resource utilization—LUTs, FFs, DSP slices, and BRAMs, ii) timing analysis, and iii) power consumption.

3. RESULTS AND DISCUSSION

3.1. Logic operations and depth analysis

The efficiency of the Gaussian filter architectures was evaluated in terms of logic operations (adders/subtractors) and logic depth (add-shift tree levels). Figure 7 compares the three architectures: CSE_csd Gaussian Filter required the most logic operations (21) with a depth of 6. Exact GB Gaussian Filter reduced logic operations to 11 but had a depth of 7. Approx GB Gaussian filter achieved the lowest depth (5) with 12 operations, demonstrating that GB methods reduce logic complexity, with Approx GB Gaussian Filter offering the least depth.

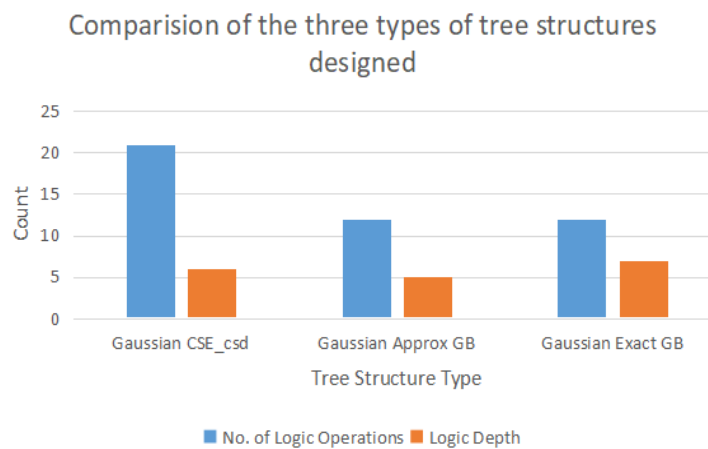


Figure 7. Comparison chart

3.2. Resource utilization analysis

Table 1 summarizes resource utilization. The CSE_csd Gaussian filtered Canny (CSE_csd GFC) architecture occupied the most LUTs (1487, 2.8%), while Approx GBGFC architecture had the least (898, 1.69%). Flip-flop (FF) usage was highest in Approx GBGFC architecture (1286, 1.21%) and lowest in CSE_csd (1166, 1.10%). BRAM usage was constant (12.86%) across all architectures, with minimal variation in I/O and BUFG utilization. Approx GBGFC architecture achieved the best hardware efficiency, significantly reducing LUT consumption.

Table 1. Resource utilization of the three architectures

Resource	Available	CSE_csd GFC	Utility %	Exact GBGFC	Utility %	Approx GBGFC	Utility %
LUT	53200	1487	2.80	1022	1.925	898	1.69
LUTRAM	17400	12	0.07	-	-	-	-
FF	106400	1166	1.10	1280	1.20	1286	1.21
BRAM	140	18	12.86	18	12.86	18	12.86
IO	200	50	25.00	49	24.50	50	25.00
BUFG	32	1	3.13	1	3.13	1	3.13

3.3. Timing analysis

Table 2 presents timing analysis. The CSE_csd GFC architecture had the lowest worst negative slack (WNS) (68 ps), indicating a weaker setup margin. Exact GBGFC architecture (335 ps) and Approx GBGFC (403 ps) performed better. Worst hold slack (WHS) values were similar across architectures, with Exact GBGFC at 49 ps and Approx GBGFC slightly better (79 ps) than CSE_csd GFC (78 ps). Worst pulse

width slack (WPWS) was highest in CSE_csd GFC (7.645 ns), while Exact GBGFC and Approx GBGFC (4.500 ns) demonstrated better clock pulse stability. The Approx GBGFC offered the fastest computation, making it ideal for high-speed applications.

Table 2. Timing analysis of the three architectures

Architectures	WNS (ps)	WHS (ps)	WPWS (ns)
CSE_csd GFC	68	78	7.645
Exact GBGFC	335	49	4.5
Approx GBGFC	403	79	4.5

3.4. Power analysis

Table 3 details power analysis. The CSE_csd GFC consumed the least total power (209 mW), followed by Exact GBGFC (210 mW), and Approx GBGFC the most (211 mW). Dynamic power was highest for Approx GBGFC (104 mW) and lowest for CSE_csd GFC (102 mW), while static power was constant (107 mW) across all architectures. While Approx GBGFC provided the fastest computation, Exact GBGFC delivered a balanced trade-off between power and speed.

Table 3. Power comparison of the three architectures

Power parameters	CSE_csd GFC architecture	Exact GBGFC architecture	Approx GBGFC architecture
Total on-chip power	209 mW	210 mW	211 mW
Dynamic power	102 mW	103 mW	104 mW
Static power	107 mW	107 mW	107 mW
Clocks power	7 mW	8 mW	7 mW
Signals power	3 mW	3 mW	4 mW
Logic power	3 mW	3 mW	3 mW
BRAM power	89 mW	89 mW	89 mW
I/O power	<1 mW (μ W range)	<1 mW (μ W range)	<1 mW (μ W range)

3.5. Overall trade-off analysis

Table 4 compares all parameters. Approx GBGFC excelled in speed and hardware efficiency but had the highest power consumption. Exact GBGFC provided a balance between speed and power. CSE_csd GFC had the highest resource usage and lowest speed but was the most power-efficient. Thus, Exact GBGFC is ideal for applications balancing speed and power, while Approx GBGFC is preferable when speed and area are critical.

Table 4. Comprehensive analysis of the three architectures

Parameter	CSE_csd GFC architecture	Exact GBGFC architecture	Approx GBGFC architecture	Best choice
Timing (ps/ns)	Highest delay (slowest)	Moderate	Lowest delay (fastest)	Approx GBGFC architecture
Total power (mW)	Lowest	Moderate	Highest	Exact GBGFC architecture
Dynamic power (mW)	Lowest	Moderate	Highest	Exact GBGFC architecture
Static power (mW)	Same across all	Same across all	Same across all	-
LUT utilization	Highest	Moderate	Lowest	Approx GBGFC architecture
FF utilization	Lowest	Moderate	Highest	Approx GBGFC architecture
BRAM utilization	Same across all	Same across all	Same across all	-
IO utilization	Similar	Similar	Similar	-
Overall efficiency	Less efficient	Balanced (better speed and moderate power)	Most efficient in speed but consumes more power	Exact GBGFC architecture best tradeoff

4. CONCLUSION

This study focused on optimizing the Gaussian filter in the Canny edge detection algorithm by evaluating three different architectures: CSE_csd, Exact GB, and Approx GB. The common subexpression elimination (CSE_csd) method, though designed for multiplierless constant multiplication, required more logic operations and consumed more resources, making it less ideal for resource-constrained applications. To

overcome these challenges, we employed the CAVM technique using Exact GB and Approx GB algorithms. Our analysis demonstrated that Exact GBGFC and Approx GBGFC architectures significantly reduced computational complexity and improved hardware efficiency compared to the CSE_csd GFC approach. Among them, Approx GB Gaussian Filter achieved the lowest logic depth and fastest computation, making it highly efficient in terms of timing and resource utilization. However, this came at the cost of increased power consumption. In contrast, Exact GBGFC provided a well-balanced trade-off between power efficiency and performance, making it a better option for power-sensitive applications. Overall, our findings confirm that GB algorithms (GBAs) offer an effective way to optimize Gaussian filters for real-time image processing applications. While the Approx GBGFC architecture is well-suited for high-speed and resource-efficient implementations, Exact GBGFC offers a practical balance between power and performance. However, this work is limited in scope to a single FPGA platform and does not yet account for dynamic power fluctuations or scalability to higher-resolution filters. As a potential extension, incorporating approximate adders within add-shift trees could further enhance efficiency by taking advantage of error tolerance in real-time image processing. Future work could include validating these architectures across different FPGA platforms and embedding them within real-time hardware systems for live video or medical imaging pipelines.

ACKNOWLEDGMENTS

The authors acknowledge the support provided by GITAM Deemed to be University in the form of computing facilities and tools used in this research. This work was conducted without a specific research grant or contract.

FUNDING INFORMATION

This research was carried out without external research grant or contract support.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Lowkya Chandaka	✓	✓	✓	✓	✓	✓		✓	✓	✓				
Madhavi Dunna		✓		✓	✓		✓	✓		✓	✓	✓	✓	

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nvestigation

R : **R**esources

D : **D**ata Curation

O : Writing - **O**riginal Draft

E : Writing - Review & **E**ditting

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

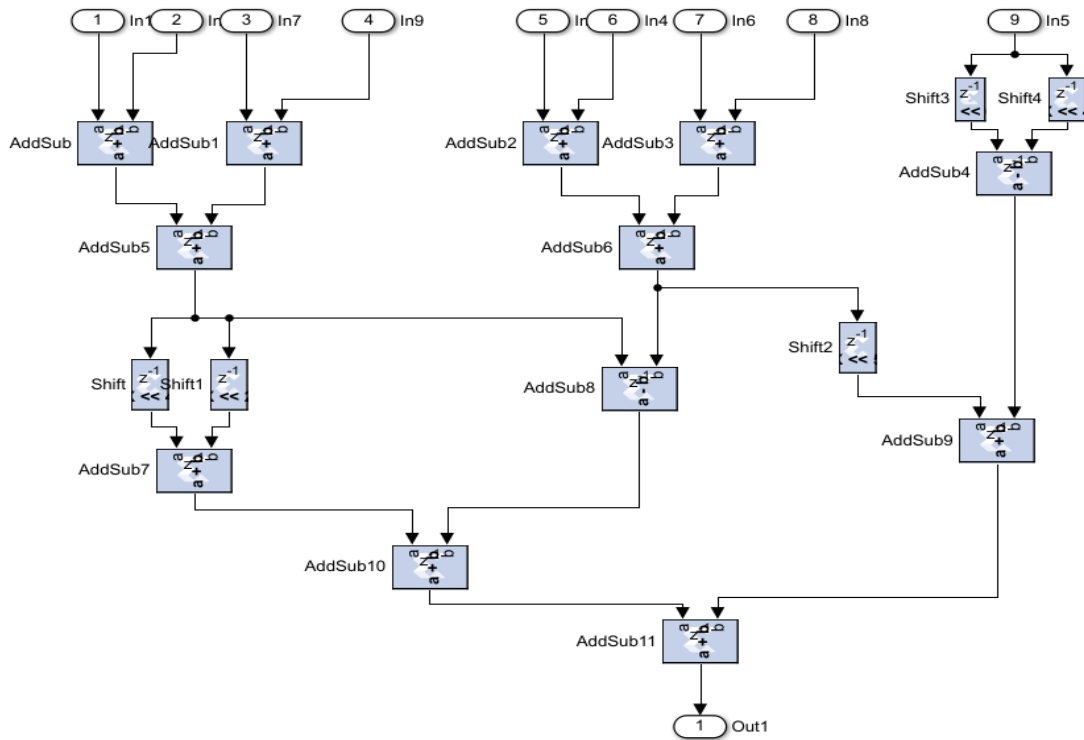
The authors confirm that the data supporting the findings of this study are available within the article and its supplementary materials.

REFERENCES

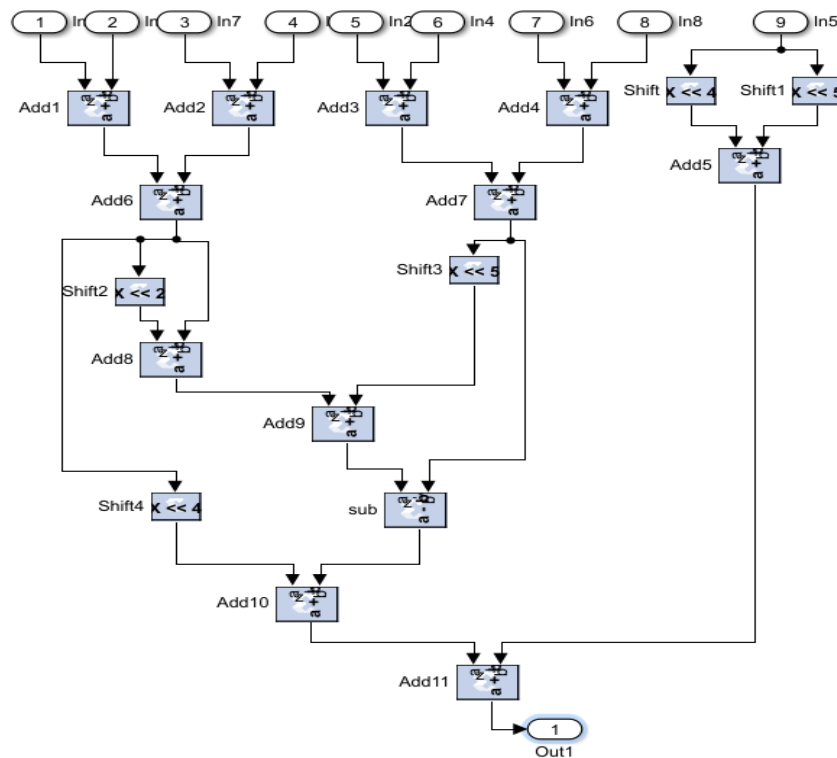
- [1] R. Garcia and A. Volkova, "Toward the Multiple Constant Multiplication at Minimal Hardware Cost," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 5, pp. 1976–1988, 2023, doi: 10.1109/TCSI.2023.3241859.
- [2] C. R. Kumar, J. Raghavendra, D. Kulkarni, E. Y. Alhawsawi, and M. A. Majid, "High-Performance and Energy-Efficient FIR Filter Architecture Using Parallel Prefix Adder-Based Triangular Common Subexpression Elimination Algorithm for IoT Enabled Wireless Sensor Network," in *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, IEEE, Aug. 2023, pp. 1–8, doi: 10.1109/ASIANCON58793.2023.10270103.

- [3] T. Habermann, J. Kuhle, M. Kumm, and A. Volkova, "Hardware-Aware Quantization for Multiplierless Neural Network Controllers," in *APCCAS 2022 - 2022 IEEE Asia Pacific Conference on Circuits and Systems*, IEEE, Nov. 2022, pp. 541–545, doi: 10.1109/APCCAS55924.2022.10090271.
- [4] M. Kumm, A. Volkova, and S. I. Filip, "Design of Optimal Multiplierless FIR Filters with Minimal Number of Adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 658–671, 2023, doi: 10.1109/tcad.2022.3179221.
- [5] N. Fiege, M. Kumm, and P. Zipf, "Bit-Level Optimized Constant Multiplication Using Boolean Satisfiability," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 1, pp. 249–261, 2024, doi: 10.1109/TCSI.2023.3327814.
- [6] R. Garcia, A. Volkova, M. Kumm, A. Goldsztejn, and J. Kuhle, "Hardware-Aware Design of Multiplierless Second-Order IIR Filters With Minimum Adders," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1673–1686, 2022, doi: 10.1109/TSP.2022.3161158.
- [7] G. K. Kumar *et al.*, "Area-, Power-, and Delay-Optimized 2D FIR Filter Architecture for Image Processing Applications," *Circuits, Systems, and Signal Processing*, vol. 42, no. 2, pp. 780–800, 2023, doi: 10.1007/s00034-022-02232-y.
- [8] N. Sajwan, I. Sharma, A. Kumar, and L. K. Balyan, "Performance of Multiplierless FIR Filter Based on Directed Minimal Spanning Tree: A Comparative Study," *Circuits, Systems, and Signal Processing*, vol. 39, no. 11, pp. 5776–5800, 2020, doi: 10.1007/s00034-020-01433-7.
- [9] R. Garcia, A. Volkova, and M. Kumm, "Truncated Multiple Constant Multiplication with Minimal Number of Full Adders," in *Proceedings - IEEE International Symposium on Circuits and Systems*, May. 2022, pp. 263–267, doi: 10.1109/ISCAS48785.2022.9937441.
- [10] C. Kalamani, S. Lekashri, A. N. Duraivel, and T. S. R. Raj, "An efficient reconfigurable FIR filter design with coefficient optimization using a modified bacterial foraging optimization algorithm," *Automatika*, vol. 65, no. 1, pp. 290–303, 2024, doi: 10.1080/00051144.2023.2296792.
- [11] P. Kumar, P. C. Shrivastava, M. Tiwari, and G. R. Mishra, "High-Throughput, Area-Efficient Architecture of 2-D Block FIR Filter Using Distributed Arithmetic Algorithm," *Circuits, Systems, and Signal Processing*, vol. 38, no. 3, pp. 1099–1113, 2019, doi: 10.1007/s00034-018-0897-2.
- [12] S. Roy and A. Chandra, "A Triangular Common Subexpression Elimination Algorithm with Reduced Logic Operators in FIR Filter," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3527–3531, 2020, doi: 10.1109/TCSII.2020.2992325.
- [13] S. Bose, A. De, and I. Chakrabarti, "Area-Delay-Power Efficient VLSI Architecture of FIR Filter for Processing Seismic Signal," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 11, pp. 3451–3455, 2021, doi: 10.1109/TCSII.2021.3081257.
- [14] V. K. Odugu, C. V. Narasimhulu, and K. S. Prasad, "Design and Implementation of Low Complexity Circularly Symmetric 2D FIR Filter Architectures," *Multidimensional Systems and Signal Processing*, vol. 31, no. 4, pp. 1385–1410, 2020, doi: 10.1007/s11045-020-00714-3.
- [15] A. C. Mert, H. Azgin, E. Kalali, and I. Hamzaoglu, "Efficient Multiple Constant Multiplication Using DSP Blocks in FPGA," in *Proceedings - 2018 International Conference on Field-Programmable Logic and Applications, FPL 2018*, IEEE, Aug. 2018, pp. 331–334, doi: 10.1109/FPL.2018.00063.
- [16] L. Aksoy, D. B. Roy, M. Imran, P. Karl, and S. Pagliarini, "Multiplierless Design of Very Large Constant Multiplications in Cryptography," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 11, pp. 4503–4507, 2022, doi: 10.1109/TCSII.2022.3191662.
- [17] L. B. Soares, M. M. A. D. Rosa, C. M. Diniz, E. A. C. D. Costa, and S. Bampi, "Design Methodology To Explore Hybrid Approximate Adders for Energy-Efficient Image and Video Processing Accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2137–2150, 2019, doi: 10.1109/TCSI.2019.2892588.
- [18] J. D. Oliveira, L. Soares, E. Costa, and S. Bampi, "Exploiting Approximate Adder Circuits for Power-Efficient Gaussian and Gradient Filters for Canny Edge Detector Algorithm," in *LASCAS 2016 - 7th IEEE Latin American Symposium on Circuits and Systems, R9 IEEE CASS Flagship Conference*, IEEE, Feb. 2016, pp. 379–382, doi: 10.1109/LASCAS.2016.7451089.
- [19] Y. Kang, J. Kim, and S. Kang, "Novel Approximate Synthesis Flow for Energy-Efficient FIR Filter," in *Proceedings of the 34th IEEE International Conference on Computer Design, ICCD*, IEEE, Oct. 2016, pp. 96–102, doi: 10.1109/ICCD.2016.7753266.
- [20] F. Ahmadi, M. R. Semati, H. Daryanavard, and A. Minaeifar, "Energy-efficient approximate full adders for error-tolerant applications," *Computers and Electrical Engineering*, vol. 110, p. 108877, 2023, doi: 10.1016/j.compeleceng.2023.108877.
- [21] L. Aksoy, P. Flores, and J. Monteiro, "A Novel Method for The Approximation of Multiplierless Constant Matrix Vector Multiplication," *Eurasip Journal on Embedded Systems*, vol. 2016, no. 1, pp. 1–11, 2016, doi: 10.1186/s13639-016-0033-y.
- [22] M. Mekhfioui, "Real Time Hardware Co-Simulation for Blind Image Separation Algorithm Using ZYNG 7000 & Xilinx System Generator," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 2, pp. 365–371, 2020, doi: 10.30534/ijeter/2020/21822020.
- [23] A. Benahmed, M. Mekhfioui, and Z. Guennoun, "FPGA based Hardware Co-Simulation Implementation for Real-Time Image Blind Separation using ICA Algorithms," *International Journal of Emerging Technology and Advanced Engineering*, vol. 12, no. 10, pp. 75–81, 2022, doi: 10.46338/ijetae1022_09.
- [24] M. Bilal, W. I. Harasani, and L. Yang, "Rapid Prototyping of Image Contrast Enhancement Hardware Accelerator on FPGAs Using High-Level Synthesis Tools," *Jordan Journal of Electrical Engineering*, vol. 9, no. 3, pp. 322–337, 2023, doi: 10.5455/jjee.204-1673105856.
- [25] B. M. Krishna, G. R. Chowdary, C. Santhosh, S. K. A. Kalam, and K. P. K. L. Naidu, "Implementation of Xilinx system generator-based image processing algorithms through FPGA," in *AIP Conference Proceedings*, 2024, p. 020011, doi: 10.1063/5.0159026.

APPENDIX






(a)






(b)

Figure 5. Gaussian architectures based on GB algorithms; (a) Approx GB Gaussian filter and (b) exact GB Gaussian filter

BIOGRAPHIES OF AUTHORS

Lowkya Chandaka    is pursuing Ph.D. (part time) in GITAM Deemed to be University. She has done her M.Tech. from Pydah College of Engineering in 2011, B.Tech. from Avanathi Institute of Science and Technology in 2009. She is having 12 years of teaching experience. She is presently working as an Assistant Professor in Sreenidhi Institute of Science and Technology. Her areas of interest include VLSI architectures and low power design. She can be contacted at email: lowkyakrishna@gmail.com.



Madhavi Dunna    has done Ph.D. from Andhra University in 2018, M.Tech. from Andhra University in 2004, AMIE in 2000. She secured Suman Sharma award for highest total in AMIE. She is having 20 years of teaching and research experience. She is working as an Associate Professor in GITAM Deemed to be University, Visakhapatnam. Her areas of interest include image processing, VLSI, and signal processing. She can be contacted at email: dmadhavi336@gmail.com.