International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 01 | Jan -2017 www.irjet.net p-ISSN: 2395-0072

An Analysis of Energy Efficient Gaussian Filter Architectures

Mrinal Dubey[1] and Shweta Agrawal[2]

[1]Research Scholar, [2]Assistant Professor,
Dept. of Electronics and Comm., SRCEM Banmore, Morena, India

Abstract - *The high usage of the portable multimedia devices such as camera, phones etc. demanding highly compute intensive operations. Therefore, energy efficient design techniques are prime requirement to reduce the battery lifetime. To achieve energy efficient designs several approaches have been proposed. Among the various techniques, the approximate computing has emerged as the promising alternative to the conventional approaches and provides very good energy efficiency. The Gaussian filter is most commonly used the pre-processing filter before edge detection to remove the unwanted edges due to noise. Most of the existing architectures for energy efficient Gaussian filtering include approximation of kernel coefficients, change of window size, architectural modification and employing approximate arithmetic in the accurate design. This paper reviews different energy efficient filter architectures and compares them.*

Key Words: Image Processing, Gaussian Filter, Integrated Circuits, Quality Tunable designs.

1. INTRODUCTION

Recently, there is increasingly usage of the portable devices employing multimedia applications such as camera, phones etc. demanding highly compute intensive applications on battery operated devices. Therefore, energy efficient design techniques are becoming more popular to reduce the battery lifetime [1]. In order to achieve energy efficient designs several approaches from algorithm/architecture to circuit level have been proposed. Among the various techniques, the approximate computing has emerged as the promising alternative to the conventional approaches and provides very good energy efficiency for the error tolerant applications. Since most image processing applications produces output for human consumption which exhibits limited perception, small amount of error in the output image cannot be discerned. All these applications are commonly called as error tolerant applications. In these applications, small amount of error is acceptable. Therefore, approximate designs are developed for various image processing computing cores. The edge detection is the frequently used operation in several computer vision applications. Since the Gaussian filtering (GF) the most compute intensive operation within edge detection, energy efficient design of GF is prime requirement.

The GF is the weighted non-linear filter and is used to blur and de-noise the image. It is most commonly used the pre-processing filter before edge detection to remove the unwanted edges due to noise [2], [3]. To efficiently implement the GF, the Gaussian equation is approximated by a kernel of different sizes. Larger the size of kernel better is the approximation of Gaussian expression that provides higher quality at the cost large computational complexity. In order to achieve filtered image, convolution of the Gaussian kernel with the image sub-matrix is performed. Since the accurate kernel exhibits floating point constant multiplier, it consumes large energy.

Most of the existing GF filter design exhibits approximation of kernel coefficients [4], change of window size, architectural modification and employing approximate arithmetic in the accurate design. The floating point kernel coefficients are approximated to fixed point and sum of power of two form to reduce the implementation complexity [5]. Further, different window size of kernel is considered for obtaining quality energy trade-off. Furthermore, some approaches modify the architectures for obtaining energy scalable Gaussian filtering [6], [7]. Finally, some of the energy efficient architectures are obtained by embedding approximate adder [8], [9]. These architectures provide quality energy trade-off.

The remainder of this paper explores different Gaussian filters with comparative analysis by implementation and simulation.

2. GAUSSIAN FILTER

The Gaussian distribution can be represented by the Equation 1 given below.

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\varepsilon^{-(x-m)^2/2\sigma^2} \quad (1)$$

where, m, g and σ represent the mean or average, gray level and the standard deviation of the noise respectively. Its distribution can be represented by a bell shaped graph.

As the Gaussian filtering is commonly employed in different image processing applications such as edge detection to remove unwanted edge, image mosaicking and tone mapping etc., it requires 2D Gaussian expression due to the 2D nature of the image. The 2D Gaussian expression is represented by Eq. 2.

$$g(x,y) = \varepsilon^{\frac{-(x^2+y^2)}{2\sigma^2}} \quad (2)$$

where, x and y are the variables representing the coordinate while sigma representing the standard deviation.

The image is filtered by processing through the Gaussian expression. The processing of image through the above expression requires implementation of above expression in the software form which performance inefficient.

Therefore, hardware efficient implementation is done for the Gaussian filtering. To reduce the implementation complexity, the above expression can be approximated by a matrix called kernel. The image processed through the kernel provides the same result as the above expression. The accurate representation of the Gaussian expression using kernel of 5x5 size is given by Eq. 3.

$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.596 & 0.0983 & 0.0996 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.596 & 0.0983 & 0.0996 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \quad (3)$$

This matrix is achieved by varying the value of variable x, y from -2 to +2 with σ=1. From the Figure it can be seen that direct implementation requires floating point multiplier to achieve smoothened pixel, therefore different approximations are done to achieve kernel coefficient which are hardware efficient. Based on the approximated coefficients, different architectures are developed which are explored in the following section.

3. ENERGY EFFICIENT FILTER DESIGN

This section reviews different energy efficient kernels developed to efficiently compute the filtered pixel.

3.1 Fixed-point Gaussian Kernel

To reduce the implementation complexity, the kernel coefficients are approximated to fixed point [4]. In fixed point representation in (l, m) format, l represents the number of bits while m represents location of coefficient least significant bits. For example, decimal equivalent of the coefficient X3X2X1X0 in a (4,3) format is X3.26+X2.25+X1.24+X0.23 While the decimal equivalent of the same coefficient in a (4, -3) format is X3.20+X2.2-1+X1.2-2+X0.2-3. The kernel rounded to the (2, -4) and (6, -8) data formats are given by Eq. (4) and Eq. (5) respectively.

$$\frac{1}{2^4}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 3 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4)$$

$$\frac{1}{2^8}\begin{bmatrix} 1 & 3 & 6 & 3 & 1 \\ 3 & 15 & 25 & 15 & 3 \\ 6 & 25 & 41 & 25 & 6 \\ 3 & 15 & 25 & 15 & 3 \\ 1 & 3 & 6 & 3 & 1 \end{bmatrix} \quad (5)$$

These kernels reduce implementation complexity of the filter which results in significant reduction in power, area and delay metrics. At the same time the proposed kernels provide the acceptable output quality.

3.2 Digital Approximated 2D Gaussian Filter

An approximated kernel that significantly reduces the implementation complexity of the Gaussian kernel is reported in [5]. In this approximate kernel each coefficient is approximated in sum of power-of-two. As multiplication of any term with a constant having value in power of two will not require hardware for implementation. The kernel with value in power-of-two is given by the Eq. 6 below.

$$(2^{-2}+2^{-6})\begin{bmatrix} 0 & 2^{-2} & 2^{-2}+2^{-4} & 2^{-2} & 0 \\ 2^{-2} & 2^{-2}+2^{-2} & 2^{-1}+2^{-2} & 2^{-2}+2^{-2} & 2^{-2} \\ 2^{-2}+2^{-4} & 2^{-1}+2^{-2} & 2^{0}+2^{-4} & 2^{-1}+2^{-2} & 2^{-2}+2^{-4} \\ 2^{-2} & 2^{-2}+2^{-2} & 2^{-1}+2^{-2} & 2^{-2}+2^{-2} & 2^{-2} \\ 0 & 2^{-2} & 2^{-2}+2^{-4} & 2^{-2} & 0 \end{bmatrix} \quad (6)$$

A simplified architecture that utilized the kernel coefficients in the power-of-two form and compute the smoothened pixel is shown Fig. 1.

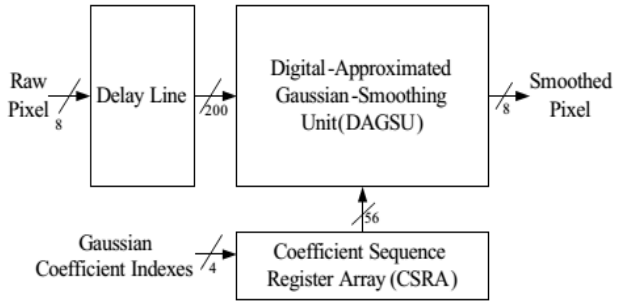


Fig-1: Digital approximated filter architecture.

3.3 Low Complexity Filter using Neighbour Pixels Similarity

Since the neighbour pixel in an image exhibits higher correlation i.e. their values will be nearly same, this property can be exploited to reduce the implementation complexity of the Gaussian filter. As the natural and most of the images have edge which is very small in number, it makes assumption of adjacent pixel to be nearly same true. In case of image smoothing, an image sub-matrix is considered for processing for example a 5x5 or 3x3 image sub-matrix. Therefore, concept of neighbour pixel to be same i.e. neighbour pixel similarity (NPS) if applied to this sub-matrix will significantly reduce the computation complexity of the smoothening filter [6]. For example, an original sub-matrix of size 3x3 as shown in Fig. 2 replaces all pixels of a row by its single value. This replacement will not cause large error the values are very near.

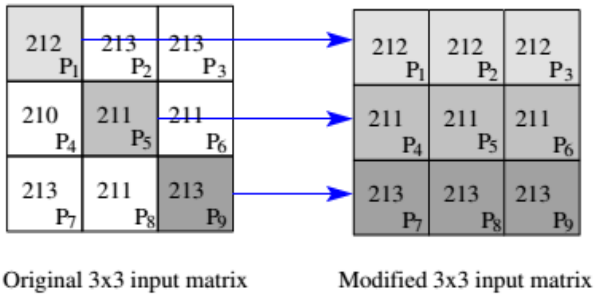


Fig-2: NPS in 3x3 image sub-matrix.

In the Fig. 2, the pixels of an image sub-matrix row are approximated to single value where the coefficient of each row can be added and multiplied with the resulting pixel, as it will provide nearly same value. The procedure to compute the smoothened pixels using the concept of neighbour pixel similarity is shown in Fig. 3 where pixels of each row is approximate by its diagonal pixel value and the row coefficient of the kernel are added to achieve the desired multiplier. This constant multiplier is then multiplier with the approximated pixel to achieve the smoothened pixel.

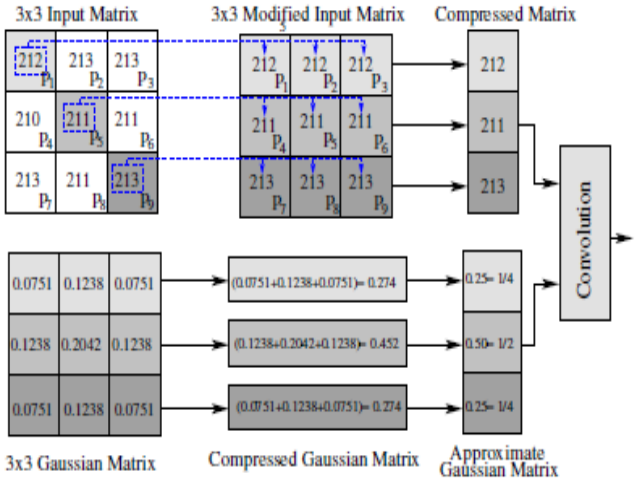


Fig-3: GSF process exploiting NPS.

3.4 Approximate GF using Approximate Adders

An approximate Gaussian filter is proposed in [8] where approximate adders are embedded in the accurate GF architecture to reduce the area, power and delay. In the proposed approximate GF, different bit-width adders are explored to achieve power quality trade-off. The kernel coefficients to design approximate filter is given by Eq. 7. These coefficients are achieved with σ is equal to 1.4.

$$\begin{bmatrix} A1 & A2 & A3 & A4 & A5 \\ A6 & A7 & A8 & A9 & A10 \\ A11 & A12 & A13 & A14 & A15 \\ A16 & A17 & A18 & A19 & A20 \\ A21 & A22 & A23 & A24 & A25 \end{bmatrix} * 1/159 \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (7)$$

The approximate adders such as error tolerant adder (ETA-1) are utilized to achieve energy efficiency. The

resulting architecture of the approximate GF is shown in Fig. 4. In the figure, the value of k representing the number of bits of adder where approximate logic is applied to compute the sum. This architecture requires small number of adders which are further approximated to reduce the implementation complexity. The author implemented an edge detector to evaluate the efficacy in the application. Further, different bit-width approximations are considered to achieve improved design and quality metrics.

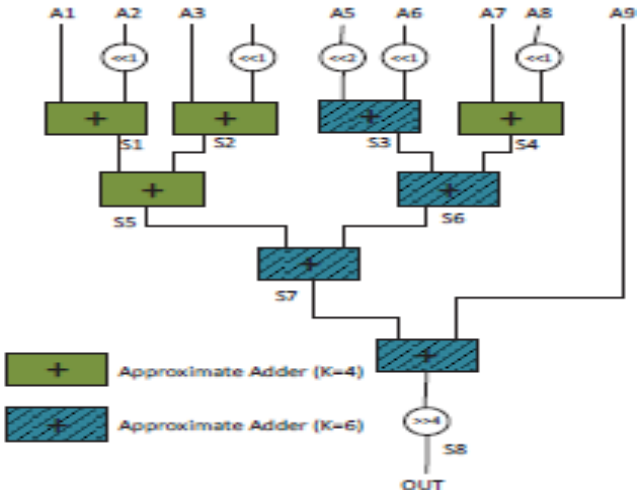


Fig. 4: 3x3 Gaussian filter architecture.

4. RECONFIGURABLE GAUSSIAN FILTER DESIGN

Due to varied requirement of the applications, reconfigurable designs are the critical requirement and the existing approximate architecture fails to exhibit large applicability. Significant efforts have been given to achieve reconfigurable architectures. These architectures exploit the concept of significant/non-significant coefficients and presented significant and non-significant boundaries as shown in Fig. 5 [7]. It observed that value of coefficient decrease from center to the boundaries therefore, boundary B1 is more significant (due to having more weight) over the B4. The ES-GSF exploits the non-significant boundaries to achieve desired quality energy trade-off. Further, the coefficient of the given boundary is of same value, the resulting architecture will compute sum of all pixel of that boundary and multiplied with the coefficient.

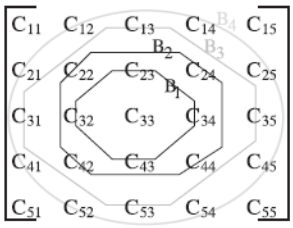


Fig. 5: Kernel coefficient with boundaries.

This architecture computes the smoothened pixels by computing only the significant boundary based on the given quality requirement and energy budget. The architecture compute approximate kernel (by decimating the non-significant boundaries) based on the given energy budget as shown in Fig. 6. Since the kernel considers the significant neighbour pixel in the computation, it provides sufficient output image quality. It can be observed from Fig. 6 that energy scalable GSF (ES-GSF) architecture computes value of different boundaries which are getting adder with the significant boundaries. In the figure, W0 and W1 provide smoothened pixel via only boundary B1. Another boundary B2 is added with this to generate smoothened pixel of higher quality at the cost of increased complexity.

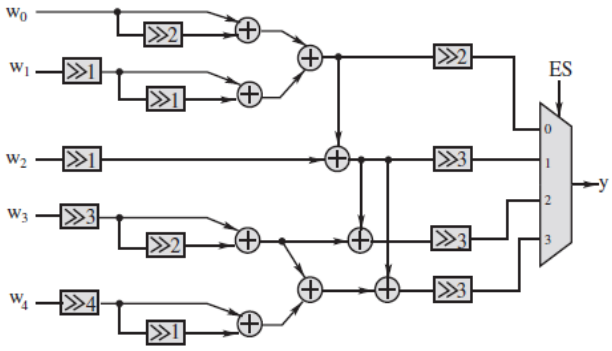


Fig-6: ES-GSF architecture.

In the similar way non-significant boundaries are added with the significant to improve the quality at the cost of computational complexity. Further, the increasing complexity increases the delay of computation and therefore it can be observed from the figure that delay computing only single boundary is two adders while it increases by one adder delay for considering each additional boundary.

5. EXPERIMENTAL RESULT & ANALYSIS

All the existing design are implemented on MATLAB and simulated with benchmark images [10], [11] to extract quality metrics. Further, these designs are implemented on Tanner. From the schematic, the spice netlists are extracted and simulated with 32nm technology files. The design metrics such as area, power and delay are extracted and compared. The area is measure in terms of number of transistors while the delay and power metrics are measure as absolute value.

5.1 Quality Metrics

Several quality metrics [12] are used to evaluate the design and are discussed in the following subsections.

5.1.1 Mean Square Error (MSE)

The MSE for an input image I and noisy output image K, is defined by the expression given below.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - k(i,j)]^2 \qquad (8)$$

where, variables m and n represent the number of row and column of the image.

5.1.2 Mean Error Distance (MED)

MED is the average error and is another name of mean error. Therefore, MED is equal to mean error.

5.1.3 Normalized Error Distance (NED)

NED is the MED divided by the maximum value of original signal. Value of NED is independent of the size of the design and depends only of the kind of architecture. Therefore, NED quantify the error metrics of the technique better than the MED.

5.1.4 Peak Signal to Noise Ratio (PSNR)

The PSNR is the parameter used widely in image/video processing applications to quantify the amount of the noise present in the image and it is equal to the maximum signal power to the noise power. The expression that computes the PSNR in decibel is given by the equation below.

$$PSNR_{db} = 10.\log_{10}\left(\frac{sig_i^2}{MSE}\right) \qquad (9)$$

Where, s_ig_i reflects the maximum signal value which for an image is 255.

5.1.5 Structural Similarity (SSIM)

In an image, the noise may be perceivable or it may not. If the noise is not perceivable noise will not affect the quality of the image. In order to compute the quality of the image i.e. perceivable errors present in the image a new quality metrics based on structural similarity is used [13]. This quality metrics is becoming more popular in recent years. Although, the SSIM better quantify the quality of the image, the commonly used parameter is the PSNR.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \qquad (10)$$

5.2 Simulation results on MATLAB

All the existing and proposed Gaussian filter architectures are implemented on MATLAB and simulated with benchmark image. The simulation results as shown in Table 1 illustrates the quality metrics namely MED, NED, MSE, PSNR and SSIM.

Table-1: Comparison of quality metrics.

Parameter	MED	MSE	NED	PSNR	SSIM
Acc_GSF	13.032	272.13	0.1489	23.78	0.8707
MA_GSF	13.61	296.88	0.1487	23.4	0.857
BG_GSF	14.59	348.93	0.1327	22.7	0.781
ES-GSF	18.2	270.78	0.168	23.8	0.8739
LASCAS	96.85	12557	96.85	7.14	0.0075

It can be observed from the Table 1 that MA-GSF provides least quality whereas BG-GSF provides higher quality in terms of PSNR as shown in Fig. 7.

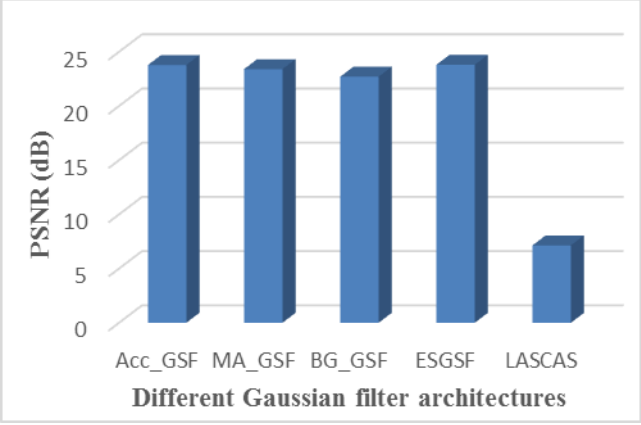


Fig-7: PSNR of filtered image using different GF.

Similarly, it can be observed that value of the SSIM is very poor for the LASCAS filter over the other existing filter architectures whereas the ES-GSF and Acc_GSF provides higher quality over BG_GSF.

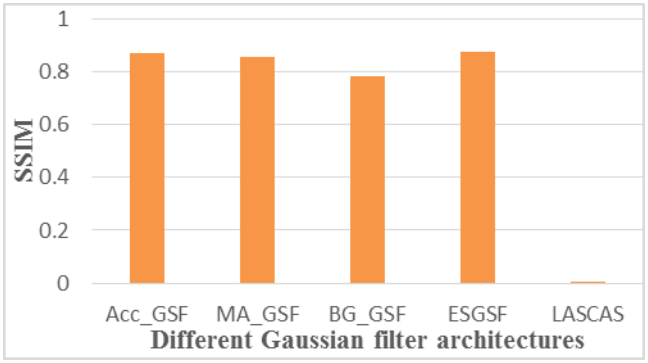


Fig-8: SSIM of filtered image using different GF.

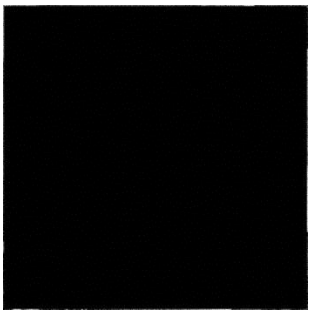
(a) Noisy image

(b) Acc_GSF

(c) MA_GSF

(d) BG_GSF

(e) ES_GSF



(f) LASCAS-GSF

Fig-9: Reconstructed filter images using various Gaussian filter architectures.

5.3 Simulation Results on Tanner

All designs are implemented on Tanner v14.1 and spice netlist is generated from the schematics implemented on Tanner. The design metrics such are area, power and delay are extracted for each design and are compared to evaluate the effectiveness of one architecture over the other. The power and delay metrics for the different GSFs are shown in Table 2.

Table-2: Design metrics for different GSF.

Design	Area (#Tran)	Power (µw)	Delay (ns)	Energy (fJ)
MA-GSF	5264	110.3	3.64	401.5
GF_LASCAS	5740	1050	4.38	4599
ES-GSF	5812	90.1	1.73	155.8

It can be observed from the Table-2 that ES-GSF consume smallest energy over the other existing filter architectures. The area is measured in terms of number of transistors whereas power and delay metrics are computed as absolute values. To compute the energy consumption, power-delay product is computed. From these simulation results it can be seen that ES-GSF design consume smallest energy over the other filter architectures as shown in Fig. 10.

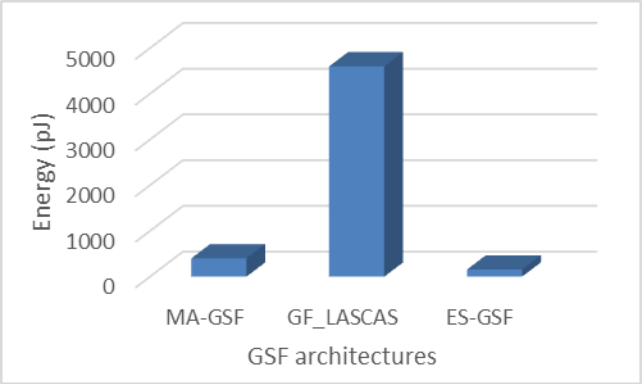


Fig-10: Energy consumption of various GSFs.

6. CONCLUSION

This paper presents an exhaustive analysis on different energy efficient Gaussian filters architecture that provides process image of acceptable quality. To evaluate the effectiveness of the different designs, all the filters architectures are implemented in MATLAB and Tanner. The designs on the MATLAB are simulated with benchmark input images and corresponding scaled image and quality metrics are extracted. The designs implemented on the Tanner's schematic editor and then spice netlists are extracted. These netlists are simulated with benchmark inputs. The simulation results show that the BG-GSF design exhibits minimum energy over the other design.

REFERENCES

[1]. V. S. S. Prasad, J. Mathews, and N. Naganathan, "Low-power design strategies for mobile computing, in VLSI Design, 19th Int. Conf. on, pp. 2, Jan 2006.

[2]. J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8 (November 1986) 679–698.

[3]. D. Marr, E. Hildreth, Theory of edge detection, Proc. R. Soc. Lond. 207 (Jan 1980) 1167.

[4]. S. Khorbotly, F. Hassan, A modified approximation of 2D Gaussian smoothing filters for fixed-point platforms, in: IEEE 43rd Southeastern Symposium on Sys- tem Theory (SSST), March 2011, pp. 151–159.

[5]. P.Y. Hsiao, C.H. Chen, S.S. Chou, L.T. Li, S.J. Chen, A parameterizable digital- approximated 2D Gaussian smoothing filter for edge detection in noisy image, in: Proceedings IEEE International Symposium on Circuits and Systems, May 2006, 4, pp. 3189–3192.

[6]. Jaiswal, B. Garg, V. Kaushal, G. Sharma, SPAA-aware 2D Gaussian smoothing filter design using efficient approximation techniques, in: Proceedings of 2015 28th International Conference on VLSI Design (VLSID), 2015, pp. 333–338. IEEE.

[7]. Garg, Bharat, and G. K. Sharma "A quality-aware Energy-scalable Gaussian Smoothing Filter for image processing applications" Microprocessors and Microsystems (2016).

[8]. Cabello, Frank, Julio León, Yuzo Iano, and Rangel Arthur. "Implementation of a fixed-point 2D Gaussian Filter for Image Processing based on FPGA." In Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2015, pp. 28-33. IEEE, 2015.

[9]. De Oliveira, J., Soares, L., Costa, E., & Bampi, S. (2016, February). Exploiting approximate adder circuits for power-efficient Gaussian and Gradient filters for Canny edge detector algorithm. In Circuits & Systems (LASCAS), IEEE 7th Latin American Symposium on, pp. 379-382, 2016.

[10]. Standard Test Images from Kodak. http://www.r0k.us/graphics/kodak/(accessed12.03.16).

[11]. Benchmark Inputs for Image Processing. http://www.imageprocessingplace.com(accessed12.03.16).

[12]. J. Liang, J. Han, F. Lombardi, New metrics for the reliability of approximate and probabilistic adders, Computer IEEE Trans. 62 (2011) 1760–1771.