VLSI Implementation of Energy Efficient Gaussian Filter: A Survey

Shalini Upadhyaya[1] and Shweta Agrawal[2]
[1]Research scholar,[2] Assistant Professor,
Dept of Electronics and Comm., SRCEM Banmore, Morena, India

***Abstract*— In the modern VLSI design for the portable gadgets, the prime concern is the power and performance. Filter is the most commonly used processing block to filter out the noise entered in the image/video at different level of processing. The Gaussian filter is commonly used to remove noise while maintains the structure of the object in the scene. The high computation complexity is the challenge to the VLSI designers. In this paper, an exhaustive literature review is done and then the performance of the existing designs is evaluated and compared. These existing designs are implemented and simulated with benchmark input to compute the efficacy over the existing architectures. The designs are modeled on MATLAB and Tanner, simulated with benchmark inputs and then quality and design metrics are evaluated where the results show that filters provide images of acceptable quality with different amount of error.**

***Keywords*— Digital Signal Processing (DSP), Median Filter, Image Processing, Integrated Circuits, VLSI, Low Power Design.**

I. INTRODUCTION

Last few decades shows significant development in the VLSI Technology which results in large number of electronic gadgets on the user hand. These gadgets are not only able to perform different task but also very efficiently. The growing number of functions on these gadgets demands VLSI architectures/design which processing signal very efficiently which ultimately increases the complexity of the designs. The complexity of today's design is very high which result in high power and delay in the present devices and it is growing with increasing functionality on the same device. The conventional approaches of scaling to improve the performance of these design is the device scaling. The scaling is approach has reached to its level and the devices cannot be scaled further due to increased effect of process and other variations.

The process variation has become so severe in the today sub-nanometre designs that designs without considering it will fail to provide desired output. Further, addition circuit to mitigate the effect of process variation is very costly in terms of power, area and delay such that gain due scaling are less than overhead. Therefore, other design methodology is required to develop designs for the modern gadgets. There are several applications where the approximate results are acceptable such as image/video processing. The relaxation on the accuracy can be exploited to reduce the complexity of the designs.

The most modern devices employ multimedia applications that produce output for human consumption [1]. Due to the limited visual perception, human can accept errors. Moreover, the noise in the present electronic devices can come from anywhere e.g. while transmission, storage etc. [2], [3]. In order to remove the effect of the error, filter is required [4]. Most commonly employed filter is the smoothing filter. Different algorithms, architectures are developed in the literature to efficiently filter the noisy input image [5]. The concept of coefficient approximation into power of two is presented such that resulting coefficients can be realized without any multiplication logic [6], [7]. Other approach utilizes the similarity existing in the neighbouring pixels.

The architectural approach where computation sharing is done is also demonstrated. In addition to these design, an energy scalable Gaussian smoothing filter (ESGSF) [8] is also presented which provides trade-off between quality and energy. The ES-GSG is based on the principle of consideration of varying size coefficient with respect to the central coefficient. The existing architectures still having higher complexity and must be reduced to fit the design within the modern portable devices.

The rest of the paper fist discusses different Gaussian filters and then implements and computes design and quality metrics to compare them.

II. Principle of Gaussian Filter

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

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

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. The graph showing the Gaussian distribution is shown in Figure 1.

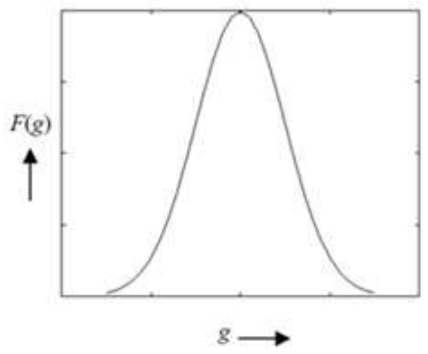


Figure 1: Gaussian distribution.

As the Gaussian smoothing is commonly employed in different image processing applications such as edge detection to remove unwanted edge, image mosaicking [4] and tone mapping etc., it requires two dimensional (2D) Gaussian expression due to the 2D nature of the image. The 2D Gaussian expression is represented by Equation 2.

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

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

In order to smooth the image, the image is processed 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 smoothing. From VLSI design point of view, direct implementation of the above expression is area and performance efficient. Therefore, the above expression can be approximated by a matrix called kernel. The image processed through the Gaussian kernel provides the same result as the above expression. The accurate representation of the Gaussian expression using Gaussian kernel of 5x5 size is shown in Figure 2.

$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.596 & 0.0983 & 0.0996 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.596 & 0.0983 & 0.0996 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix}$$

Figure 2: Normalized coefficients of a 5x5 Gaussian kernel with σ=1.

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. Following subsection details different architectures developed for the energy efficient smoothing.

III. Approximate Gaussian Filter Design

This section provides different energy efficient kernels, given that can be used to efficiently compute the smoothened pixel.

3.1 Fixed-point Gaussian Kernel

A fixed point implementation of Gaussian coefficient [5] is used to achieve approximate kernel. 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 $X_3X_2X_1X_0$ in a (4,3) format is $X_3.2^6+X_2.2^5+X_1.2^4+X_0.2^3$ While the decimal equivalent of the same coefficient in a (4, -3) format is $X_3.2^0+X_2.2^{-1}+X_1.2^{-2}+X_0.2^{-3}$ Fig. 3(a) shows the Gaussian smoothing kernel rounded to the (2, -4) and (6, -8) data formats.

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

(a)

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

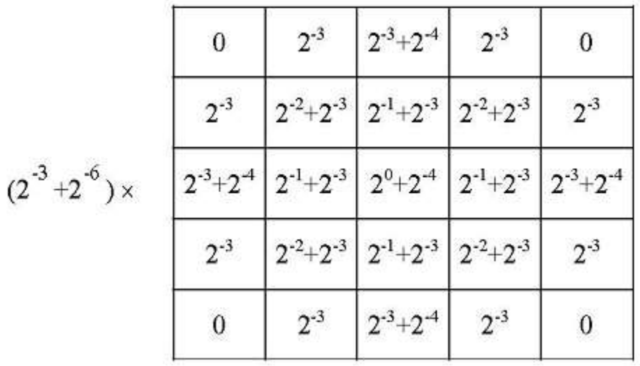
(b)

Figure 3: The Gaussian smoothing kernel (a) data format (2, -4) (b) data format (6, -8).

These kernels reduce implementation complexity of the Gaussian 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 Parameterizable Digital Approximated 2D Gaussian Smoothing Filter

The computation complexity of the floating point multiplier is very high which makes the direct implementation of the 2D Gaussian smoothing kernel energy inefficient. A digital approximation of the kernel coefficient presented in [6] that significantly reduces the implementation complexity of the Gaussian kernel. 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 any hardware for implementation. The proposed Gaussian kernel with value in power-of-two is shown in Figure 4.



$(2^{-3}+2^{-6})\times$

0	2^{-3}	$2^{-3}+2^{-4}$	2^{-3}	0
2^{-3}	$2^{-2}+2^{-3}$	$2^{-1}+2^{-3}$	$2^{-2}+2^{-3}$	2^{-3}
$2^{-3}+2^{-4}$	$2^{-1}+2^{-3}$	$2^{0}+2^{-4}$	$2^{-1}+2^{-3}$	$2^{-3}+2^{-4}$
2^{-3}	$2^{-2}+2^{-3}$	$2^{-1}+2^{-3}$	$2^{-2}+2^{-3}$	2^{-3}
0	2^{-3}	$2^{-3}+2^{-4}$	2^{-3}	0

Figure 4: Simplified kernel coefficients in power-of-two form.

A simplified architecture is developed that utilized the kernel coefficients in the power-of-two form and compute the smoothened pixel. The architecture is shown in Figure 5.

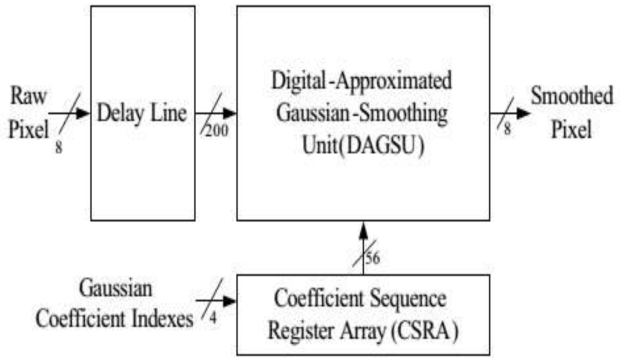

Raw Pixel
8
Delay Line
200
Digital-Approximated Gaussian-Smoothing Unit(DAGSU)
8
Smoothed Pixel
56
Gaussian Coefficient Indexes
4
Coefficient Sequence Register Array (CSRA)

Figure 5: Parameterizable 2D Gaussian smoothing filter architecture.

3.4 SPAA-Aware 2D Gaussian Smoothing Filter

The higher computational complexity of the existing smoothing filter limits the energy-efficiency which can be improved by exploiting the property of image. A speed, power area and accuracy (SPAA) aware 2D Gaussian smoothing filter is demonstrated in [7] that exploits the neighbour pixel similarity existing within an image. This architecture significantly reduces the implementation complexity and exhibits significantly improved design metrics. The following subsection reveals the basic principle to reduce the complexity and architectural approach considered.

3.4.1 Neighbour Pixel Similarity

In an image, the adjacent pixels exhibit higher correlation [7] i.e. their values are nearly same. This higher correlation in the image is due to the fact that image have smoothened nature and the large difference will occur only on the edge. 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 if applied to this sub-matrix will significantly reduce the computation complexity of the smoothening filter. For example, an original sub-matrix of size 3x3 as shown in Figure 6 is replaces all pixels of a row by its single value. This replacement will not cause large error the values are very near.

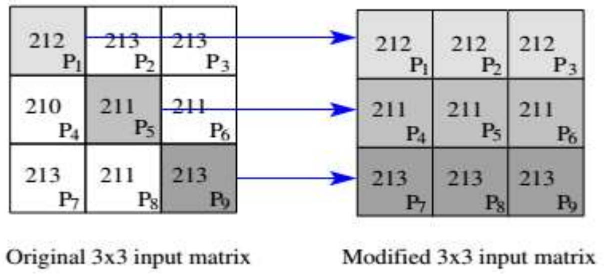


Figure 6: Neighbour pixel similarity in 3x3 image sub-matrix.

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. This procedure is shown in Figure 7 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. From the figure can be observed that it requires only five multipliers with few adders to added the coefficients.

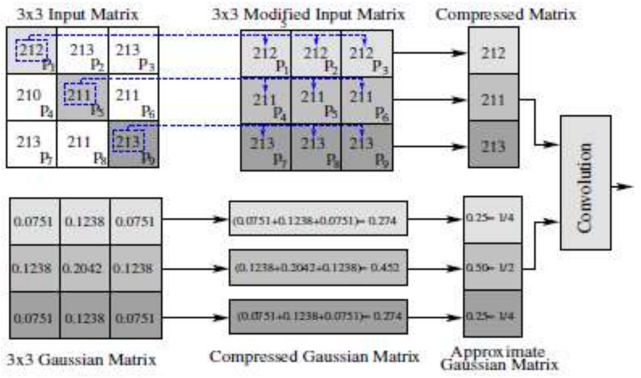


Figure 7: Approximate GSF process using 3x3 image sub-matrix.

IIV. Energy Scalable Gaussian Filter Design

The ever changing requirement of the real-time applications demands configurable smoothing filter is high as the existing designs provides output with fixed energy-quality budget. Based on the applications and its criticalness, the design should be able to adapt different energy-quality trade-off which is achieve by the energy scalable GSF architecture [8]. This architecture exploits the concept of significant/non-significant coefficients and presented significant and non-significant boundaries as shown in Figure 8. It observed that value of coefficient decrease from centre to the boundaries therefore, boundary B_1 is more significant (due to having more weight) over the B_4. 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.

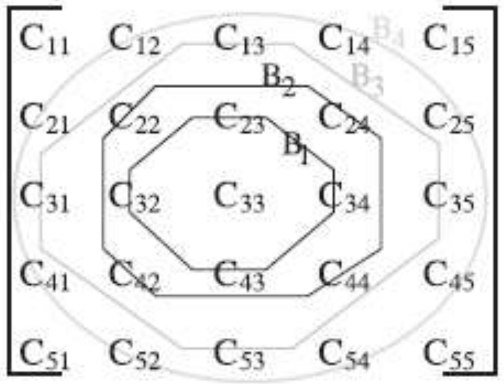


Figure 8: Kernel coefficient with boundaries.

The algorithm accepts image and energy-scalability as input while provide smoothened image as output. In this algorithm, the input image is sub-divided into 5x5 image sub-matrix and these sub-matrices are processed. The algorithm compute approximate kernel (by decimating the non-significant boundaries) based on the given energy budget. Since the kernel considers the significant neighbour pixel in the computation, it provides sufficient output image quality. The architecture that implement the functionality presented in the algorithm is shown in Figure 9.

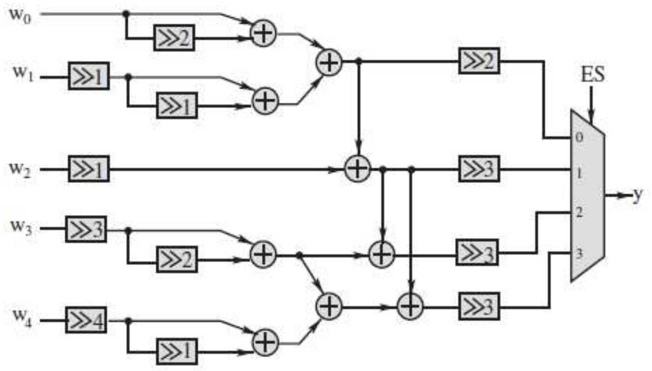


Figure 9: Energy scalable GSF architecture.

It can be observed from Figure 9 that energy scalable GSF architecture computes value of different boundaries which are getting adder with the significant boundaries. For example, W_0 and W_1 both providing smoothing pixel via only boundary B_1. Another boundary B_2 is added with this to generate smoothened pixel of higher quality at the cost of increased complexity. 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. Therefore, ES-GSF provides trade-off between performance and quality.

IV. EXPERIMENTAL RESULT & ANALYSIS

In order to evaluate the quality metrics [9], [10], MATLAB tool is to model the existing architectures of the Gaussian filters. These implemented designs on MATLAB are then simulated with standard test images [11], [12] such as Lena and Baboon as shown in Figure 10. The scaled images are extracted and compared. On the other hand, to evaluate the design metrics designs are implemented on Tanner schematic editor. Finally, the design metrics such as area, power and delay are extracted for the proposed and existing designs and compared.

Figure 10: Benchmark images

4.1 Quality and Design Metrics

Various quality and design parameters are used to evaluate the design. This subsection introduces the different quality parameters which are used in our design.

4.1.1 Mean Square Error (MSE)

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

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - k(i,j)]^2 \quad (3)$$

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

4.1.2 Mean Error Distance (MED)

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

4.1.3 Normalized Error Distance (NED)

Normalize error distance is the mean error distance 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.

4.1.4 Peak Signal to Noise Ratio (PSNR)

The peak-signal-to-noise-ratio 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 mathematical expression that computes the PSNR in decibel is given by the equation below.

$$PSNR_{db} = 10.\log_{10}(\frac{Sig_I^2}{MSE}) \quad (4)$$

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

4.1.5 Structural Similarity (SSIM)

The existing quality metrics signifies the quantity of error in the given input signal. 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 (SSIM) is used. This quality metrics is becoming more popular in recent years.

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

Although, the SSIM better quantify the quality of the image, the more commonly used parameter is the PSNR.

4.2 Simulation results on MATLAB

To evaluate the quality metrics of the proposed median filter, the proposed and existing filter architectures are implemented on the MATLAB and simulated with benchmark input image. Mean error distance, normalized error distance, PSNR and SSIM are computed for each Gaussian filter as shown in Table 5.1.

Table 5.1: Error metrics of different GSF.

Parameter	MED	MSE	NED	PSNR	SSIM
Noisy	14.21	316.21	0.1755	22.59	0.7822
Acc_GSF	13.032	273.02	0.1484	23.66	0.8715
MA_GSF	13.61	297.82	0.1478	23.28	0.8579
BG_GSF	14.59	348.99	0.1373	25.34	0.9565
ES_GSF	11.49	207.04	0.1507	24.85	0.9509

It can be observed from the Table 5.1 that MA-GSF provides least quality whereas BG_GSF provides higher quality in terms of PSNR as shown in Figure 11.

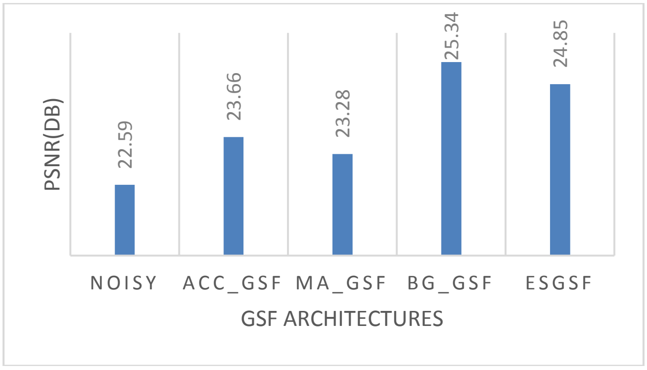


Figure 11: PSNR for different GSF architectures

Similarly, Figure 12 shows that BG_GSF exhibits higher SSIM over the others.

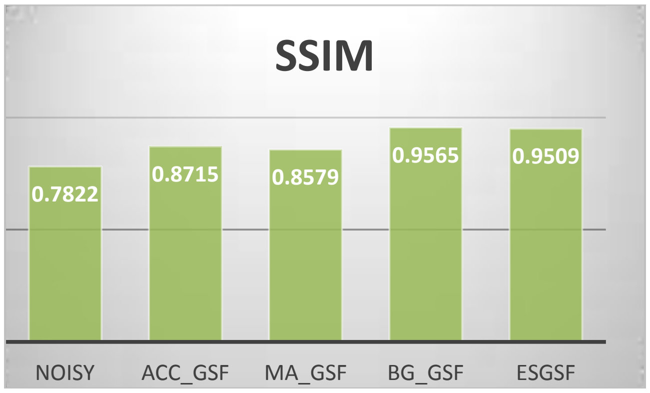
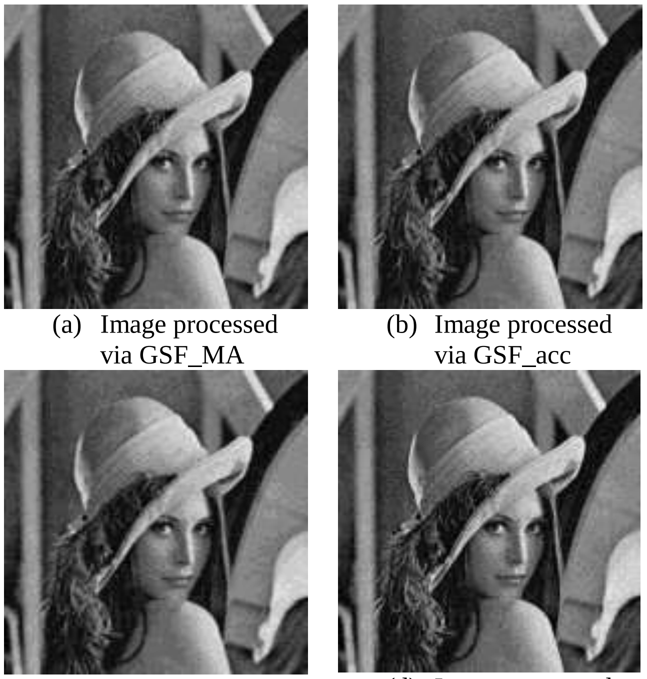


Figure 12: SSIM for different GSF architectures

Finally, the processed image using different Gaussian filters as shown in Figure 13 reflect that images quality of smoothened images is of acceptable quality.



(a) Image processed via GSF_MA

(b) Image processed via GSF_acc

(c) Image processed via GSF_bg

(d) Image processed via ES_GSF

Figure 13: Lena image filter via different GSF filters.

4.3 Simulation Results on Tanner

To evaluate the design metrics all designs are implemented on Tanner v14.1. The 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 GSF architecture are shown in Table 5.2.

Table 5.2: Design metrics for different GSF.

Design	Area (#Tran)	Power (µw)	Delay (ns)	Energy (fJ)
MA-GSF	5264	0.8688	2.52	2.189
BG-GSF	1288	0.12144	1.446	0.175
ES-GSF	5812	0.901	1.73	1.55

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 BG_GSF design provides very small delay over all the existing architectures. Further power consumption is comparable to the other architecture. Finally, the energy efficiency of the BG_GSF is high over the all the existing GSFs as shown in Figure 14.

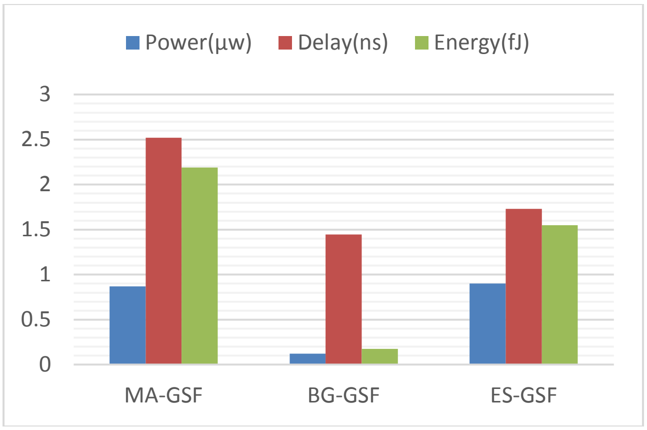


Figure 14: Design metrics comparison.

V. 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 requirement over the other design.

REFERENCES

[1] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, K. Roy, IMPACT: Imprecise adders for low-power approximate computing, in: Proceedings of 2011 Inter- national Symposium on Low Power Electronics and Design (ISLPED), Aug. 2011, pp. 409–414.

[2] B. Garg, N.K. Bharadwaj, G. Sharma, Energy scalable approximate DCT architecture trading quality via boundary error-resiliency, in: Proceedings of 2014 27th IEEE International System-on-Chip Conference (SOCC), 2014, pp. 306–311. IEEE.

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

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

[5] 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.

[6] 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 ISCAS, May 2006, 4, pp. 3189–3192.

[7] A. 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 VL SI Design (VL SID), 2015, pp. 333–338. IEEE.

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

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

[10] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process 13 (April 2004) 600–612.

[11] Standard Test Images from Kodak. http://www.r0k.us/graphics/kodak/ (ac- cessed 12.03.16).

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