Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 2008–2015

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Efficient Hardware of RGB to Gray Conversion Realized on FPGA and ASIC

Kaushal Kumar[a,*], Ritesh Kumar Mishra[a], Durgesh Nandan[b]

[a]*Department of ECE, National Institute of Technology, Patna, India*
[b]*Aditya Engineering College, Surampalem, AP, India*

Abstract

RGB to gray conversion is an integral part of various computer vision applications such as face detection, object detection and surveillance systems. The resource required for the real time implementation of all these applications decreases to a great extent if computation is performed on gray images, which has 8 bit wide pixel, rather than color images, which has 24 bit wide pixel. In this paper, hardware efficient implementation of RGB to gray image is proposed which is realized on both FPGA and ASIC. FPGA realization is performed on digilent Zedboard having Artix-7 FPGA while the ASIC implementation is performed using Cadence Genus and Innovus tool at 45 nm process technology. ASIC implementation of proposed technique brings about total area utilization of 262 um^2 and ADP of 18.078 $um^2 * ns$ which are respectively 81.42% and 96.55% less contrasted with existing design. The proposed system is seen to operate at high frequency of 3 GHz.

Keywords: ASIC; Computer Vision; FPGA; Gray; Image Processing; RGB

1. Introduction

Image processing plays an important role in various computer vision applications such as remote sensing, surveillance systems, biomedical imaging, object detection, and its localization [1]-[9]. Real-time images are usually composed of three primary colors, i.e., red, green, and blue, popularly known as RGB. Other colors are produced by combining red, green, and blue colors. In various image processing applications, various processings are required to conduct on each pixel. It is not feasible to process RGB pixels because of the high computation complexity and storage requirement. To overcome such issues, the RGB images are first converted into gray images, and then the required processing is carried out. Gray image is a monochrome image consisting of only brightness information. Figure 1 provides a general block diagram of steps involved in the implementation of various computer vision ap-

* Corresponding author.
E-mail address: kaushal.ec16@nitp.ac.in

10.1016/j.procs.2020.04.215

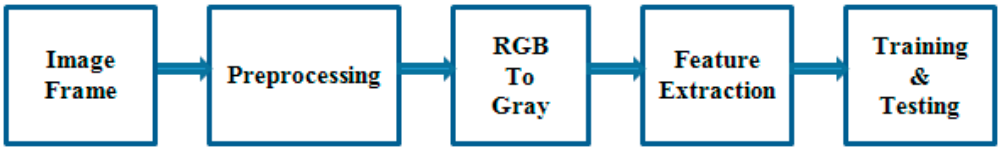

Fig. 1. Block diagram of steps involved in computer vision applications using image processing.

plications involving image processing. Since the computer vision applications work on the features extracted from images, the need for color images is not compulsory as it increases the computation cost. This enhances the importance of efficient RGB to gray conversion for real-time computer vision applications. In literature, various techniques are presented for translating color image into gray image [10][11][12]. Laszlo *et al.* presented techniques based on observations performed on the Coloroid system [13]. The method seems to conserve the appearance of a color image. Amy *et al.* presented a technique in which color image is first translated into uniform color space, the difference is then evaluated, and finally, least-square optimization is applied [14]. This technique seems to handle isoluminant color superbly. But the cost of the experimental setup and optimization is in proportion to image size. Also, resource utilization is quite high. Yongjin *et al.* presented a fast algorithm for RGB to gray conversion, which preserves the actual color lightness and color ordering [15]. It utilizes a nonlinear global mapping optimization technique. [16][17] provides the various techniques of gray to RGB conversion, which is required at the output stage. Ye *et al.* presented the software implementation of HSV (HUE, SATURATION., AND VALUE) quantization algorithm of gray scale conversion [18]. Grundland *et al.* proposed a technique which improves the contrast of image despite RGB to gray conversion [19]. It involves Gaussian pairing for image sampling and dimensionality reduction. Xiuyu *et al.* presented efficient gradient-based RGB to gray conversion having automatic optimization of parameters [20]. The parameters are optimized with respect to structural similarity index measurement (SSIM). Raveendran *et al.* presented a conversion technique based on reversible logic gates [21]. It implemented averaging and desaturation methods for RGB to gray conversion. The design is implemented on Kintex 7 FPGA. In brief, many authors proposed various techniques for RGB to gray conversion, but very few of them have discussed its hardware implementation.

For a real-time image processing system, there is a need for efficient hardware implementation of RGB to gray conversion algorithm to minimize overall area and power requirements. FPGA is suitable for real-time implementation of image processing applications because of its parallel processing capabilities. In this paper, we present the hardware efficient FPGA and ASIC implementation of color to gray level conversion. ASIC implementation is performed using 45 nm process technology. FPGA prototyping of RGB to gray conversion is performed on images of different resolutions. The rest of the paper is organized as follows: section II presents the proposed architecture for the efficient implementation of RGB to gray conversion, while section III provides a discussion on the obtained results. Section IV concludes the paper highlighting important findings.

2. Proposed Architecture

A digital color image is composed of a 24-bit wide pixel. Out of 24 bits, the first 8 bits from MSB are specified for red color, next 8 bits for the green color, and the last 8 bits for blue color. In a gray image, each pixel is composed of only 8 bits, which provides only brightness information and no information about color. Various algorithms exist to accomplish RGB to gray conversion such as average method, luminosity or weighted method, desaturation method and decomposition method. Among them, the widely used and standard algorithm adopted by various image processing software and MATLAB ”rgb2gray” function is the luminosity method [22]-[24]. This algorithm is based on human eye brightness perception, which is strongest for green followed by red and then blue. Hence, the luminosity algorithm is adopted in this paper for RGB to gray conversion, given by (1).

$$Y = 0.281R + 0.562G + 0.093B \tag{1}$$

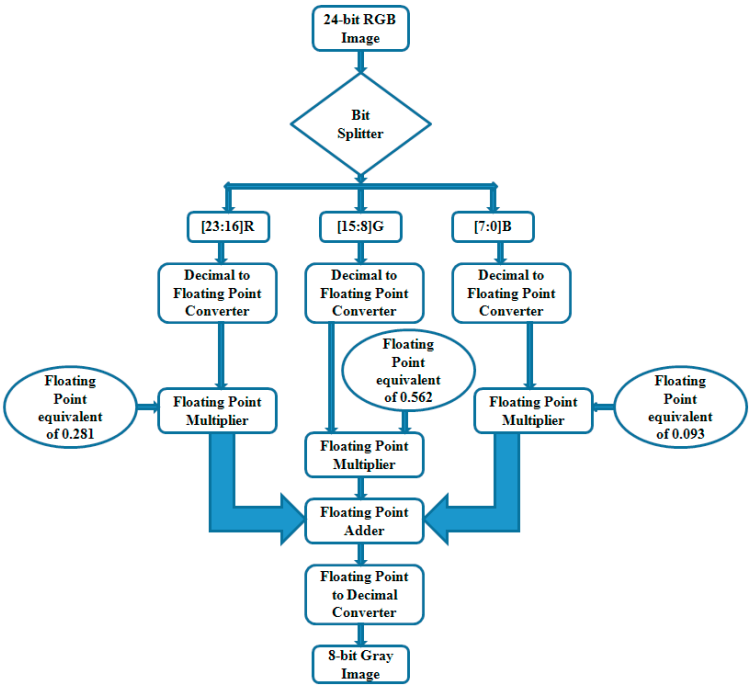

Fig. 2. Existing technique of RGB to Gray conversion.

Where R, G and B represent 8 bit red, green and blue values respectively and Y represents the equivalent gray value of RGB pixel. From (1), it can be inferred that the gray value of pixel is composed of 28.1% of red, 56.2% of green and 9.3% of blue. However, in luminosity algorithm, the grayscale value is calculated by summing 0.299 of R, 0.587 of G, and 0.114 of B. For simplicity in hardware implementation without significant loss of information, the coefficients are modified as shown in (1). The existing architecture of RGB to gray conversion is shown in figure 2. It can be inferred that various floating-point computational units are required to perform gray scale conversion, which leads to an increase in resource utilization. In the proposed architecture, shift registers are used to get the required fractions of red, green and blue pixels which increases the design efficiency in terms of area, power and speed as shown in figure 3. The step by step process of the proposed architecture of RGB to gray conversion is explained below:

Algorithm:

Step 1	Take a 24 bit pixel from color image.
Step 2	Split the 24 bit pixel into R, G and B. First 8 bits are assigned to B, next 8 bits to G and the last 8 bits to R.
Step 3	Using right shift operator, shift 8 bits of R in right to obtain 28.1% of red. Similarly, right shift 8 bits of green and blue to obtain 56.2% of green and 9.3% of blue.
Step 4	The results obtained after right shift operations are finally added using ripple carry adder (RCA) which is 8 bit adder.
Step 5	The final 8 bit result obtained from RCA represents the gray pixel.
Step 6	Repeat step 1 to 5 for all the pixels present in the image.

For better understanding of the approach followed for RGB to gray conversion, an example is shown below:

Let us consider a 24 bit pixel of color image given as A=24'b110101101111111001011100. After splitting the 24 bits among R, G and B, we get

R= 8'b11010110, G=8'b11111110, B=8'b01011100 R1=R>>2=8'b00110101, R2=R>>5=8'b00000110

G1=G>>1=8'b01111111, G2=G>>4=8'b00001111

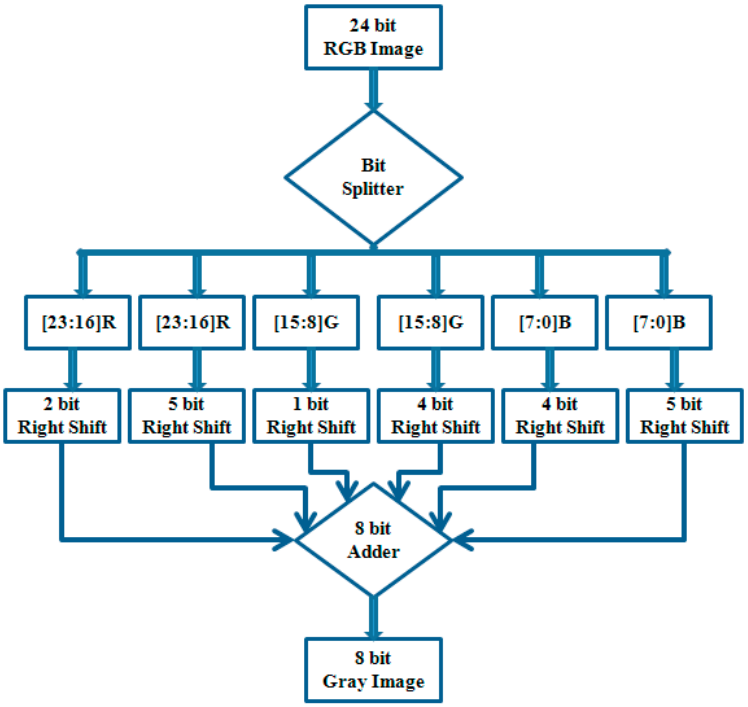

Fig. 3. Proposed technique of RGB to Gray conversion.

Table 1. Comparison of the proposed RGB to gray conversion architecture with existing architecture on FPGA.

Resource	Proposed	Existing
LUT	22	435
Flip Flop	8	56
IO	27	33
BUFG	1	1

B1=B>>4=8'b00000101, B2=B>>5=8'b00000010
The gray value is given by,
Y= R1 + R2 + G1+ G2 + B1 +B2 =8'b011010000.

3. Results And Discussions

The proposed architecture of RGB to gray conversion is implemented using Xilinx Vivado 2016.2 tool. The design is implemented on Digilent Zedboard (xc7z020clg484-1), having Artix-7 FPGA. Verilog hardware description language (HDL) is used for creating proposed architecture. Table 1 provides the post-implementation FPGA resource utilization for RGB to gray conversion module and its comparison with the existing architecture. It can be observed that there is a reduction of 94% and 85.71% in LUT and Flip Flop requirements, respectively, for the proposed architecture. However, Table 2 provides the resource utilization for a complete system of color to gray conversion, which also includes the VGA module for the VGA display. It can be observed that 20 more LUTs and 58 more FFs are required for displaying the output on the VGA display.

ASIC implementation is performed using Cadence Genus and Innovus tools at 45 nm process technology. It can be observed that the area delay product (ADP) for the proposed architecture is quite less, i.e., 18.078 $um^2 * ns$ in comparison to the existing framework, i.e., 523.674 $um^2 * ns$. Arrival time is the time taken by data signal to arrive at

Table 2. Resource utilization of the complete system of RGB to gray conversion on FPGA.

Resource	Utilization	Available	Utilization%
LUT	42	53200	0.08
Flip Flop	66	106400	0.06
IO	15	200	7.50
BUFG	1	32	3.13

Table 3. ASIC implementation of the proposed architecture at 45 nm process technology..

Parameters	Proposed	Existing
Total Area (um^2)	262	1410
Total Power (μW)	110.4	597.3
Arrival Time (ps)	69	371.4
ADP ($\mu m^2 * ns$)	18.078	523.674

destination flip flop. It is used in the determination of the frequency of operation of the system. It should be minimum along with non-negative slack, and the arrival time is found to be 69 ps for the proposed framework, which is quite less in comparison to 371.4 ps for existing architecture. Slack is defined as the difference between data required time and data arrival time. Figure 4 provides the variation of slack time with respect to frequency for the proposed architecture. For the proper functioning of the system, the slack time should be positive or zero. Negative slack can result in improper output. For the existing architecture, a slack of 8728 ps resulted at 20 MHz clock frequency, which is reduced to 23 ps at 25 MHz and 1 ps at 28 MHz. With further increase of clock frequency, zero slack is achieved at 29 MHz, which limits the speed of operation of existing architecture up to 29 MHz. However, for the proposed framework, at a low frequency of approximately 100 MHz, a slack of 9798 ps is observed. As the frequency increases, the slack time decreases. It can be observed that when frequency crosses 1 GHz, a drastic reduction in slack timing can be observed. It is also found that as the frequency approaches 3 GHz, the slack time approaches to zero. In this way, the proposed architecture of RGB to gray conversion can work at exceptionally fast rather than the existing design, which works till 29 MHz.

The validation of the proposed color to gray conversion is performed using four test images from different datasets, i.e., Caltech 101[25], Cadik[26], Color250[27], INRIA[28]. Figure 5 provides the results of the proposed framework for test images, and it can be observed that faithful conversion is obtained. Figure 6 shows the FPGA implementation of the proposed RGB to gray conversion, where the output is displayed on 1024×768 resolution VGA display working at 60Hz.

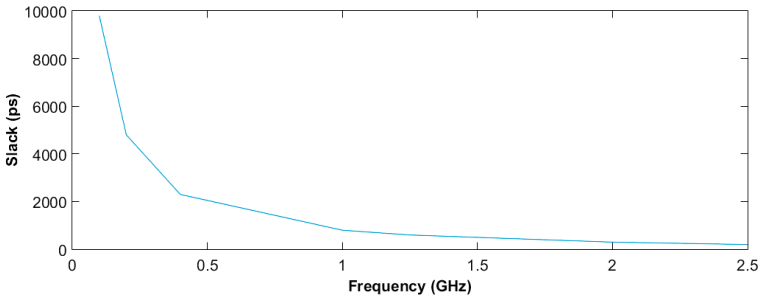

Fig. 4. Variation of slack with frequency for the proposed architecture.

Fig. 5. Outcome of the proposed hardware architecture using images from different datasets[25]-[28].

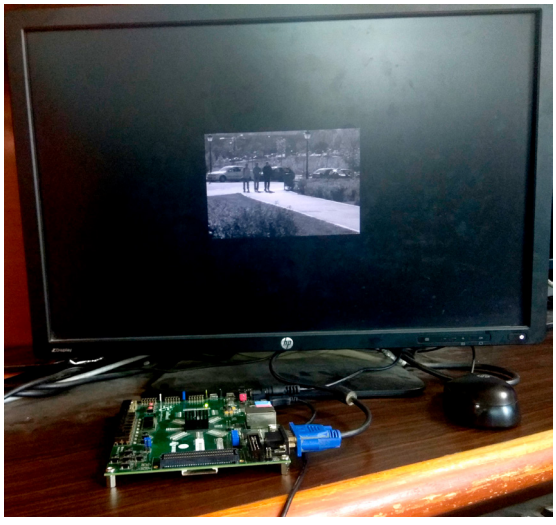

Fig. 6. FPGA implementation of RGB to gray conversion.

4. Conclusion

This paper proposes the efficient hardware architecture of color to gray image conversion. The proposed architecture is implemented on both FPGA and ASIC and is tested with real-time images. The obtained results are found to provide a faithful conversion of color images into gray images. The proposed architecture is realized using Verilog HDL and implemented on Digilent Zedboard, which consists of Artix 7 FPGA. The results of the prototype system are obtained at 1024×768 resolution VGA display. The ASIC implementation is done at 45 nm process technology. The proposed architecture is found to consume resources very efficiently and thus can be employed in various real-time computer vision applications.

References

[1] Kryjak, T., and Gorgon, M. (2011) "Real-Time Implementation Of Moving Object Detection In Video Surveillance Systems Using Fpga." Computer Science 12 (1): 149–162. doi:10.7494/csci.2011.12.0.149.

[2] Chan, S.C., Zhang, S., Wu, J., Tan, H., Ni, J.Q., and Hung, Y.S. (2013) "On the hardware/software design and implementation of a high definition multiview video surveillance system." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 3 (2): 248–262. doi:10.1109/ JETCAS.2013.2256822.

[3] Cahyono, D., and Nugroho, A. (2013) "Design and realization camera controller for a remote sensing payload of nanosatellite fpga (field programmable gate array) system based," In: 2013 6th International Conference on Recent Advances in Space Technologies (RAST), pp. 73–77. doi:10.1109/RAST.2013.6581306.

[4] Dillinger, P., Vogelbruch, J.F., Leinen, J., Suslov, S., Patzak, R., Winkler, H., and Schwan, K. (2005) "FPGA based real-time image segmentation for medical systems and data processing", In: 2005 14TH IEEE-NPSS Real Time Conference, pp. 161–165. doi:10.1109/RTC.2005.1547401.

[5] Kyrkou, C., and Theocharides, T. (2012) "A parallel hardware architecture for real-time object detection with support vector machines." IEEE Transactions on Computers 61 (6): 831–842. doi:10.1109/TC.2011.113.

[6] Zhao, P., Zhu, H., Li, H., and Shibata, T. (2013) "A directional-edge-based real-time object tracking system employing multiple candidate-location generation." IEEE Transactions on Circuits and Systems for Video Technology 23 (3): 503–517. doi:10.1109/TCSVT.2012.2210665.

[7] Kumar, K., and Mishra, R.K. (2019) "A Robust mRMR Based Pedestrian Detection Approach Using Shape Descriptor." Traitement du Signal 36 (1): 79-85. doi:10.18280/ts.360110

[8] Salamh, A.B.S. (2017) "Investigation the Effect of Using Gray Level and RGB Channels on Brain Tumor Image", In: Computer Science & Information Technology (CS & IT), pp. 141–148. doi:10.5121/csit.2017.71012.

[9] Padmavathi, K., and Thangadurai, K. (2016) "Implementation of RGB and grayscale images in plant leaves disease detection - Comparative study." Indian Journal of Science and Technology 9 (6). doi:10.17485/ijst/2016/v9i6/77739.

[10] Kumar, T., and Verma, K. (2010) "A Theory Based on Conversion of RGB image to Gray image." International Journal of Computer Applications 7 (2): 5–12. doi:10.5120/1140-1493.

[11] Li, J., Hao, P., and Zhang, C. (2008) "Transferring colours to grayscale images by locally linear embedding", In: BMVC 2008 - Proceedings of the British Machine Vision Conference 2008, pp. 83.1–83.10. doi:10.5244/C.22.83.

[12] Welsh, T., Ashikhmin, M., and Mueller, K. (2002) "Transferring color to greyscale images", In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, pp. 277–280. doi:10.1145/566570.566576.

[13] Neumann, L., Cadik, M., and Nemcsics, A. (2007) "An Efficient Perception-based Adaptive Color to Gray Transformation." In: Proceedings of the Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging 2007 (CAe 2007, June 20–22, 2007, Banff, Alberta, Canada) , 73–80. doi:10.2312/COMPAESTH/COMPAESTH07/073-080.

[14] Gooch, A.A., Olsen, S.C., Tumblin, J., and Gooch, B. (2005) "Color2gray: Salience-preserving color removal", In: ACM SIGGRAPH 2005 Papers, ACM, New York, NY, USA. pp. 634–639. doi:10.1145/1186822.1073241.

[15] Kim, Y., Jang, C., Demouth, J., and Lee, S. (2009) "Robust Color-to-gray via Nonlinear Global Mapping. ACM Transactions on Graphics" 28 (5): 1–4. doi:10.1145/1618452.1618507.

[16] NIDHI, G., and DOLLEY, S. (2017) "A Reconstruction of Gray Scale Image Into Rgb Color Space Image Using Ycbcr Color Spacing and Luminance Mapping in Matlab." i-manager's Journal on Pattern Recognition 4 (3): 17. doi:10.26634/jpr.4.3.13886.

[17] Kumar, S., and Swarnkar, A. (2012) "Colorization of gray scale images in l$\alpha\beta$ color space using mean and standard deviation", In: 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science: Innovation for Humanity, SCEECS 2012. doi:10.1109/SCEECS.2012.6184773.

[18] Ji, Y., and Chen, Y. (2008) "Rendering greyscale image using color feature", In: Proceedings of the 7th International Conference on Machine Learning and Cybernetics, ICMLC, pp. 3017–3021. doi:10.1109/ICMLC.2008.4620924.

[19] Grundland, M., and Dodgson, N.A. (2007) "Decolorize: Fast, contrast enhancing, color to grayscale conversion." Pattern Recognition 40 (11): 2891–2896. doi:10.1016/j.patcog.2006.11.003.

[20] Zheng, X., Feng, J., and Zhou, B. (2015) "Efficient color-to-gray conversion for digital images in gradient domain", In: Proceedings - VRCAI 2015: 14th ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry, pp. 85–88. doi:10.1145/2817675.2817687.

[21] Raveendran, S., Edavoor, P.J., Kumar, N.Y., and Vasantha, M.H. (2019) "Design and Implementation of Reversible Logic based RGB to Gray scale Color Space Converter", In: IEEE Region 10 Annual International Conference, Proceedings/TENCON, pp. 1813–1817. doi:10.1109/TENCON.2018.8650243.
[22] Mathworks, . Matlab convert RGB image or colormap to grayscale. URL: https://www.mathworks.com/help/matlab/ref/rgb2gray.html.
[23] International Telecommunication Union, (2011) BT 601: "Studio encoding parameters of digital television for standard 4: 3 and wide-screen 16:9 aspect ratios". ITU.
[24] Kanan, C., and Cottrell, G.W. (2012) "Color-to-Grayscale: Does the Method Matter in Image Recognition?" PLoS ONE 7 (1): e29740. doi:10.1371/journal.pone.0029740
[25] Fei-fei L., Fergus R., and Perona P. (2004) "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories." Proc Computer Vision and Pattern Recognition (CVPR-2004). doi: 10.1109/CVPR.2004.109
[26] Ĉadík, M. (2008) "Perceptual evaluation of color-to-grayscale image conversions," Comput. Graph. Forum, 27 (7): 1745–1754.
[27] Lu, C., Xu, L., and Jia, J. (2014) "Contrast preserving decolorization with perception-based quality metrics," Int. J. Comput. Vis., 110 (2): 222–239.
[28] Dalal N., and Triggs B. (2005) "Histograms of oriented gradients for human detection." In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition; pp. 886-893.