HARDWARE ACCELERATION OF LUCKY REGION FUSION ALGORITHM USING FPGA FOR REAL TIME VIDEO DEBLURRING

Project Report submitted to the
Indian Institute of Technology Bhubaneswar
For award of the degree

of

Bachelor of Technology

by

ABHISHEK PATNAIK
16EC01044

Under the guidance of

Dr. Srinivas Boppu
(Supervisior)
Dr. Niladri Bihari Puhan
(Co-Supervisior)

SCHOOL OF ELECTRICAL SCIENCES
INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

APPROVAL OF THE VIVA-VOCE BOARD

Certified that the thesis entitled **Hardware Acceleration of Lucky Region Fusion Algorithm using FPGA for Real time Video Deblurring**, submitted by **Abhishek Patnaik** to the Indian Institute of Technology Bhubaneswar, for the award of the degree of Bachelor of Technology has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Dr. Debapratim Ghosh) (Dr. Nijwm Wary) (Dr. Srinivas Boppu)

(Supervisor)

CERTIFICATE

This is to certify that the thesis entitled **Hardware Acceleration of Lucky Region Fusion Algorithm using FPGA for Real time Video Deblurring**, submitted by **Abhishek Patnaik** to Indian Institute of Technology Bhubaneswar, is a record of bonafide research work under my supervision and I consider it worthy of consideration for the award of the degree of Bachelor of Technology of the Institute.

Date :

Dr. Srinivas Boppu
Assistant Professor
School of Electrical Sciences
Indian Institute of Technology Bhubaneswar
Bhubaneswar, India

DECLARATION

I certify that

a. the work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.

b. the work has not been submitted to any other institute for any degree or diploma.

c. I have followed the guidelines provided by the institute in writing the thesis.

d. I have conformed to the norms and guidelines given in the ethical code of conduct of the institute.

e. whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Abhishek Patnaik

Acknowledgments

I would like to express my sincere gratitude to my supervisors Dr. Srinivas Boppu Dr. Niladri Bihari Puhan for providing their invaluable guidance, comments and suggestions throughout the course of the project.

I want to thank especially Dr. Srinivas Boppu for constantly motivating me to work harder and handholding me into the field of hardware design through his courses and rightly timed suggestions and Dr. Niladri Bihari Puhan for giving me the required image processing basics.

I also want to thank two of my friends Aditya Terkar and Arpit Bal who have helped me immensely through their suggestions. Lastly, I would like to thank my friends and family who have continuously inspired me to stay calm and keep working. Any omission in this brief acknowledgement does not imply my lack of gratitude to them.

Abhishek Patnaik

Abstract

Accelerating image processing algorithms can be of immense use in the field of defence, astronomy and surveillance. The lucky region fusion is a well established image processing algorithm which can extract high quality regions from a image and fuse such regions together from a set of images to get a better quality image. The lucky region fusion is effective but not for all cases. On the other hand, SSIM is well established metric for quantifying pixel by pixel image quality. This report comes up with a newer version of lucky region fusion algorithm with SSIM quality metric and there upon discusses the hardware implementation of it over FPGA for parallel processing of algorithm for real time video deblurring. The new algorithm produces better quality images and the pipelined hardware implements it with latency of 20 clock cycles.

Contents

Certification of Approval i

Certificate ii

Declaration iii

Acknowledgments iv

Abstract v

1 Introduction and Motivation 2

2 Literature Survey 4

2.1 FIRST GENERATION SYSTEM 5

2.2 SECOND GENERATION SYSTEM 5

3 The Lucky Region Fusion Algorithm 7

3.1 IDEA BEHIND LRF ALGORITHM 7

3.2 STEPS OF THE LRF ALGORITHM 7

3.3 MATHEMATICS BEHIND ALGORITHM 8

3.4 MATLAB SIMULATION AND RESULTS 9

4 SSIM based LRF Implementation 10

4.1 STRUCTURAL SIMILARITY INDEX MEASURE (SSIM) 10

4.2 SSIM CALCULATION 11

4.3 SSIM BASED LRF 12

4.3.1 MATLAB BASED SSIM LRF OUTPUT 13

4.3.2 REMARKS 13

4.4 HYBRID SSIM (H-SSIM) 13

4.4.1 MATLAB BASED H-SSIM LRF OUTPUT 14

4.4.2 REMARKS 14

5 Hardware Implementation of H-SSIM 15

5.1 CONVOLUTION ARCHITECTURE 16

5.1.1 PIPELINED CONVOLUTION ARCHITECTURE 17

5.1.2 REMARKS 18

5.2 HSSIM ARCHITECTURE 18

5.2.1 Results and Remarks 21

5.3 FUSION MODULE ARCHITECTURE 22

5.4 FULL SYSTEM ARCHITECTURE 24

5.4.1 RESULTS AND REMARKS 25

6 Improvements and Possibilities 28

References 29

Chapter 1

Introduction and Motivation

Many military applications such as target acquisition, remote surveillance, target tracking, etc require long range imaging. However, long range visual identification and detection is usually hindered by the turbulence created by the ever-changing atmospheric conditions between the camera and the object. This atmospheric turbulence causes various distortions in the captured images. As the distance between the target object and camera increases, the captured images or video degrade more in quality.

The solutions to this problem generally falls in two categories: 1. Mechanical solutions such as adaptive optics,which correct deformations before processing but are quite complex and expensive and 2. Software solutions (image processing techniques) such as lucky region fusion, blind deconvolution,etc which are much cheaper, smaller and less complex.

The high-resolution cameras used these days can overpower any computer memory quite easily. Again, computers don't have required processing capabilities to handle real time data. Hence, implementation of the software algorithms on computers can only be limited to post acquisition processing and not real-time applications. Therefore,

required hardware needs to be developed for acceleration of such algorithms to meet real-time needs.

A variety of image improvement algorithms have been developed over years to counter the effects of atmospheric turbulence on visual data and lucky region fusion (LRF) is one among them. The nature of LRF algorithm makes it easier to implement on parallel processing platforms such as FPGA. Hence, if proper hardware is developed for accelerating LRF algorithm using FPGA then real time video improvement can be achieved.

Chapter 2

Literature Survey

There have been previously very few attempts at developing required hardware for accelerating the LRF algorithm in order to obtain real time video improvement. Two significant efforts in this direction were made by William Maignan and Christopher Robert Jackson[2][3] both belonging to University of Delaware. Since, both of them belong same university and worked under same advisor, it is safe to assume that Jackson improvised over the system developed by Maignan since he mentions that he used work of Maignan. Hence, discussion of FPGA system developed by Jackson would effectively include contribution from both of them.

Jackson developed the system in two generations similar to how Maignan did, the first generation was developed as proof of concept and the second generation was the final system developed.

2.1 FIRST GENERATION SYSTEM

The first generation was developed as a proof of concept. A FastVision FastCamera13 and attached Xilinx II FPGA was used. A baseboard heater was used to simulate atmospheric turbulence. Xilinx II only contained 2160 Kbits of block RAM (BRAM), which could be used to store only a single frame at any instant of time. As a result, parallel processing capability of FPGA couldn't be used to its full potential and hence, the output frame rate was affected because if N frames were combined to give one LRF output then frame rate of output became 1/N times of input frame rate. As an improvement over this, a constant was added to IQM of the new frames while comparing with IQM of the old fused frame, to give preference to new frames. In this way, the presence of older frames on current fused frame decayed as the iterative process continued and the output frame rate also matched with the input frame rate.

2.2 SECOND GENERATION SYSTEM

The second generation FPGA LRF hardware acceleration system adopted a "black box" approach. In this methodology, the LRF algorithm was implemented on a separate FPGA, and an independent camera. This removed all dependency on the FastCamera13 while simultaneously enabling them, a wider choice of potential cameras or FPGAs. Xilinx Virtex 7 VC709 Connectivity kit was used since it has larger BRAM(52,920 Kbits of BRAM). The kit as well contains two 4GB DDR3 SODIMMs memory which were used for storing the frames.

In place of the linear frame buffer used in first generation system, a circular frame buffer was developed for handling the parallel processing and outputting of new frames so as to match the input frame rate. LRF parameters like blur radius, size of gaussian

blur, etc. were dynamically changed as per input. For this purpose, MicroBlaze processor of kit was used which changed parameters as per the requirement. EDT PCI Digital Video Cameralink Simulator was used to produce real working environment since it sent bits from pre-recorded video as if the bits were coming from a real time video. The circular frame used the VC709's DDR3 SODIMMs. They developed a DDR3 memory controller module using previous work of David Koeplinger and Memory Interface Generator tool provided by Xilinx for communication with DDR3.

Still prior to synthesis the developed hardware could only process at max 30 frames to give one LRF output because of constraint in how much data you can exchange at time from DDR3. This then further reduced to 16 (1 incoming frame+15 LRF frames) at synthesis step because processing of each synthetic frame used 4 to 6 percent of inbuilt BRAM.

It was estimated by them that the developed FPGA hardware required "(2R+2)F" number of 36Kb of BRAM for purpose of blurring and another "(R+2)F" for delay (where R=Blur Radius and F=No. of frame to be processed). Maximizing the summation of these two functions under the constraint of available BRAM led to optimum values of R=25 and F=16. Hence they used 15 input frames to produce one LRF output frame.

Unfortunately, the output of the full system didn't meet the promising results of the simulation. After synthesis and implementation, output had black patches with randomly placed white streaks but, at the same time it is important to mention that the frame rate of output matched input frame rate. They guessed that this could be happening because of overflow in frame buffer FIFO modules or because the data coming from Cameralink was not synchronized properly.

Chapter 3

The Lucky Region Fusion Algorithm

3.1 IDEA BEHIND LRF ALGORITHM

The lucky region fusion (LRF) algorithm[1] is a multiframe image restoration technique i.e. it takes a set of degraded images and combines them a get an image of better quality. It takes advantage of the fact that the atmospheric turbulence never affects regions of the image with same intensity all the time. Hence, the region which is not clear in one frame or image, may be of very good quality in next frame. These good quality regions which appear in the captured image are referred to as "lucky regions". The lucky region fusion algorithm finds such 'lucky regions' in each frame and combines them to get a better quality image.

3.2 STEPS OF THE LRF ALGORITHM

Major steps in the algorithm are as follows:

1. 1. First a reference image is chosen out of the series of images, which could be the first image or the average image. Then, based on some parameter which defines image quality an image quality map (IQM) is generated.

2. 2. An iterative process is initiated with the reference image being taken as current frame and subsequent frames being compared with current frame.

3. 3. At each step, IQM of current frame and incoming frame is compared and regions of higher IQM value are merged with current image. Hence, this current frame is also referred to as the fused image since, it is the combination of lucky regions of all past frames.

The fused image at the end contains all the high quality regions from all the images.

3.3 MATHEMATICS BEHIND ALGORITHM

Let $M_f(r), M_1(r), M_2(r)\ldots\ldots\ldots, M_N(r)$ be the IQMs of fused image (I_f) and set of images $(I_1, I_2 ... I_N)$.

At nth iteration, IQM of current fused image and incoming image is compared and an anisotropic gain function (), is calculated.

$$\delta_n = \begin{cases} (M_f(r) - M_n(r) & if M_f(r) > M_n(r) \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta_n = \delta_n / max(_n)$$

Based on n, fused image (If) for next iteration is calculated as follows:

$$I_{f(new)} = [1 - \Delta_n].I_{f(old)} + \Delta_n.I_n$$

The gain n controls the weight associated with the image region being fused during

the LRF process. Again, this weight is related to the local image quality improvement with respect to the current fused image. This ensures the fused image If (r) incorporates regions with the best image quality within the set of images.

3.4 MATLAB SIMULATION AND RESULTS

A MATLAB simulation of the LRF algorithm was implemented, which used edgemap blurred with Gaussian kernel as the IQM and the average image was chosen as the reference image. Following are the results of simulation on a graphical chart distorted by atmospheric turbulence.

Figure : (a) Sample image affected from turbulence (b) Edge Map by Sobel Edge Detection (c) Image Quality Map

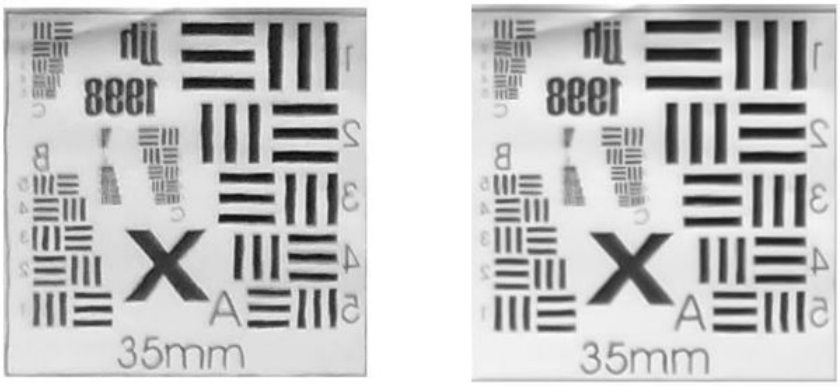


Figure : (a) Sample image affected from turbulence (b) Output image after application of LRF

Chapter 4

SSIM based LRF Implementation

4.1 STRUCTURAL SIMILARITY INDEX MEASURE (SSIM)

SSIM is used for measuring the similarity between two images, a reference and a target image. It gives better quantification of image quality the way humans perceive it.

SSIM is based on similarity in structural,luminance and contrast information in between the reference and the target image and is calculated using statistical parameters such as mean and correlation coefficients.

The original 2004 SSIM paper has been cited over 20,000 times according to Google Scholar, making it one of the highest cited papers in the image processing and video engineering field.

4.2 SSIM CALCULATION

The luminance of the surface of an object being observed is the product of the illumination and the reflectance, but the structures of the objects in a scene are independent of the illumination. Consequently, to explore the structural information in an image, we wish to separate the influence of the illumination. We define the structural information in an image as those attributes that represent the structure of objects in the scene, independent of the average luminance and contrast. Since luminance and contrast can vary across a scene, we use the local luminance and contrast.

$$\text{Luminance function },l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$$

$$\text{Contrast function },\ c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$$

$$\text{Structural function },\ s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

$$Choosing c_3 = c_2/2$$

$$SSIM(x,y) = l(x,y)\ .\ c(x,y)\ .\ s(x,y)$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)\ .\ (2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)\ .\ (\sigma_x^2 + \sigma_y^2 + c_2)}$$

All the statistical parameters are actually matrices, which multiplied according to previous equation to gives a matrix which shows which regions of target image are more similar to that of reference image.

The Statistical parameters used in SSIM calculation, are calculated as follows:-

μ_x *or* μ_y = Image convolved with Gaussian kernel/mean filter.

σ_x^2 = Variance = Gaussian blurring(X^2) - μ_x^2

σ_y^2 = Variance = Gaussian blurring(Y^2) - μ_y^2

σ_{xy} = Co-variance = Gaussian blurring($X.Y$) - $\mu_x\mu_y$

4.3 SSIM BASED LRF

In previous LRF implementation,the following were used:

(1) Reference Image-¿ Average Image

(2) Image Quality Parameter-¿ Edge map with gaussian blurring

Improvement in the output was indeed seen, but not all portions of the image were improved. So, a newer implementation of LRF algorithm was needed and SSIM has proved to be quite successful in quantifying image quality the way human brain perceives it.

Hence, in the newer implementation we go for:

(1)Reference Image-¿ First Image in the set of images

(2) Image Quality Parameter-¿ SSIM of the present frame with average of this and last N-1 frames.

4.3.1 MATLAB BASED SSIM LRF OUTPUT

Figure : (a) (Left) Original Image (b) (Middle) Previous LRF Implementation (c)(Right) SSIM based LRF

4.3.2 REMARKS

In the above figure, the previous LRF implementation does improve the quality , but region near the staircase (bottom) doesn't improve significantly. On the other hand , the SSIM based LRF implementation improves the bottom region as well. But, when played as frames of video, the SSIM based LRF creates a "wobbly" effect around edges which isn't present with the previous edge map based LRF implementation. Hence, a midway between the two approaches is needed. This is explored in the next section.

4.4 HYBRID SSIM (H-SSIM)

The edge map based LRF does it's LRF computations on sobel edge map and the output of which had stabilised edges.So, rather than applying SSIM on the present frame with respect to average of present and past N-1 frames, we apply SSIM on edge

map of the present frame with respect to edge map of the average of present and past N-1 frames. This edge map based SSIM is hereon referred to as Hybrid SSIM or H-SSIM.

4.4.1 MATLAB BASED H-SSIM LRF OUTPUT

Figure : (a) (Left) Edge Map based LRF (b) (Middle) SSIM based LRF Implementation (c)(Right) H-SSIM based LRF

4.4.2 REMARKS

The HSSIM based LRF does improve all image regions like SSIM and the outut frames have more stable edges than SSIM based LRF implementation. It takes better quality from SSIM and more stable edges from the edge map, and offers the much needed algorithm which if accelerated can do better video deblurring output than earlier LRF implementations.

Chapter 5

Hardware Implementation of H-SSIM

The hardware of HSSIM based LRF consists of the following steps:

1. Calculation of average(image) of present and past N-1 frames.

2. Calculation of HSSIM values of fused and incoming frame using averge image.

3. Fusion of higher quality regions from incoming frame into fused frame to create a new fused frame.

4. Storing of fused and incoming frame at appropriate memory locations.

5. The above process should repeat in a pipelined fashion so as to facilitate parallel processing of frames and at the same time maintaining the output frame rate to be same as the input frame rate.

5.1 CONVOLUTION ARCHITECTURE

The proposed H-SSIM architecture involves convolution blocks in most of it's basic operations. Hence, it is important for the convolution block to be fast for real time processing. So, we go for pipe-lined convolution architecture.Firstly, the usual convolution hardware(direct 2D convolution) is described then follows the pipelined convolution hardware.

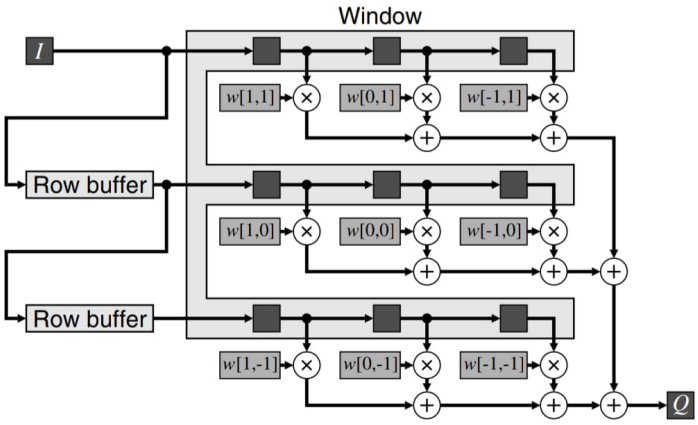


Figure : Direct 2D Convolution Implementaion

In the direct filter implementation, the inputs are stored in the row buffers and from appropiate places of row buffer the inputs are taken and are multiplied by filter coefficients and added to produce output. Delay is large in this implemenation. Consider the case of convolution of image with a 3X3 matrix, every time you perform 9 multiplications and 8 additions to get the required sum. The 8 additions can be done with least delay in a 4 stage adder tree.

$$Delay = T_{mul} + 4 * T_{add}$$

.

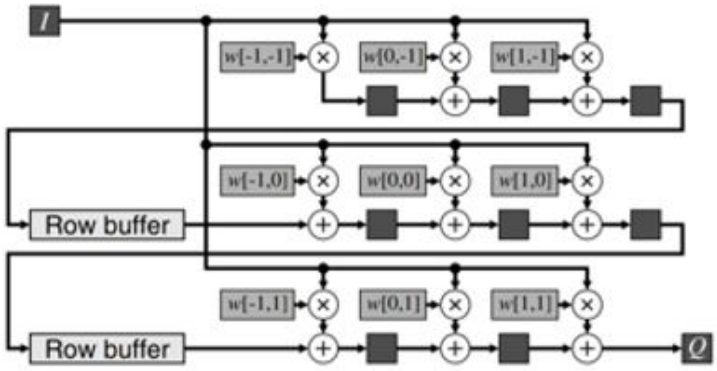


Figure: Pipelined 2D filter implementation
(Source:Bailey G.,Design for Embedded Image Processing on FPGAs)

In pipelined convolution implementation, partial products are stored in place of inputs. The partial products corresponding to the pixels which would get affected by input, are added at appropriate places of one large row buffer and these partial products get appropriately added to each other as they come out of row buffer. For same case, delay is less since the additions happen at different places and hence can happen in parallel.

$$Delay = T_{mul} + T_{add}$$

5.1.1 PIPELINED CONVOLUTION ARCHITECTURE

Following are the output images for edge map, obtained using convolution of image with sobel edge matrices for horizontal and veritical edges and then taking their average.

Figure : (a) (Left) Original Image (b) (Middle) Edge Map (MATLAB) (c)(Right) Edge Map (Verilog)

5.1.2 REMARKS

As seen from above figures, the convolution block performs operation as expected.

5.2 HSSIM ARCHITECTURE

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1) \,.\, (2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1) \,.\, (\sigma_x^2 + \sigma_y^2 + c_2)}$$

Based on above equation , different statistical parameters μ and σ are calculated by using multiple convolution blocks, with different filter coefficients.

Statistical parameters:-

μ_x *or* μ_y = Image convolved with Gaussian kernel/mean filter.

σ_x^2 = Variance = Gaussian blurring(X^2) - μ_x^2

σ_y^2 = Variance = Gaussian blurring(Y^2) - μ_y^2

σ_{xy} = Co-variance = Gaussian blurring($X.Y$) - $\mu_x\mu_y$

The following operations are needed to get the value of HSSIM:

1. Calculation of the edge map of both reference and target image calculated.

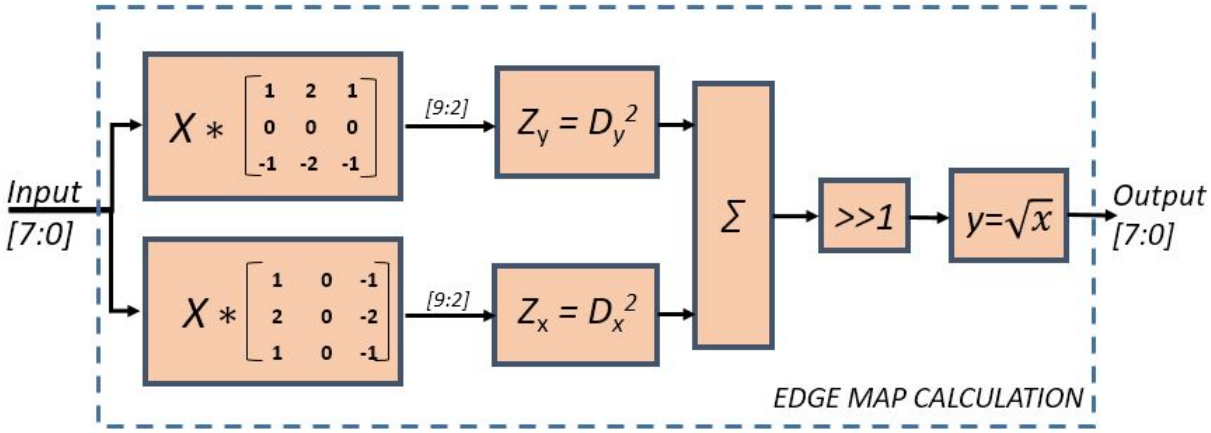


Figure : Image Edge Map calculation block diagram

The hardware first calculates the horizontal and vertical edge maps by convolving input image with appropriate sobel edge matrices . The resultant horizontal and vertical edge maps are squared and added together and divided by 2 (right shift by 2). The final output is square root of this sum. The square root calculation was done by square root IP provided by Xilinx. The bit shifts are done to maintain the final bitwidth of 8.

2. Calculation of mean (μ_x *or* μ_y) of edge map of target and reference image.

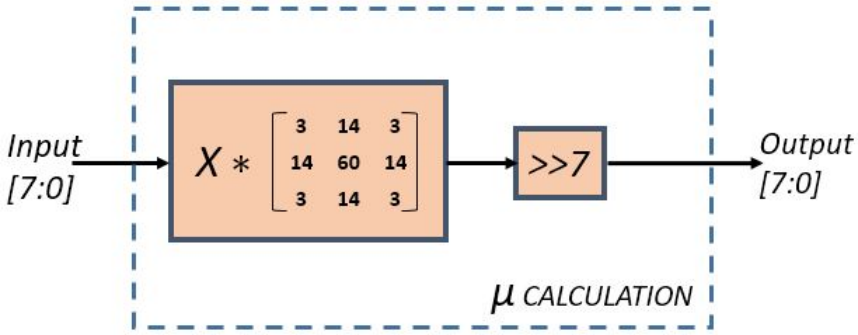


Figure : Image Mean calculation block diagram

The matrix used for mean is gaussian 2d matrix for sigma=0.5 multiplied by 128, to convert them into values greater 1, because our system deals with integer values.The result is then divided by 128 (right shift by 7), to get the correct output.

3.Calculation of variance (σ_x^2 $or\sigma_y^2$) of edge map of target and reference image and their covariance(σ_{xy}).

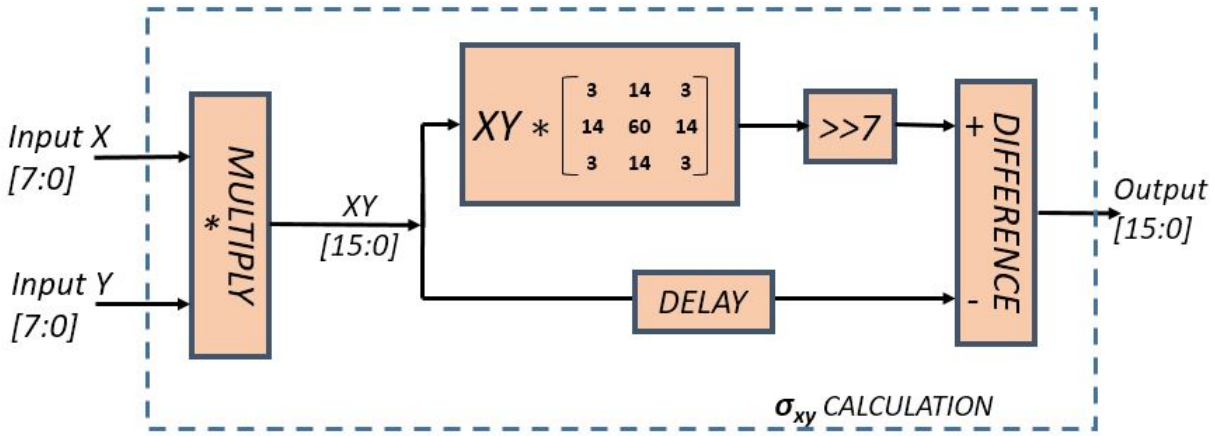


Figure : Image Co-variance calculation block diagram

The product matrix signal is first blurred with gaussian matrix with sigma=0.5 thereafter product matrix signal input is subtracted from it to get the covariance between 2 images. For getting variance, you pass the same signal as X and Y.

4. Calculation of nmerator and denominator terms of HSSIM.

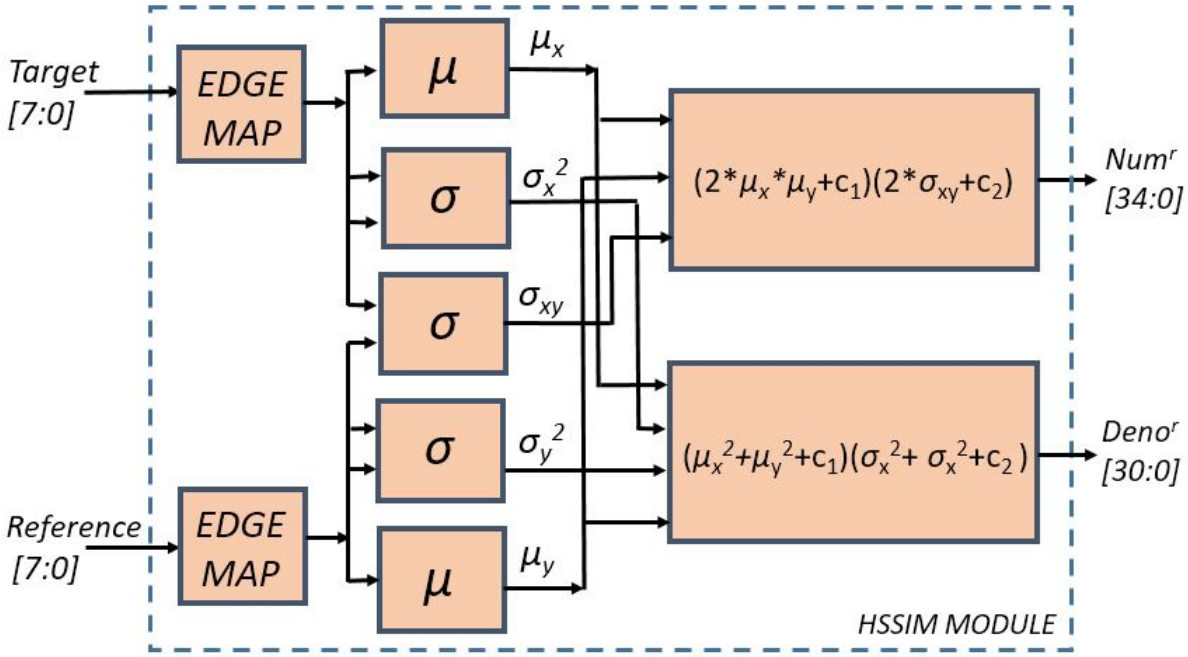
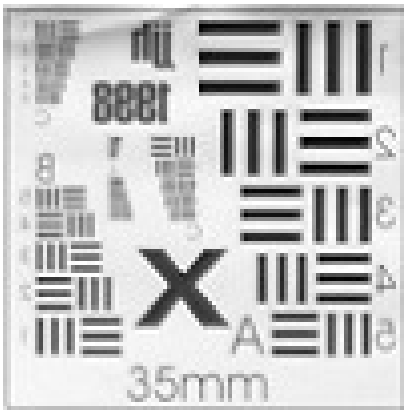


Figure : HSSIM calculation block diagram

The Numerator term and denominator term of HSSIM are calculated separately using mean and covariance(or variance) values of edgemaps.

5.2.1 Results and Remarks



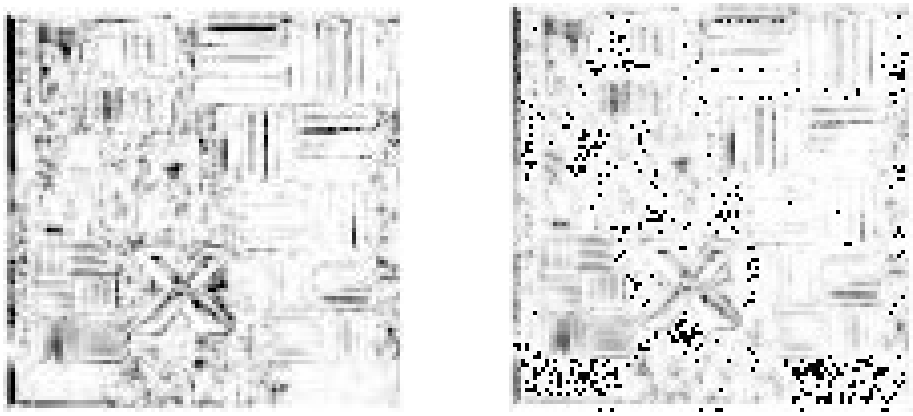


Figure : (a) (Top Left) Ref. Image = Average Image (b) (Top Right) Target Image (c) (Bottom Left) HSSIM (MATLAB) (d)(Bottom Right) HSSIM (Verilog)

. As seen from above figures, the HSSIM block implemented in verilog produces output very close that generated from MATLAB, except for few black dots.

5.3 FUSION MODULE ARCHITECTURE

The fuser module implemented takes HSSIM values for the fused and incoming frame.If the HSSIM value corresponding to incoming frame is higher than HSSIM value of old fused image, then the resultant pixel intensity is combination of pixel intensities of fused and incoming frame combined in the ratio of their del values otherwise the new fused image pixel is same as the old image pixel.

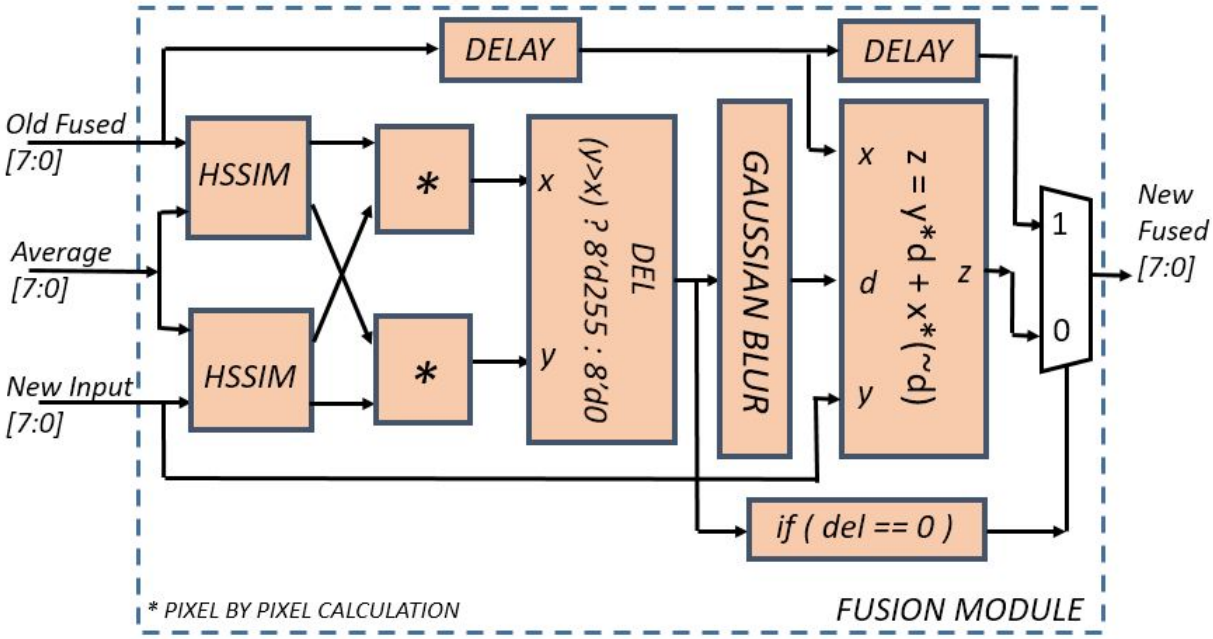


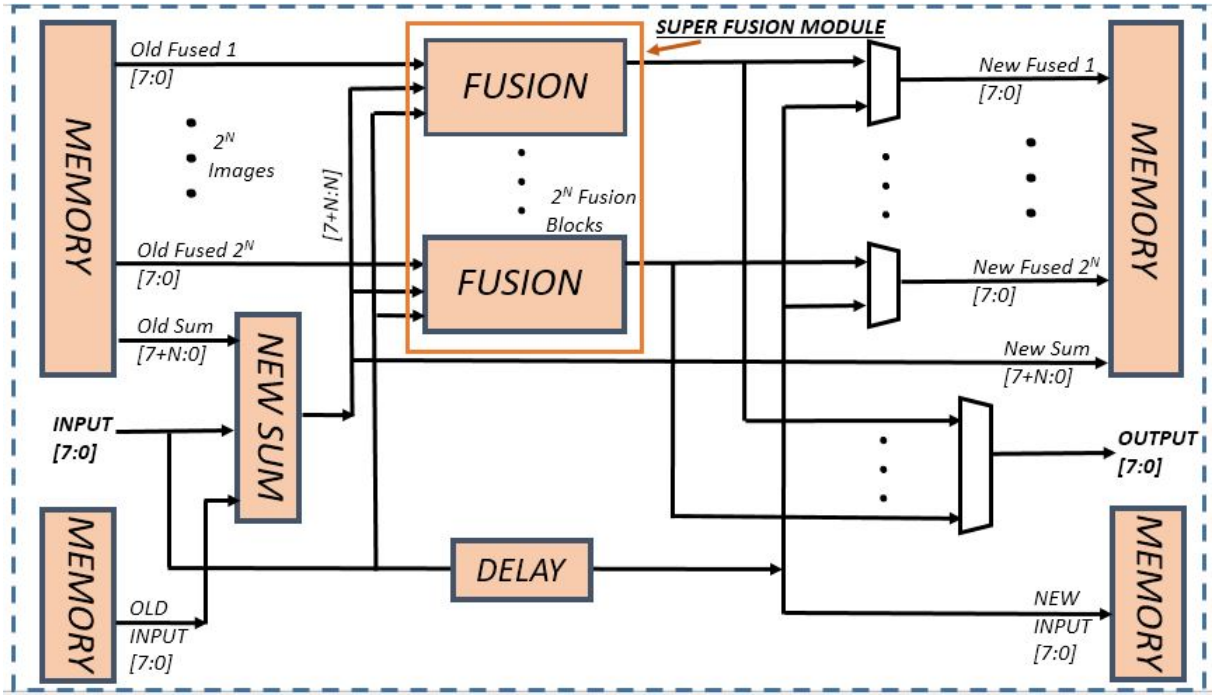
Figure : HSSIM calculation block diagram

$$Now, if HSSIM_1 > HSSIM_2$$
$$=> \frac{Numr1}{Deno1} > \frac{Numr2}{Deno2}$$

s

$$=> \begin{cases} (Numr1 * Deno2) > (Numr2 * Deno1) & if (Deno1) * (Deno2) > 0 \\ (Numr1 * Deno2) < (Numr2 * Deno1) & otherwise \end{cases}$$

5.4 FULL SYSTEM ARCHITECTURE



In the final system, the fused images, sum of images and old input images till last 2^N frames are continuously stored in and retrieved from memory using address generated by FSM modules. The rest of the system works as follows:

1. First the sum of last 2^N frames is calculated by subtracting old image value from present sum value and adding the new image intensity value.

2. Thereafter, the most significant 8 bit of new sum are taken as average of past 2^N frames are reference image and is passed on to the super fusion module along with new input image and 2^N old fused frames. The super fusion module contains 2^N fusion modules combined together to save on redundant hardware, since the HSSIM of new input image is calculated repeatedly across fusion modules.

3. The system then chooses between the new input image and fusion module output for 2^N cases. If for a particular case, the input image is used as new fused image, then

respective fusion module output appears as system output.This happens cyclically across the 2^N cases according a FSM module which runs from 0 to $2^N - 1$(increments when processing of one image is complete).

4. The new fused image, new sum and new input image are then stored in the memory and the process keeps repeats.

5.4.1 RESULTS AND REMARKS

The important point to mention here that the first valid output comes after 2^{N+1} frames, the first 2^N images to reach a valid average image and the next 2^N frames to produce image which is fused output of 2^N images.

The output of behavioural simulation, post synthesis functional simulation and post implementation functional simulation of each of the modules has been matched pixel by pixel with the expected output generated from MATLAB.

The latency between input and output of pipelined model is 20 clock cycles.

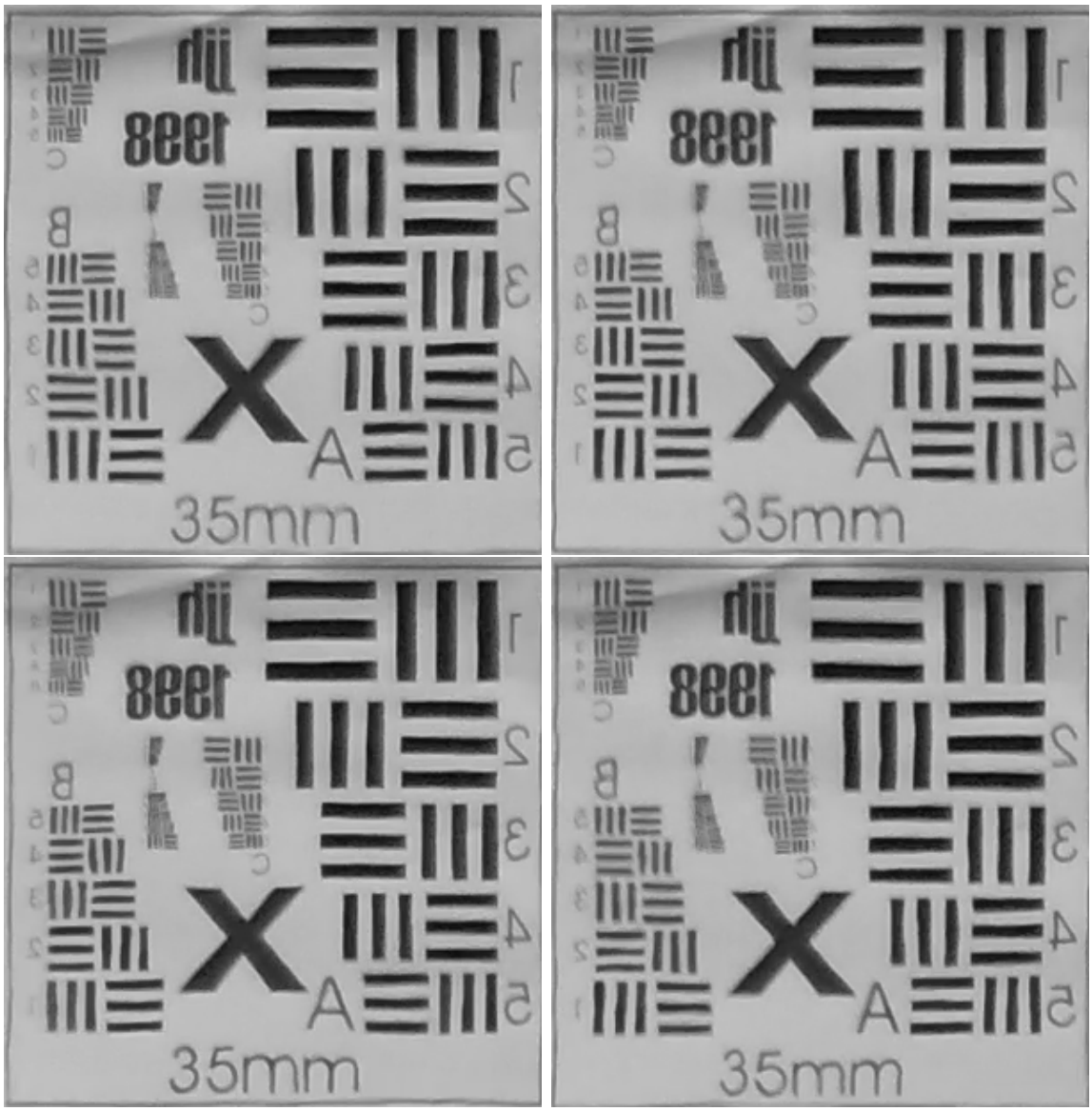
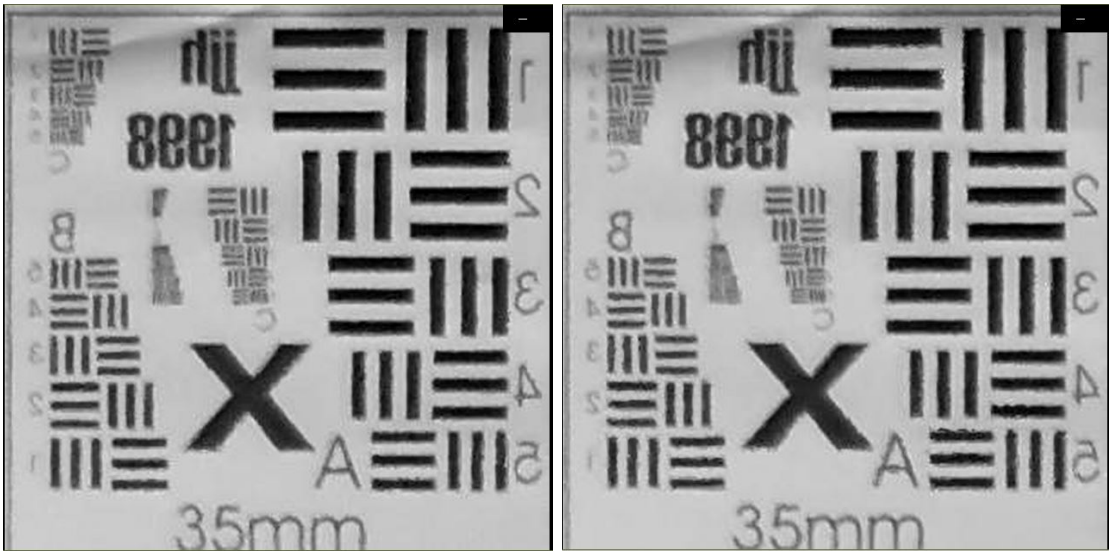


Figure : Sample frames of a video distorted by atmospheric turbulence



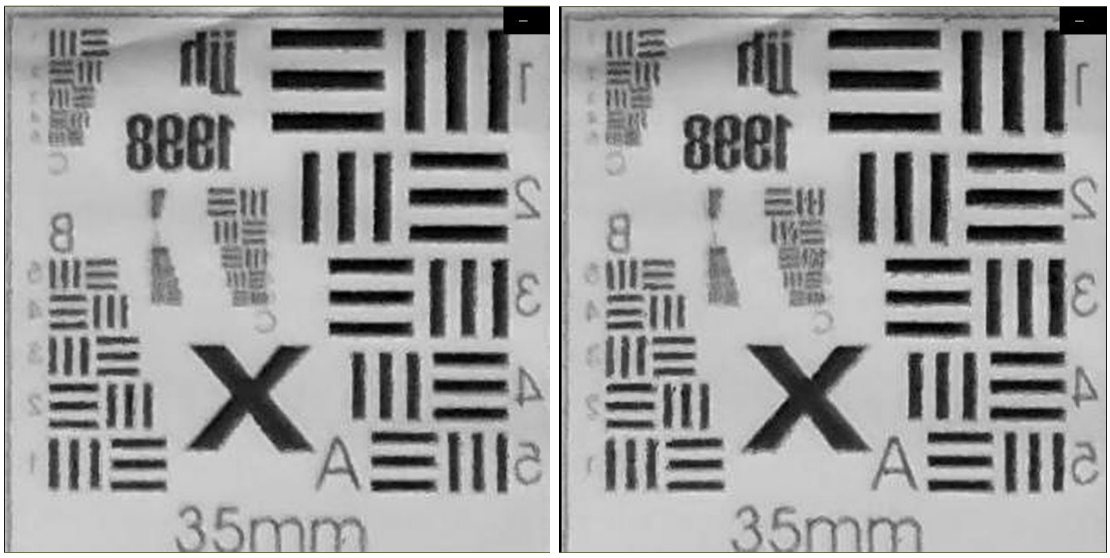


Figure : Sample frames of a video distorted by atmospheric turbulence

Chapter 6

Improvements and Possibilities

The present system has not been incorporated with DDR3 interface and HDMI interfaces. Hence, integrating the present system with DDR3 and HDMI interface will provide a more complete system.

The developed system works for static images, incorporating motion detection and thereupon tweaking the algorithm and it's hardware implementation is necessary for the system to work in on field scenarios.

Again, the system at present works on grayscale 8 bit images , improving it to include RGB color images would be a much required improvement.

References

[1] M. Aubailly, M. A. Vorontsov, G. W. Carhat. And M. T. Valley. “Automated video enhancement from a stream of atmospherically-distorted images: the lucky-region fusion approach”. Proceedings of SPIE, 2009.

[2] Maignan, William, David Koeplinger, Mathieu Aubailly, Gary W. Carhart, Fouad Kiamilev, and J. Jiang Liu. ”Hardware acceleration of lucky-region fusion (LRF) algorithm for image acquisition and processing ”, Proc. SPIE 8720.

[3] Jackson, Christopher R. ”Real Time Mitigation of Atmospheric Turbulence in Long Distance Imaging Using the Lucky Region Fusion Algorithm with FPGA and GPU Hardware Acceleration.” University Microfilms, Inc. 2015.

[4] The Development Channel. (2019, October 19). VIVADO HLS 2D Convolution on hardware- part 1 [Video File]. Retrieved from https://www.youtube.com/watch?v=38lj0VQci7E

[5] Bailey,Donald G., Design for Embedded Image Processing on FPGAs. 1st ed. (2011)

[6] Cabello F.,Leon J. , Iano Y. and Arthur R. “Implementation of Fixed-Point 2D Gaussian Filter for Image Processing based on FPGA”. IEEE SPA 2015.

[7] Xilinx Synthesis and Implemenation guide for Vivado.

[8]florentw (UserName) (2020, May). Xilnx Forums :https://forums.xilinx.com/t5/Video-and-Audio/Xilinx-Video-Series/td-p/849583

[9] Garcia Edgard (2020, May). Writing RTL Code for Virtex-4 DSP48 Blocks with XST 8.1: Writing RTL code for your DSP applications is easy and efficient. : http://www.mvd-fpga.com/cores/en/files/publications$_0$8.*pdf*