VLSI ARCHITECTURE OF AN AREA EFFICIENT IMAGE INTERPOLATION

John Moses C [*1], Selvathi D[#2]

[#1]Assistant Professor, Dept. of ECE, St. Xavier's Catholic College of Engineering, Nagercoil, Tamil Nadu, India
[1] jofjef@yahoo.com

[*2]Professor, Dept. of ECE, Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu, India
[2]dselvathi@gmail.com

***Abstract*— Image interpolation is widely used in many image processing applications, such as digital camera, mobile phone, tablet and display devices. Image interpolation is a method of estimating the new data points within the range of discrete set of known data points. Image interpolation can also be referred as image scaling, image resizing, image re-sampling and image zooming. This paper presents VLSI (Very Large Scale Integration) architecture of an area efficient image interpolation algorithm for any two dimensional (2-D) image scalar. This architecture is implemented in FPGA (Field Programmable Gate Array) and the performance of this system is simulated using Xilinx system generator and synthesized using Xilinx ISE smulation tool. Various VLSI parameters such as combinational path delay, CPU time, memory usage, number of LUTs (Look Up Tables) are measured from the synthesis report.**

Keyword- Convolution interpolation, FPGA, Re-sampling, Line buffer, Weight generator

I. INTRODUCTION

Image interpolation technique is a widely used scheme in image processing, medical imaging, and computer graphics [1]. Image interpolation is a method of constructing new data points within the range of a discrete set of known data points [2], [3]. Interpolation processes are transformations between two habitually sampled grids, one at the input resolution, and another at output resolution [4]. A variety of applications require image zooming, such as digital cameras, electronic publishing, third-generation mobile phones, medical imaging, and image processing [5]. An image resolution limit the scope to which zooming develops clarity, limits the quality of digital photograph magnifications and in the circumstance of medical images can avert the correct diagnosis. Lone image interpolation (zooming, up-sampling, or resizing) can synthetically increase image resolution for displaying or printing, but is usually limited in conditions of enhancing image precision, or revelling higher frequency substance. Image interpolations based on estimates of the model sinc kernel (pixel replication, bilinear, bi-cubic, and higher-order splines) are normally used for their flexibility and speed, though these methods commonly to blurring, ringing artifacts, jagged edges, and abnormal depiction (curves of substance intensity) [6]. But these methods can be optimized by updating the sinc-approximating kernel to the image being interpolated.

In recent years many type of image interpolation techniques have been proposed. The cubic convolution interpolation task which affords a good conciliation between the computational complexity and rebuilding precision has been exploited [7]. In the midst of various proposed interpolation algorithms, the simplest one is the nearest neighbour algorithm [8]. It needs a quite low time complexity and a moderately easy implementation since of its only selecting the nearest value from the nearest points as outcomes. Though, the images that are interpolated by the nearest-neighbour method are complete of blocking and aliasing artifacts. Another simplest method of interpolation is bi-linear algorithm which employs linear interpolation form to compute unknown pixels. But it makes serious blurring problem [5]. Bicubic is one of the conventional image interpolation techniques. This method is attractive on the aspect of algorithmic simplicity, which is highly desirable for fast implementation. But, this method may introduce blurring and other annoying image artifacts especially around edges [9]. Extended linear convolution interpolation is proposed [10]. It can give high image quality compared to bi-cubic interpolation. Another interpolation called the Error Amended Sharp Edge (EASE) scheme is used to reduce the interpolation error which is based on bilinear interpolation method [11].

In many practical real time applications, the interpolation process is included in the end user equipment. It has become a considerable trend to design a low-cost, high class, and high speed interpolation by the VLSI techniques from home appliances to medical image processing [12]. VLSI hardware architecture for any application can be implemented by using FPGA. FPGA is an integrated circuit designed to be configured by a customer or a designer often manufacturing.

This work presents VLSI architecture of low-complexity image interpolation algorithm based on convolution kernel [13] and EASE interpolation [11]. The remaining portion of this paper is structured as follows. Chapter II gives a brief analysis of different image interpolation techniques. Chapter III describes in detail the FPGA implementation of the proposed interpolation architecture. Chapter IV compares the different

VLSI optimization parameters such as number of Look up tables (LUTs), power and combinational path delay of both existing method [13] and proposed method.

II. RELATED WORKS

Winscale image interpolation [14] is implemented by using an area pixel model for image scaling. Winscale performs the scale up/down transform using an area pixel model rather than a point pixel model. This method has high frequency and image quality than bi-linear interpolation method. This work requires small amount of memory and four line buffers. The synthesis results show that the winscale utilizes 29000 NAND equivalent gate counts.

VLSI implementation of low-power high-quality colour interpolation [15] is proposed for CCD camera. This work is based on edge oriented weighting and local gain approach. This method provides a colour interpolation technique for a single-chip charge coupled device (CCD) with colour-filter array format. The VLSI architecture can interpolate different colours using a common computational kernel, reducing the circuit complexity. The interpolation chip uses about 10 K gates and two line buffers. The synthesis report shows that the circuit complexity with 518 slices, 163 slice flipflops and 795 look-up tables (LUTs).

An Edge oriented Image scaling processor [16] is proposed for low-cost with low-complexity VLSI architecture. This method uses a simple edge catching technique to preserve the image edge features and to achieve better image quality. The results of this work states that the seven stage VLSI architecture contains 10.4 K gate counts and yields a processing rate of about 200 MHz by using TSMC 0.18-µm technology. The hardware architecture of this algorithm uses a single line buffer.

VLSI design of bi-cubic convolution interpolation [17] is proposed for digital image processing. This architecture reduces the computational complexity of generating coefficients and decreasing number of memory access time. The synthesis results show that the bi-cubic convolution utilizes 30643 gates at 279 MHz in a 498 x 498 μm^2 with 0.13 µm VLSI technologies.

FPGA architecture of extended linear convolution interpolation [18] is proposed for real-time digital image scaling. This work says that the bi-linear convolution interpolation is a low-cost architecture with the interpolation quality compatible to that of bi-cubic convolution interpolation method. This architecture saves about 60% of hardware cost. Also, this is implemented on the Virtex – II FPGA and utilizes about 379 CLBs at 104 MHz. But the bi-cubic interpolation [5] utilizes about 437 CLBs at 279 MHz.

A low-cost high-quality adaptive scalar [19] is proposed for real-time multimedia applications. This work adopts bilinear interpolation algorithm due to its low-complexity and high-quality. The bi-linear interpolation is simplified by hardware sharing technique to reduce computing resource and hardware costs. The VLSI architecture of this method can achieve 280 MHz with 9.28 K gate counts, and its chip area is 46418 μm^2 synthesized by 0.13 µm CMOS process. This work utilizes the sum of 4 line buffers.

A piecewise linear convolution interpolation with third-order approximation [20] is proposed for real-time image processing. This method reduces the computational complexity of generating weighting coefficients and gives simple hardware architecture with low-cost. The architecture has been designed on the Virtex-II FPGA and it utilizes 393 CLBs at 104.3 MHz in 447 x 447 μm^2 with TSMC 0.13 µm VLSI technology.

Fast first-order polynomial convolution interpolation [21] is proposed for real-time digital image reconstruction. The kernel of this method is built up of first-order polynomials and approximates the ideal sinc-function in the interval [-2, 2]. This architecture reduces the computational complexity of generating weighting coefficients and provides simple hardware architecture. The architecture was implemented on the Virtex-II FPGA with the TSMC 0.13 µm CMOS process. The synthesis results show that the interpolation quality of this architecture is better than cubic convolution interpolation. This method utilizes 414 Configurable Logic Blocks (CLBs) with gate count 28904.

VLSI Implementation of low-cost high quality image scaling processor [12] is proposed by for low-complexity and low-memory requirement algorithm. This implementation uses a T-model and an inverse T-model convolution kernel for realizing the sharpening spatial and clamp filters. Also, it combines two T-models into a combined filter to utilize one-line buffer memory. The combined filter reduces the computing resource and hardware cost. The VLSI architecture of this implementation can achieve 280 MHz with 6.08 K gate counts, and its core area is 30378 μm^2 synthesized by 0.13 µm CMOS process. This work reduces gate count by 34.4% from the previous bilinear algorithm.

VLSI implementation of an adaptive edge-enhanced image-scalar is proposed [22] for multimedia applications. This work is based on bi-linear interpolation algorithm. This uses edge detector to discover the edges by low-cost edge catching technique. To reduce the blurring effect a sharpening spatial filter has been used. A hardware sharing technique is used to simplify bilinear interpolation, which reduces the computing resources and chip area. This design can process streaming data by using only a one-line buffer memory. The

VLSI architecture of this work contains 6.67 K gate counts and achieves about 280 MHz by using the TSMC μm VLSI technologies.

Iterative linear interpolation based on fuzzy gradient model is proposed [23] for low-cost VLSI interpolation architecture. This produces quadratic iterative linear interpolation polynomials to perform interpolation. It adopts fuzzy gradient model to estimate gradients of the target point according to its neighbour sample points in different direction. It produces better robustness compared with bi-cubic interpolation. The TSMC 0.18 μm technology shows that only 7256 gates are required for running a 200 MHz, 8-bit input/output 15-bit fixed-point data path.

A novel interpolation chip [13] is proposed for real –time multimedia applications. This work provides a novel scaling algorithm for the implementation of two dimensional (2-D) image scalar. This interpolation scheme is based on the interpolation error theorem and bi-lateral error-amender. Based on interpolation error theorem and bi-lateral error amender, this work develops an interpolation kernel for image scaling. Also, this method adopts the principle of the edge-weighted scheme and convolution interpolation to enhance edge features and to provide better image quality. This architecture uses resource sharing to decrease the computing resources and hardware cost. A nine-stage pipelined VLSI architecture achieves 278 MHz with 13K gate counts using TSMC 0.13 μm technology. Also, this work utilizes about four line buffers.

An error-amended sharp edge (EASE) scheme [11] is proposed for image zooming. This EASE scheme is a modified bi-linear method. EASE can remove/reduce interpolation artifacts such as image blur and the check-board effect by using interpolation error theorem. This may turn out similar effect as cubic interpolation method.

The goal of this proposed work is to reduce the computational complexity by reducing number of components used for image interpolation. The proposed hardware for convolution kernel based interpolation is developed by using only one line buffer and it eliminates the reorder module. Therefore, the proposed architecture can reduce the number of combinational elements such as adder and multiplexer. So the new architecture has less computational complexity as compared with the existing architecture [13].

III. FPGA IMPLEMENTATION

The proposed FPGA implementation of image interpolation is based on EASE interpolation algorithm [11] The EASE interpolation method uses interpolation error theorem and convolution kernel [13]. The goal of the proposed work is to reduce the computational complexity by reducing number of components used for interpolation. This proposed work tries to reduce the number of line buffers and combinational elements such as adder and multiplexer of the existing architecture [13].

A. *Concept of Convolution Based Interpolation Algorithm*

Convolution kernal based two dimentational interpolation is generally decomposed as two one dimentational operations as vertical interpolation and horizontal interpolation to decrease the computational complexity. The interpolation circuit receives stream-in data from orginal images and produce stream-out results for scaled images. With the execution direction of interpolation, the interpolation circuit first interpolates the four intermediate values through the vertical direction, and then uses these values to interpolate the target value through the horizontal direction. Figure 1 denotes the luminance values of cocordinate (x,y) at the source pixel $S_{x,y}$, intermediate pixel $I_{x,y}$, and target pixel $T_{x,y}$ respectively, where (x_c,y_c) represents the coordinate of the correct target pixel to be interpolated.

The convolution interpolation kernal can be defined as [13][24][25]

$$C_0(s) = -(1-s)^2 s,$$
$$C_1(s) = (1-s) + 2(1-s)^2 s - (1-s)s^2$$
$$C_2(s) = s + 2(1-s)s^2 - (1-s)^2 s$$
$$C_3(s) = -(1-s)s^2 \qquad (1)$$

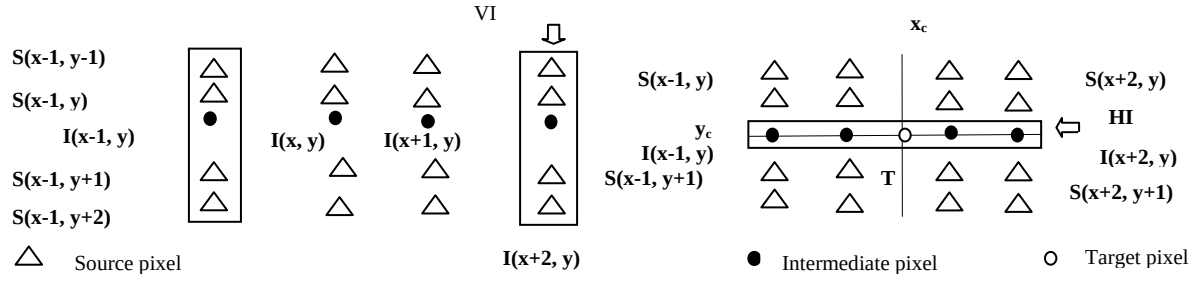

Figure 1: An example to interpolate 2-Dimensional images

The first intermediate value $I_{\lfloor x_c \rfloor -1, y_c}$ through vertical path can be intended as,

$$I_{\lfloor x_c \rfloor -1, y_c} = C_0(S_v) \times S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor -1} + C_1(s_v) \times S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor} + C_2(s_v) \times S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor +1} + C_3(S_v) \times S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor +2} \quad (2)$$

$\lfloor x \rfloor \rightarrow$ is the biggest integer not bigger than x

$S_v \rightarrow$ is the present distance in the vertical path

$S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor +1}$, $S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor}$, $S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor +1}$ and $S_{\lfloor x_c \rfloor -1, \lfloor y_c \rfloor +2}$ are the four neighbouring values in the vertical path. Three additional intermediate values $I_{\lfloor x_c \rfloor, y_c}$, $I_{\lfloor x_c \rfloor +1, y_c}$, $I_{\lfloor x_c \rfloor +2, y_c}$ can be intended in the similar way by the dissimilar neighbouring values (Figure 1). These values are able to interpolate the target value T_{x_c, y_c} throughout the horizontal path as,

$$T_{x_c, y_c} = C_0(S_h) \times I_{\lfloor x_c \rfloor -1, y_c} + C_1(S_h) \times I_{\lfloor x_c \rfloor, y_c} + C_2(S_h) \times I_{\lfloor x_c \rfloor +1, y_c} + C_3(S_h) \times I_{\lfloor x_c \rfloor +2, y_c} \quad (3)$$

$S_h \rightarrow$ is the present distance in the horizontal path. $C_0(S_h)$ to $C_3(S_h)$ be the consequent weights mentioned in equation (1). Equation (3) can also be written as

$$T_{x_c, y_c} = (1 - S_h) \times I_{\lfloor x_c \rfloor, y_c} + S_h \times I_{\lfloor x_c \rfloor +1, y_c} + \left[(1 - S_h)(2I_{\lfloor x_c \rfloor, y_c} - I_{\lfloor x_c \rfloor -1, y_c} - I_{\lfloor x_c \rfloor +1, y_c}\right) + S_h(2I_{\lfloor x_c \rfloor +1, y_c} - I_{\lfloor x_c \rfloor, y_c} - I_{\lfloor x_c \rfloor +2, y_c}]S_h(1 - S_h) \quad (4)$$

Equation (4) shows that it needs 14 operations per pixel. To decrease the needed computing resources of the hardware architecture, the design [13] uses the connection between the neighbour pixels and modifies equation (4) as

$$T_{x_c, y_c} = (1 - S_h) \times I_{\lfloor x_c \rfloor, y_c} + S_h \times I_{\lfloor x_c \rfloor +1, y_c} + (1 - S_h)(2E_C^L) + S_h(2E_C^R)]S_h(1 - S_h) \quad (5)$$

Where $2E_C^L$ and $2E_C^R$ are the present 2-Dimensional error amenders. The error amended system is based on the bilinear interpolation technique.

B. Hardware Architecture

The block diagram of the image inaterpolation architecture is shown in figure 2. It includes the Scaling Coordinate Accumulator (SCA) , the Weight Generator (WG), the Vertical Interpolation unit (VI) and the Horizontal Interpolation unit (HI).

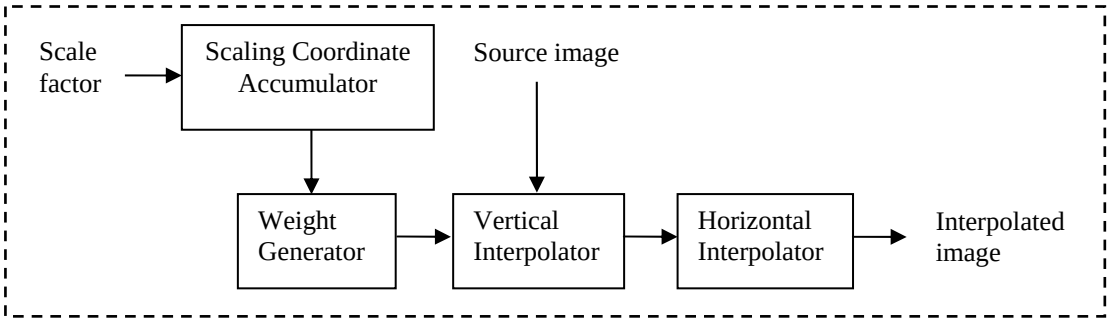

Figure 2: Block diagram of Image Interpolation

The Coordinate Accumulator (CA) calculates the corresponding coordinate of the current target pixel to be interpolated. Its inputs are two scaling factors one is for the Horizontal direction denoted as scale_x and another one is for the Vertical drection denoted as scale_y. These factors provide the scaling ratio of each direction. The current target pixels coordinate (x_c, y_c) is calculated as,

$$x_c = x_{c-1} + scale\_x$$

$$y_c = y_{c-1} + scale\_y$$

Where (x_{c-1}, y_{c-1}) coordinate of the previous target pixel.

The detailed structure of the Weight Generator (WG) [13] is depticted in Figure 3

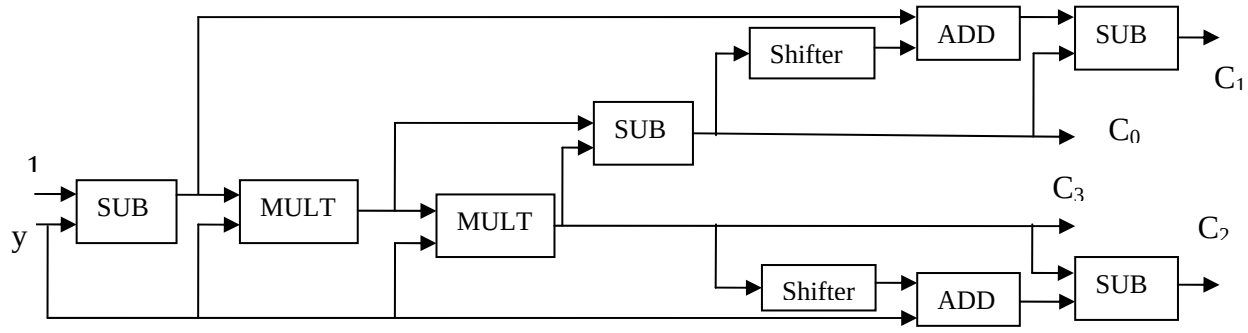

Figure 3: Architecture of Weight Generator

The WG generates the corresponding four weights $C_0(S_h)$ to $C_3(S_h)$ in Equation (1). The WG first obtains the value of S_v as

$$S_v = Y_c - \lfloor Y_c \rfloor$$

Equation (1) shows that the values of both $C_0(s)$ to $C_3(s)$ are negative or zero. Therefore, adopt $C_0'(S_v)$ and $C_3'(S_v)$ intend $C_0(s)$ and $C_3(s)$ of are defined as

$$C_0'(S_v) = (1 - S_v)(1 - S_v)S_v \quad (6)$$
$$C_3'(S_v) = [(1 - S_v)S_v]S_v$$

Therefore, WG produces the four weights in the series as,

$$C_3'(S_v) = [(1 - S_v)S_v]S_v$$
$$C_0'(S_v) = [(1 - S_v)S_v - C_3']$$
$$C_1(S_v) = (1 - S_v) + 2C_0' - C_3'$$
$$C_2(S_v) = S_v + 2C_3' - C_0' \quad (7)$$

The complete organization of the Vertical Interpolator [13] is shown in Figure 4.

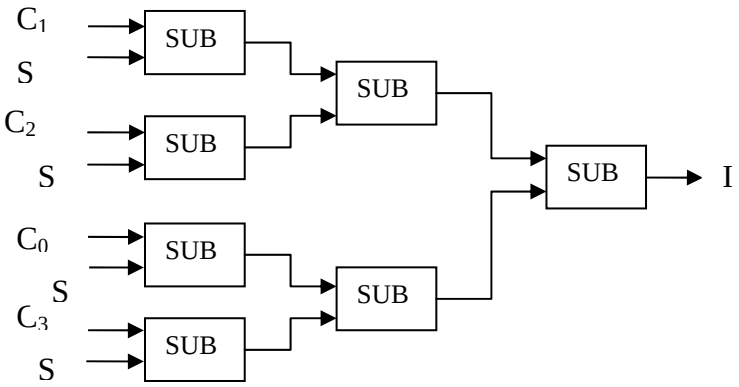

Figure 4: Architecture of Vertical Interpolator (VI)

The VI is used to generate the intermediary values using the four weights and Luminance values acquired from the WG and RM respectively.

$$I_{\lfloor x_c \rfloor - 1, y_c} = \left(C_1(S_v) \times S_{\lfloor x_c \rfloor - 1, \lfloor y_c \rfloor} + C_2(S_v) \times S_{\lfloor x_c \rfloor - 1, \lfloor y_c \rfloor + 1}\right) - \left(C_0'(S_v) \times S_{\lfloor x_c \rfloor - 1, \lfloor y_c \rfloor} + C_3'(S_v) \times S_{\lfloor x_c \rfloor - 1, \lfloor y_c \rfloor + 2}\right) \quad (8)$$

The complete structure of the Horizontal Interpolator [13] is shown in figure 5.

The 2-Dimensional values of $2E_C^R$ and $2E_C^L$ can be intended as

$$2E_C^R = 2I_{\lfloor x_c \rfloor + 1, y_c} - I_{\lfloor x_c \rfloor + 2, y_c} - I_{\lfloor x_c \rfloor, y_c}$$
$$2E_C^L = 2I_{\lfloor x_c \rfloor + 1, y_c} - I_{\lfloor x_c \rfloor + 2, y_c} - I_{\lfloor x_c \rfloor, y_c} \quad (9)$$

Equation (5) can be reorganized [13] as

$$T_{x_c,y_c} = (1 - S_h) \times I_{\lfloor x_c \rfloor,y_c} + S_h \times I_{\lfloor x_c \rfloor+1,y_c} + [(1 - S_h)(2E^R_{C-1}) + S_h(2E^R_C)]S_h(1 - S_h)] \quad (10)$$

Where, $(1 - S_h) \times I_{\lfloor x_c \rfloor,y_c} + S_h \times I_{\lfloor x_c \rfloor+1,y_c} = I_{\lfloor x_c \rfloor,y_c} - S_h \times I_{\lfloor x_c \rfloor,y_c} + S_h \times I_{\lfloor x_c \rfloor,y_c}$

$$= I_{\lfloor x_c \rfloor,y_c} + S_h \times (I_{\lfloor x_c \rfloor+1,y_c} - I_{\lfloor x_c \rfloor,y_c}) \quad (11)$$

The computation of $2E^R_C$ can be reorganized as,

$$2E^R_C = 2I_{\lfloor x_c \rfloor+1,y_c} - I_{\lfloor x_c \rfloor+2,y_c} - I_{\lfloor x_c \rfloor,y_c}$$
$$= \left(I_{\lfloor x_c \rfloor+1,y_c} - I_{\lfloor x_c \rfloor,y_c}\right) - (I_{\lfloor x_c \rfloor+2,y_c} - I_{\lfloor x_c \rfloor+1,y_c})$$

Let, $D_{\lfloor x_c \rfloor,y_c} = I_{\lfloor x_c \rfloor+1,y_c} - I_{\lfloor x_c \rfloor,y_c}$

$D_{\lfloor x_c \rfloor+1,y_c} = I_{\lfloor x_c \rfloor+2,y_c} - I_{\lfloor x_c \rfloor+1,y_c}$

$$2E^R_C = D_{\lfloor x_c \rfloor,y_c} - D_{\lfloor x_c \rfloor+1,y_c} \quad (12)$$

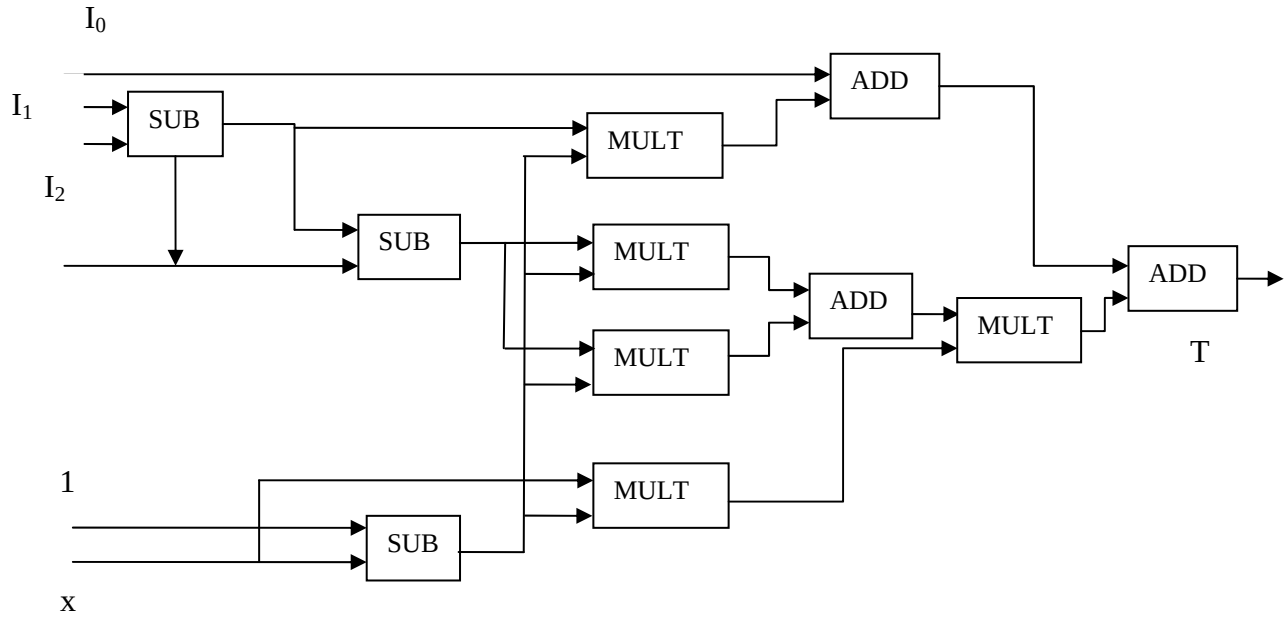

Figure 5: Architecture of Horizontal Interpolator

Equation (10) can be modified [13] as,

$$T_{x_c,y_c} = I_{\lfloor x_c \rfloor,y_c} + S_h \times D_{\lfloor x_c \rfloor,y_c} + (CS_h \times 2E^R_{C-1} + S_h \times 2E^R_C) \times S_h \times CS_h \quad (13)$$

Where, $x_c \rightarrow$ is the horizontal coordinate position from CA. $S_c = x_c - \lfloor x_c \rfloor$, and $CS_h = 1 - S_h$.

C. *The Proposed Hardware Architecture*

In order to design a less area and low power architecture an optimized architecture of decreasing computational complexity is proposed as shown in figure 6.

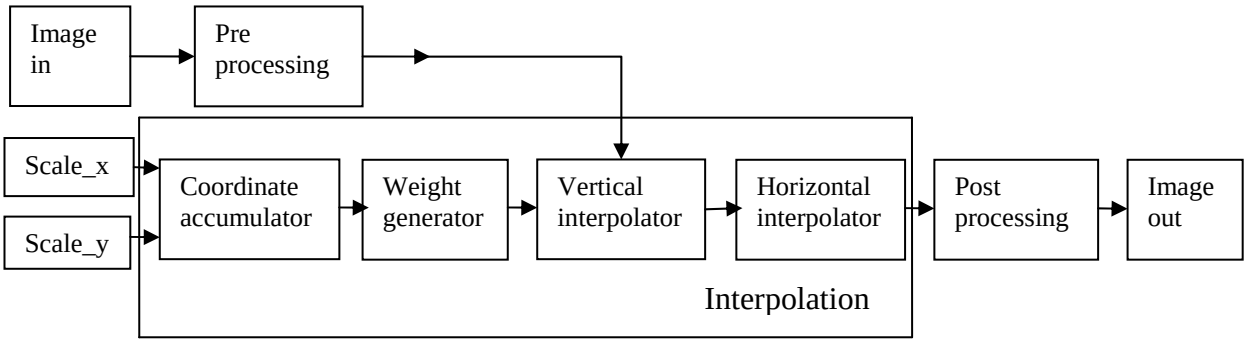

Figure 6: Block diagram of the proposed interpolation method

Image preprocessing in Matlab simulink helps in providing input to FPGA as specific test vector array which is suitable for FPGA bitstream compilation using system generator. The block diagram of image pre-processing [26] is shown in Figure 7.

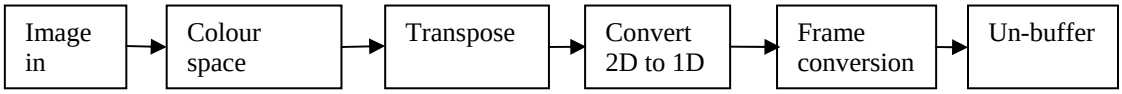

Figure 7: Block diagram of preprocessing

The proposed architecture has no line buffer and reorder module to provide the source image pixels to the vertical interpolation unit. But the existing architecture [13] uses four number of line buffers and a reorder module. So that by eliminating the line buffers in the interpolation architecture the computational complexity is very much reduced from the existing method and it also reduced the number of logics or number of LUTs for the FPGA implementation.

Image post processing helps recreating image from 1D array. Figure 8 shows the block diagram of post-processing operation [26] used to produce the output image.

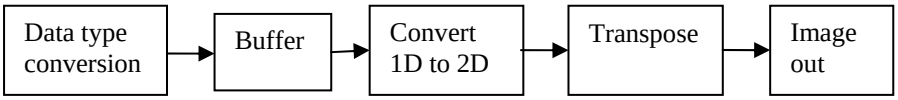

Figure 8: Block diagram of post processing

IV. Result and Discussion

The performance of scaling algorithm is evaluated based on two categories such as quality measure and performance measure. Quality measure specifies the performance of the algorithm based on metrics such as Mean Squared Error and Peak Signal to Noise Ratio. Performance measure specifies the computational complexity of image interpolation algorithms. Here quality measure and performance measure of NICFMA [13] and the proposed method are evaluated by using MATLAB simulation and MATLAB Simulink with Xilinx System Generator respectively.

First, the image quality of the scaled images is analysed for NICFMA method. This algorithm is simulated in MATLAB simulation tool for the selected gray scale/colour images of size 512x512 in USC-SIPI database. Two different quality analysis measurements are used on these test images. In order to obtain these measures, the selected images are first down sampled by a factor of 0.5 and then magnified by a factor of two. The measurements are Mean Squared Error (MSE) and Peak Signal to Noise Ratio (PSNR).

Then the computational complexity of image interpolation algorithms are calculated for the two methods and the comparison was made. To measure the computational complexity, the image interpolation circuit of two different methods such as NICFMA [13] and the proposed method are simulated using Xilinx System Generator and synthesized using Xilinx ISE simulation tool. The specifications such as summary of device utilization (Number of slice registers, Number of Slice LUTs, Number of occupied Slices), memory usage and power consumption are found out from the synthesis report.

A. Database

For testing the quality of scaling algorithms, images from USC-SIPI database of size (512 × 512) are selected and coded in MATLAB and executed. The USC-SIPI image database is a collection of digitized images. The database is divided into volumes based on the basic character of the pictures. Images in each volume are of

various sizes such as 256x256 pixels, 512x512 pixels, or 1024x1024 pixels. All images are 8 bits/pixel for black and white images and 24 bits/pixel for colour images.

Figure 9 shows the test images from the USC-SIPI database considered for experimentation. The images are cameraman, boat, lena, stream and bridge, girl, san-diego (aerial) , house, mandrill, pepper and splash.

Figure 9: Input images considered for experimentation

These test images are coded using Matlab simulation tool and tested for various scaling ratios. Of the selected ten images from USC-SIPI database, simulation output of "Splash" image which is one of the test images for downscaling and upscaling factors of ½ and 2 respectively are shown in Figure 10.

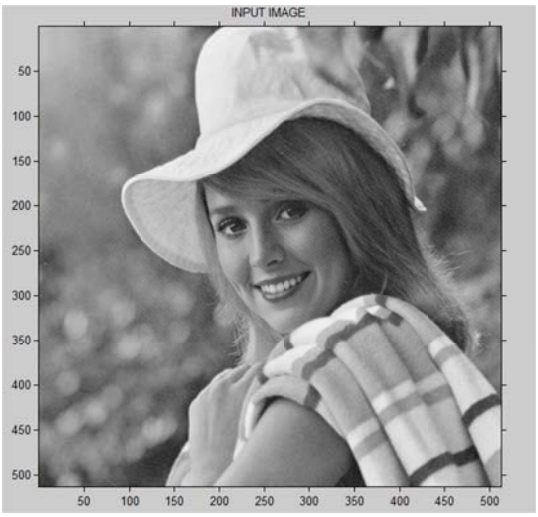

Figure 10.a Original image of girl (512x512)

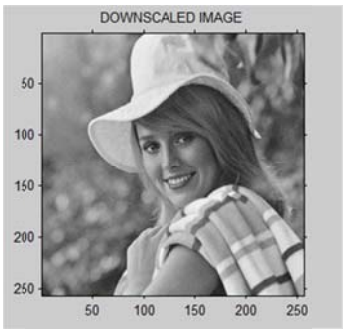

Figure 10.b Downscaled image of girl by 0.5

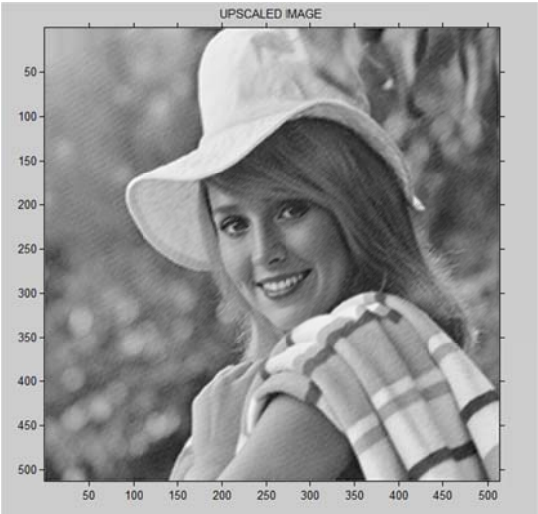

Figure 10.c Upscaled image of girl by 2

B. Quality Measure

Quality measure is the figure of merit used for the evaluation of image processing techniques. A good objective quality measure reflects the distortion on the image due to blurring, noise, compression, and sensor inadequacy. The two different qualitative analysis measurements were used on the selected test images to analyse the quality of the scaling algorithms. In order to obtain these measures, the selected images were first down sampled by a factor of 0.5 and then magnified by a factor of two. The measurements are: Mean Squared Error (MSE) and Peak Signal to Noise Ratio (PSNR).

Mean Squared Error is the difference between values implied by an estimator and the true values of the quantity being estimated. It measures the average of the squares of the errors.

Peak Signal to Noise Ratio is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is usually expressed in terms of the logarithmic decibel scale. The PSNR is most commonly used measure to determine the quality of scaled images. Higher the value of Peak Signal to Noise Ratio indicates the quality of scaled image is better. The measured MSE and PSNR values are listed in table I.

TABLE I
Quality Metrics of Various Test Images (512 x 512) for Upscaling Ratio of 2 after 1/2 Downscaling

Image	Quality Measure	
	MSE	PSNR
Cameraman	374.6768	51.5646
Boat	196.3020	58.0287
Lena	90.1179	65.8141
Stream and Bridge	451.1590	49.7071
Girl	84.2520	66.4872
San-diego (aerial)	400.2071	50.9054
House	265.4787	55.0099
Mandrill	336.3772	52.6429
Pepper	129.1797	62.2132
Splash	40.1915	73.8887

Table I shows the quality analysis measurements of various test images for varying- scaling ratios using convolution based interpolation algorithm [13]. The results shown in Table I demonstrate that the MSE is low and PSNR value is high for the image "Splash". Based on this experimental result, we know that this interpolation [13] is mostly suited for multimedia applications.

B. Performance Measure

Performance measure calculates the computational burden and memory requirements of the scaling technique. It includes the overall device utilization summary and total memory usage of weighting coefficient circuit, as the computational complexity is directly tied up to the weighting coefficient circuit for scaling algorithms.

To evaluate the performance of proposed image interpolation, the proposed architecture and the existing architecture [13] are simulated, synthesized and implemented for Virtex 6 xc6vsx315t by using MATLAB Simulink

and Xilinx ISE 13.2. The performance measure can be done by estimating resource utilization factors, timing to perform the operation and power consumption by the circuit.

The various device utilization factors are estimated by synthesizing the existing method [13] and proposed method. The device (FPGA) selected for this purpose is Virtex 6 vsx315tff1156-3 and the generated HDL by the Xilinx System Generator is synthesized by Xilinx XST.

TABLE II
Comparison of VLSI Design Parameter of Proposed Method with interpolation based on NICFMA [13]

Parameter	Interpolation Method	
	Based on NICFMA [13]	Proposed
Logics (LUTs)	364	336
LUT Flipflops	387	336
Slice Registers	64	16
Bonded IOBs	49	49
IOs	50	50
Occupied Slices	102	84

Table II shows that the comparison of different device utilization factors between interpolation with line buffer and reorder module [13] and interpolation with without buffer and reorder module [proposed]. Based on table II, the proposed method utilizes less number of LUTs (336 out of 196,800). Figure 11 shows the technology schematic of the proposed method generated by Xilinx XST.

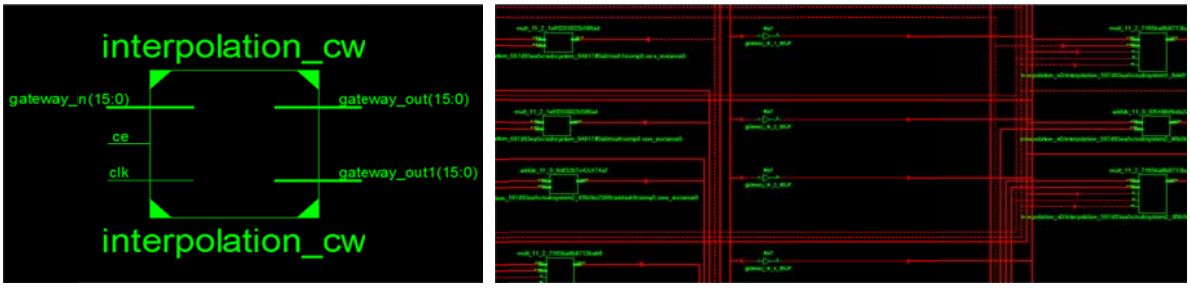

Figure 11: Technology Schematic

Table III shows the number of arithmetic elements used in existing method [13] and proposed method.

TABLE III
Comparison of Combinational Elements of Proposed method with NICFMA

Operation	Interpolation method	
	NICFMA [13]	proposed
Addition	14	7
Subtraction	8	8
Multiplication	11	11
Multiplexer	4	0

Based on the table III, the proposed method requires less number of additions (7 numbers) and there is no multiplexer in the proposed method. Thus the proposed VLSI architecture has reduced the number of components of interpolation scheme and provided an area efficient hardware circuit.

The timing (speed) of the both NICFMA [13] and the proposed method is verified after synthesizing the circuit.

TABLE IV
Comparison of Timing of Proposed Method with interpolation based on NICFMA

Parameter	Interpolation Method	
	Based on NICFMA [13]	Proposed
Combinational path delay (ns)	0.651	0.353
CPU time (sec)	10.52	9.69
Power (W)	4.315	4.307

Table IV shows the maximum combinational path delay i.e., time required to propagate the signal from input to output, total real time to execute the operation and time taken by the CPU. As shown in the table IV, the proposed method has less combinational path delay (0.353 ns). Thus, the proposed method can be used to high performance operation.

Both NICFMA [13] and the proposed image interpolation methods are implemented for the selected FPGA by using Xiling ISE. The Xilinx ISE performs translate, mapping, place and route the net-list created by XST. All the mapping and place and route reports are analysed to identify the optimized area utilization factors and power analysis is also done by using power analyser. The area utilization factors like number of occupied slices and IOs are added in table II for investigation. As shown in table II the proposed method occupies less number of slices (84 out of 49, 200) in the selected FPGA.

Table V
Power consumption

Parameter	Interpolation Method	
	Based on NICFMA [13]	Proposed
Power (Watts)	4.335	4.322

The power analyser shows that both the NICFMA [13] and proposed method consume almost similar power around four Watts as shown in table V. Figure 12 shows that the view of floor planning and figure 13 shows that the schematic of routing and IO planning for the selected FPGA device to implement the proposed design.

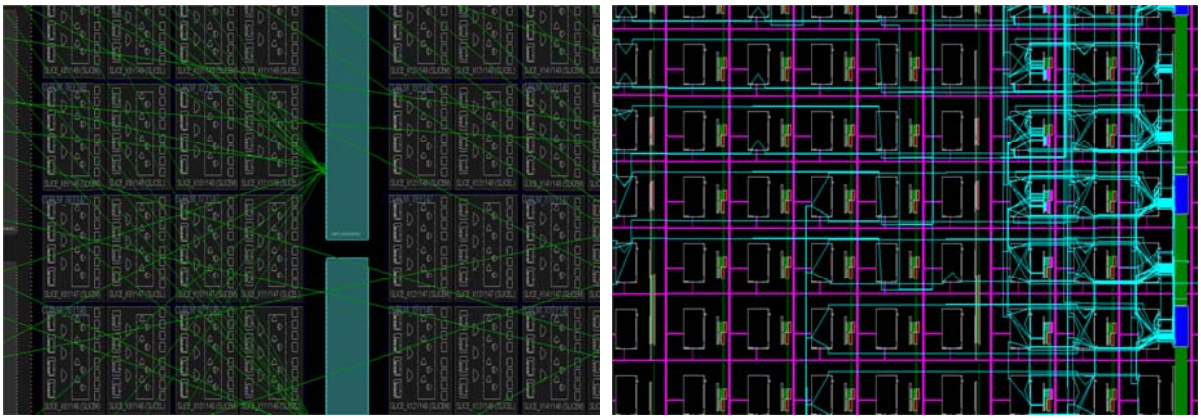

Figure 12: View of floor planning

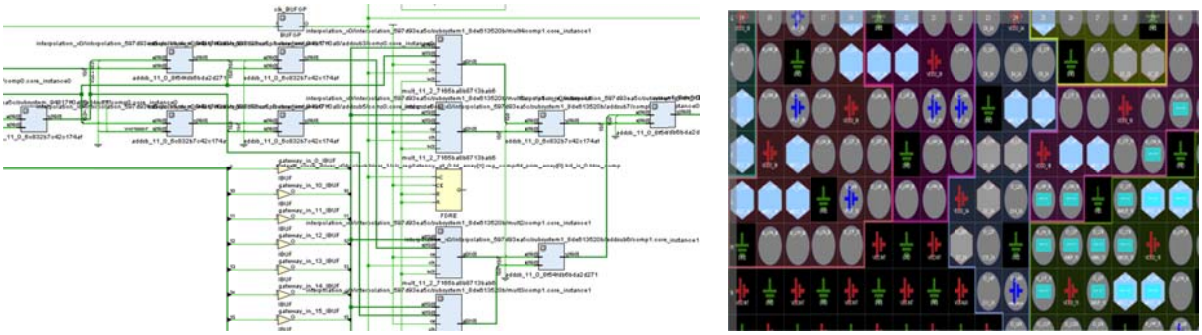

Figure 13: Schematic of routing and IO planning

The synthesized and implemented results, shown in Table II and IV describe that the proposed architecture has utilized only 366 logics (LUTs) at 0.353 ns combinational path delay. But the existing architecture [13] requires 364 logics (LUTs) at 0.651 ns combinational path delay. In addition, the other device utilization parameters such as slice registers, LUT flipflops are also compared as shown in table II. Further, the number of arithmetic operations required for both the architectures are compared. Table III describes that the proposed architecture has decreased the number of additions to seven from 11 of the existing architecture [13]. The results demonstrate that the proposed architecture for image interpolation is more area efficient than the other interpolation algorithms.

V. CONCLUSION

This paper proposes an area efficient VLSI architecture of convolution kernel based interpolation for digital image scaling. The operation of proposed architecture requires seven additions, eight subtractions and 11 multiplications with single line buffer and no reorder module. Therefore, the number of arithmetic elements in the proposed architecture is much less than the other interpolation methods. Further, the proposed method can be implemented without using line buffers. Consequently, the proposed VLSI architecture has solved the problem of computation complexity of image interpolation and furthermore, simplified the hardware for FPGA implementation and reduced the chip area. The architecture works with less area, but it may be extended for working with high accuracy.

REFERENCES

[1] Juelin Leng, Guoliang Xu and Yongjie Zhang, "*Medical Image Interpolation based on Multi-resolution Registration*", Computers and Mathematics with Applications 66, ELSEVIER, 2013, pp. 1-18.

[2] C. John Moses, Dr. D. Selvathi and S. Sajitha Rani, "*FPGA Implementation of an Efficient Partial Volume Interpolation for Medical Image Registration*", IEEE Conference proceeding ICCCCT-10, 2010, pp. 132-137

[3] C. John Moses, Dr. D. Selvathi, J. Perpet Beena and S. Sajitha Rani, "*FPGA Accelerated Partial Volume Interpolation*", IEEE Conference proceeding of ICETECT 2011, pp. 816-819.

[4] Francisco Cardells-Tormo and Jordi Arnabat-Benedicto, "*Flexible Hardware_Friendly Digital Architecture for 2-D Separable Convolution Based Scaling*", IEEE Transactions on Circuits and Systems-II, vol. 53, no. 7,July 2006, pp 522-526.

[5] Angelos Amanatiadis, Ioannis Andreadis and konstantinos Konstantinidis, "*Design and Implementation of a Fuzzy Area-Based Image-Scaling Technique*", IEEE Transaction on Instrumentation and Measurements, vol.57, no.8, August 2008, pp. 1504-1513

[6] Christine M. Zwartakes, "*Segment Adaptive Gradient Angle Interpolation*"' IEEE Transaction on Image Processing, vol. 22, no.8, August 2013, pp. 2960-2969.

[7] Shao-Hua Hong, Trien-kien Truong and Tsung-Ching Lin, "*Novel Approach to the Parametric Cubic-Spline Interpolation*", IEEE Transaction on Image processing vol.22, no.3, March 2013, pp. 1233-1241

[8] Karis S. Ni and Truong Q Nguyen, "*An Adaptable K-Nearest Neighbors Algorithm for MMSE Image Interpolation*", IEEE Transaction on Image Processing, vol.18, no. 9, September 2009, pp. 1976-1987.

[9] Zhe Wei and Kai-Kuang Ma, "*Contrast Guided Image Interpolation*", IEEE Transaction on Image processing, vol.22, no. 11, November 2013, pp. 4271-4285.

[10] Chung-chi Lin, Ming-hwa Sheu, Huann-keng Chiang, Wen-kai Tsai, Zeng-chuan Wu, "*Real-Time FPGA Architecture of Extended Linear Convolution for Digital Image Scaling*", International conference on ICECE Technology 2008, pp: 381-384.

[11] Youngjoon Cha and Seongjai Kim, "*The Error-Amended Sharp Edge (EASE) scheme for Image Zooming*", IEEE Tranaction on Image Processing, vol.16, no. 6, June 2007, pp. 1496-1505.

[12] Shih-Lun Chen, "*VLSI implementation of a Low cost High quality Image Scaling Processer*" IEEE Transactions on circuit and system: Express briefs, vol. 60, No. 1 January 2013, pp. 31-35.

[13] Chien-Chuan Huang, Pei-Yin Chen, Member, IEEE, and Ching-Hsuan Ma, "*A Novel Interpolation Chip for Real-Time Multimedia Applications*", IEEE Transactions on circuits and systems for video technology, Vol.22, No.10, October 2012, pp. 1512-1525.

[14] Chun-Ho Kim, Si-Mun Seong and Jin-Aeon Lee-sup Kim "*Winscale: an image scaling algorithm using an area pixel model*" IEEE Transactions on circuits and systems for video technology, vol. 13, no. 6, 2003, pp. 549-553

[15] Shih-Chang Hsia, Ming-Huei Chen and Po-Shien Tsia, "*VLSI Implementation of Low-power High-Quality Color Interpolation Processor for CCD Camera*, IEEE Transaction on VLSI Systems, vol. 14, April 2006, pp. 361-369.

[16] Pei-Yin Chen, Chih-Yuan Lien and Chi-Pin Lu, "VLSI *Implementation of an Edge-Oriented Image Scaling Processor*" IEEE Transactions on very large scale integration (VLSI) systems, vol. 17, no. 9, September 2009, pp.1275-1284.

[17] Chung-chi Lin, Ming-hwa Sheu, Huann-keng Chiang, Chishyan Liaw and Zeng-chuan Wu, "*The Efficient VLSI Design of BI-CUBIC Convolution Interpolation for Digital Image Processing*" in Proc. IEEE Int. Symp. Circuits Syst., 2008, pp. 480-483.

[18] Chung-chi Lin, Ming-hwa Sheu, Huann-keng Chiang, Wen-kai Tsai and Zeng-chuan Wu, "*Real-Time FPGA Architecture of Extended Linear Convolution for Digital Image Scaling*", International conference on ICECE Technology 2008, pp: 381-384.

[19] Shih-Lun Chen, Hong-Yi Huang, and Ching-Hsing Luo, " *A Low-Cost High-Quality Adaptive Scalar for Real-Time Multimedia Applications*" IEEE Transactions on circuits and systems for video technology, vol. 21, no. 11, November 2011, pp. 1600-1611.

[20] Chung-chi Lin, Chishyan Liaw and Ching-tsorng Tsai, "*A Piecewise Linear Convolution Interpolation with Third-order Approximation for Real-time Image Processing*", IEEE Conf 2010, pp. 3632-3637.

[21] Chung-chi Lin, Ming-hwa Sheu, Chishyan Liaw, and Huann-keng Chiang, "*Fast First-order Polynomials Convolution Interpolation for Real-Time Digital Image Reconstruction*", IEEE Transactions on Circuits and Systems for Video Technology, vol.20, no.9, September 2010, pp. 1260-1264.

[22] Shin-Lun Chen, "*VLSI Implementation of an Adaptive Edge-Enhanced Image Scalar for Real-Time Multimedia Applications*", IEEE Transactions on Circuits and Systems for Video Technology, vol. 23. No. 9, September 2013, pp. 1510-1522.

[23] Chao-Lieh and Chien-Hao Lai, "*Iterative Linear Interpolation Based on Fuzzy Gradient Model for low-cost VLSI Implementation*", VLSI Systems, 2013, DOB: 10.1109/TVLSI.2013-2276410 (IEEE early access article).

[24] Hsieh S Hou , Harry C Andrews "*Cubic splines for image interpolation and digital filtering*" IEEE Transactions on acoustics, speech and signal processing , vol. ASSP-26, no.6, December 1978.

[25] Chung-Chi Lin, Ming-Hwa Shen, Huann-Keng Chiang, Chishyan Liaw, Zeng-Chuan Wu and Wen-Kai Tsai, "*An Efficient Architecture of Extended Linear Interpolation for Image processing*", Journal of Information Science and Engineering 26, 2010, pp. 631-648.

[26] S. Allin Christe, M. Vignesh, Dr. A. Kandaswamy, "*An Efficient FPGA Implementation of MRI Image Filtering and Tumour Characterization Using Xilinx System Generator"*, International Journal of VLSI Design and Communication Systems (VLSICS), Vol.2, No.4, December 2011, pp. 95-109.