A High Quality Image Scaling Processor With Reduced Memory

Amal Mole .S[1], Sarath Raj .S[2]

[1]P G Scholar, VLSI and Embedded Systems, Department of ECE, T K M Institute of Technology, Kollam, India

[2]Associate Professor, Department of EBE, T K M Institute of Technology, Kollam, India

Abstract*: The digital images can be resized and the process of doing it is called as image scaling. The applications such as sharpening of the image, image zooming, processing edge structures in an image etc, uses image scaling as one of its important method. Image scaling is a computationally expensive operation. High memory requirement and computation complexity are characteristics of most of the high quality image scaling algorithms. For very large scale integration (VLSI) implementation, low complexity and low memory requirement image scaling algorithms are necessary. Here, the image scaling algorithm consists of sharpening spatial filter, clamp filter and simplified bilinear interpolation. The sharpening spatial filter and clamp filter serves as pre-filters prior to bilinear interpolation operation. These filters are combined into a combined filter by the 2D convolution of T- model or inverse T- model convolution kernels that represent them. The filter combining technique reduces computation resources and memory buffer. Hardware sharing techniques are used to reduce the computational complexity and computing resource needed. Bilinear interpolation is an image restoring algorithm. It is popularly used in VLSI implementation because of its low complexity and simple architecture. The architecture can be modeled in Verilog HDL, simulated using ModelSim XE III 6.3c and synthesized using Xilinx ISE design suite 8.2i and can be implemented in Spartan 3 FPGA.*

Keywords: Bilinear interpolation, clamp filter, 2D- convolution, image scaling, sharpening spatial filter, very large scale integration (VLSI).

1. Introduction

Image scaling is the resizing of digital images, wherein interpolation techniques are used to achieve an optimum between factors such as efficiency and smoothness. It can be separated into two different operations – reconstruction of image and re-sampling at output grid rate. Nowadays, images of different sizes and formats are available to users from different sources such as mobile phones, digital camera and internet. With the emerging trends in multimedia, there exists demand of outstanding image scaling techniques. Digital image scaling applications ranges from consumer electronics to medical imaging.

Image scaling algorithms convert image of one resolution to another without losing the visual content. They can be classified into polynomial based methods and non-polynomial based methods. Polynomial based methods are based on direct manipulation on pixels. They are easy to perform, require less calculation cost and follow same pattern for all pixels. An uncomplicated and simple polynomial based method is the nearest neighbor algorithm which gives good result when image has high resolution pixels. But some information at edges are lost. The most popular in implementation of VLSI chips is the bilinear interpolation algorithm, due to its simple architecture and low complexity. It linearly interpolates four nearest neighbor pixels of an un-restored image to obtain pixel of a restored image as a forward function. However, its high frequency response behavior is poor. The best among all polynomial based methods is the bi-cubic interpolation that gives sharper image compared to others, but requires more computation time.

The polynomial based methods stores the low frequency components of the original image and causes blocking and aliasing artifacts. The image must preserve high frequency components for better visual quality. Many non- polynomial based methods have been proposed in recent years. They provide better result and consider features like intensity value, edge information, texture etc. The bilateral filter, curvature interpolation, data dependent triangulation, autoregressive model, new edge directed interpolation, iterative curvature based interpolation are some efficient techniques used by non-polynomial methods to enhance image quality and to reduce blocking, aliasing and blurring effects. These image scaling algorithms have high complexity and memory requirement.

The complexity/ latency of the hardware architecture is determined by the interpolation technique used. It is difficult to implement image scaling algorithms of high complexity and memory requirement using VLSI technology. For cost and time to market reasons many real time scaling applications uses traditional low complexity image scaling algorithms to implement in VLSI technology.

The design and implementation of image processing algorithms in VLSI is an expanding area of research. The complexity of VLSI design is the main obstacle that blocks the widespread use of it in real time image processors. In this work a high quality algorithm with low complexity and low memory is used. To reduce the memory requirement and computation cost filter combining, hardware sharing and reconfigurable techniques had been used in the scaling algorithm. Due to computational efficiency and qualitative stability bilinear interpolation algorithm is selected by trading off complexity and quality. Because of its low complexity and simple architecture bilinear interpolation is efficient for VLSI implementation. The coding can be synthesized using Xilinx ISE Design Suite 8.2i, simulated

Licensed Under Creative Commons Attribution CC BY

using ModelSim XE III 6.3c and can be implemented using Spartan 3 FPGA.

2. Related Works

To achieve demand of real time image scaling applications some low complexity methods for VLSI implementation have been proposed. Winscale [8] image interpolation is implemented by using area pixel model for image scaling. This method has high frequency and image quality than bilinear interpolation method. An edge oriented image scaling processor [7] with low complexity VLSI architecture uses a simple edge catching technique for edge preservation and to achieve better image quality. The hardware architecture of this algorithm uses a single line buffer memory.

A low cost high quality adaptive scalar [4] for real time multimedia application adopts bilinear interpolation algorithm. The bilinear interpolation algorithm is simplified by hardware sharing technique to reduce hardware cost and computing resource. The work utilizes a total of four line buffer memory. VLSI implementation of an adaptive edge-enhanced image scalar [3] for multimedia applications uses edge detector with low cost edge catching technique and is based on bilinear interpolation algorithm. Sharpening spatial filter is used to reduce the blurring effect. By using one line buffer the design can process streaming data.

3. Methodology

The image scaling algorithm consists of a sharpening spatial filter, clamp filter and bilinear interpolation. The basic block diagram for the scaling algorithm for image zooming is shown in the Figure 1.

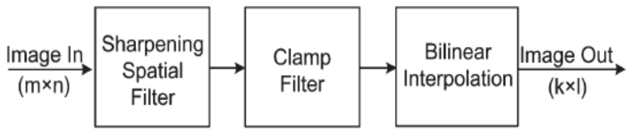

Figure 1: Block diagram of scaling algorithm for image-zooming.

Image in is the input image of size (m×n) and image out is the scaled output image of size (k×l). The input pixels of source image are first given to sharpening spatial filter [5], which removes associated noise and enhances the edges. These filtered pixels are again filtered by clamp filter [5], to reduce aliasing artifacts and to smooth unwanted discontinuous edges of boundary region. These filters serve as pre-filters prior to the bilinear interpolation operation and reduce the blurring and aliasing artifacts. Finally, these filtered pixels are bilinear interpolated for performing scaling operation. T-model or inverse T-model convolution kernels are used for the realization of sharpening spatial filter and clamp filter that reduces memory buffers and computation cost.

3.1 System Overview

The VLSI architecture of the real time image scaling processor consists of four main blocks. They are: a register bank, a combined filter, simplified bilinear interpolator and a controller. In the scaling algorithm, the source image is filtered by a sharpening spatial filter and then filtered again by a clamp filter. In the architecture of the image scaling algorithm, these two pre-filters are combined to form a combined filter. The block diagram for the VLSI architecture of image scaling processor is shown in Figure 2.

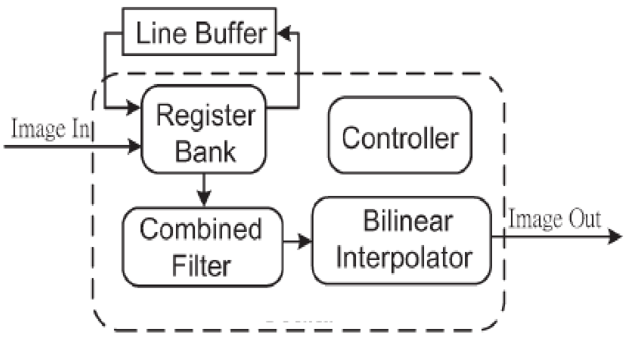

Figure 2: Block diagram of the VLSI architecture for real-time image scaling processor.

The details of each block will be discussed in the following sections.

3.2 Register Bank

The register bank consists of ten shift registers and designed along with a one line buffer memory. The register bank along with this line buffer provides ten values for the immediate usage of combined filter. When the controller produces the clocking and reset signals, a new value will be read into register Reg 41 and the stored value of the row n+1 in each register will get shifted into next register or to the line buffer memory. The Reg 40 reads a new value from the line buffer memory and each value in row n, that is stored in other registers gets shifted to the next register. The architecture of the register bank is shown in Figure 3.

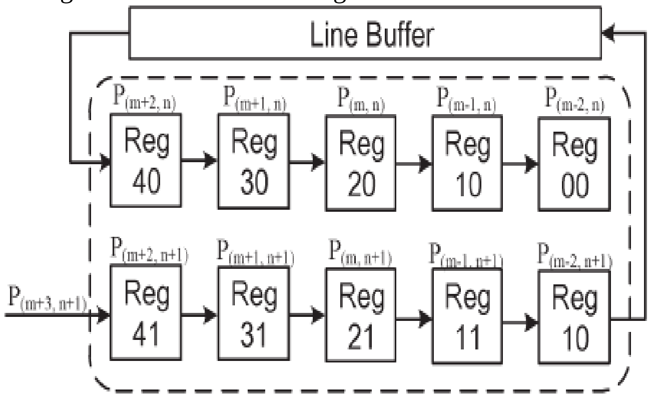

Figure 3: Block diagram for the architecture of the register bank.

3.3 Combined Filter

The sharpening spatial filter and clamp filter can be represented by convolution kernels [4]. Convolution kernel is the matrix of weights. The image quality can be increased if a large sized convolution kernel is used. But this increases the hardware cost and memory requirement. For example, the use of a 3×3 convolution sharpening spatial filter and 3×3 convolution clamp filter produces a 5×5 combined filter is shown in Figure 4.

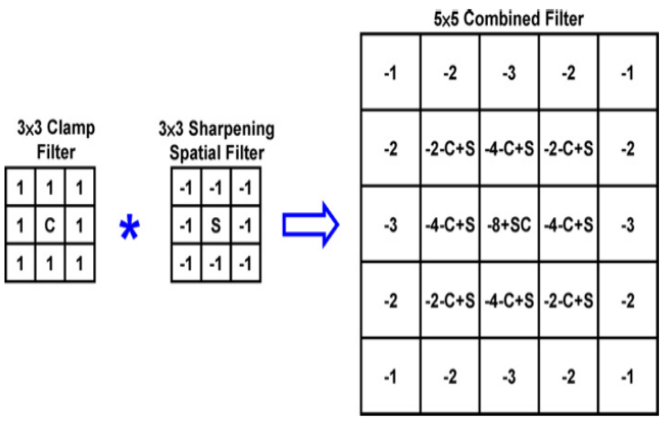
5x5 Combined Filter
3x3 Clamp Filter
3x3 Sharpening Spatial Filter
1 1 1
1 C 1
1 1 1
-1 -1 -1
-1 S -1
-1 -1 -1
-1 -2 -3 -2 -1
-2 -2-C+S -4-C+S -2-C+S -2
-3 -4-C+S -8+SC -4-C+S -3
-2 -2-C+S -4-C+S -2-C+S -2
-1 -2 -3 -2 -1

Figure 4: 3×3 clamp filter combined with 3×3 sharpening spatial filter to form 5×5 combined filter

Where S and C are the sharpening and clamp filter parameters. This requires four line buffer memory and twenty five arithmetic units. To reduce the complexity the 3×3 convolution kernel can be replaced by a cross model which cuts down four parameters. For further improvement a T-model or inverse T-model convolution kernel can be used to design the filters. Then, two line buffers are required to store input data or intermediate values of filtering. The filter combining technique can be used to decrease the memory requirement to one line buffer and computation cost can also be decreased. The 3×3, cross model and T-model convolution kernels are shown in Figure 5.

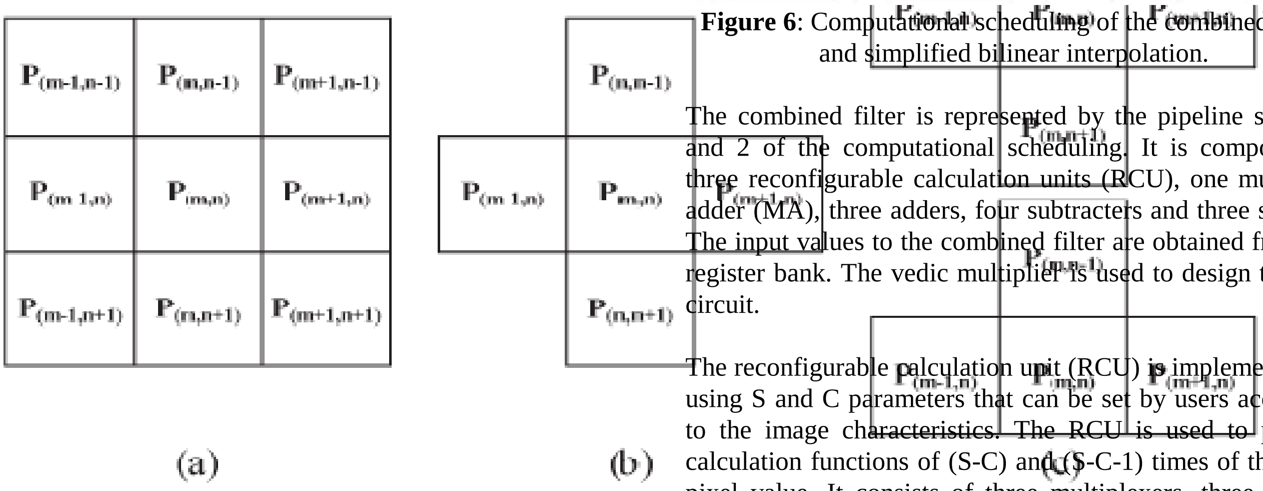

Figure 5: Weights of convolution kernels (a) 3×3 convolution kernel. (b) cross model convolution kernel. (c) T-model and inverse T-model convolution kernels.

The pre-filters can be combined as,

$$P'_{(m,n)}=\left[P^{*}_{(m,n)}\begin{bmatrix}-1 & S & -1\\ & -1 & \end{bmatrix}/(S-3)\right]^{*}\begin{bmatrix}1 & C & 1\\ & 1 & \end{bmatrix}/(C+3)$$

$$=P^{*}_{(m,n)}\begin{bmatrix}-1 & S-C & SC-2 & S-C & -1\\ & -2 & S-C & -2 & \\ & & -1 & & \end{bmatrix} \quad (1)$$

$$/[(S-3)\times(C+3)]$$

$$\approx P^{*}_{(m,n)}\begin{bmatrix}-1 & S-C & SC-2 & S-C & -1\\ & -2 & S-C & -2 & \end{bmatrix} \quad (2)$$

$$/[(S-3)\times(C+3)]$$

Where S and C are the sharpening and clamp filter parameters, $P'_{(m,\ n)}$ is the filtered pixel and $P_{(m,\ n)}$ is the source pixel to be filtered by the combined filter. With the combined filter gain, the results need to be divided once. But, the gain can be eliminated by a shifter. The computational scheduling of the combined filter and bilinear interpolator is shown in Figure 6.

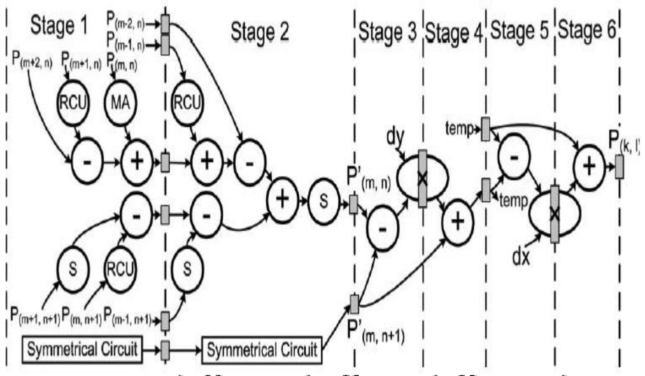

Figure 6: Computational scheduling of the combined filter and simplified bilinear interpolation.

The combined filter is represented by the pipeline stages 1 and 2 of the computational scheduling. It is composed of three reconfigurable calculation units (RCU), one multiplier adder (MA), three adders, four subtracters and three shifters. The input values to the combined filter are obtained from the register bank. The vedic multiplier is used to design the MA circuit.

The reconfigurable calculation unit (RCU) is implemented by using S and C parameters that can be set by users according to the image characteristics. The RCU is used to provide calculation functions of (S-C) and (S-C-1) times of the input pixel value. It consists of three multiplexers, three adders, four shifters and a sign block. The block diagram architecture of the RCU is shown in Figure 7.

Licensed Under Creative Commons Attribution CC BY

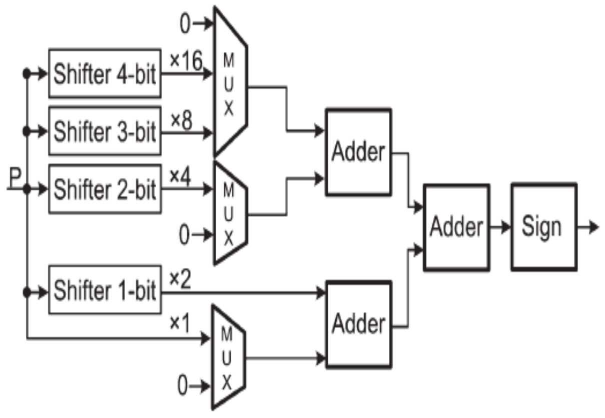

Figure 7: Block diagram for the architecture of RCU

The shifter 1-bit produces two times the input pixel value. The shifters 2, 3 and 4 –bit produces four, eight and sixteen times the input pixel values respectively. One of the multiplexers produces the result by selecting multiples of one and zero times input value and another produces the result by selecting multiples of zero and four times the input pixel value. The third multiplexer produces the result by selecting the multiples of zero, eight and sixteen times the input pixel value. The multiplexer outputs are added by adders as shown in Figure 7. The adder output is fed to the sign block for producing the sign magnitude value of its input. S and C parameter values are selected from the Table 1.

Table 1: Parameters and computing resources of RCU

Parameters	Values	Computing Resource
C	5, 13, 29	Add and Shift
S	7, 11, 19	Add and Shift
S-C	2, -6, -22, 6, -2, -18, 14, 6, -10	Add, Shift, and Sign
S-C-1	1, -7, -23, 5, -3, -19, 13, 5, -11	Add, Shift, and Sign

3.4 Simplified Bilinear Interpolation and Controller

Without losing the visual content, an image from one resolution can be converted to another resolution by using image interpolation algorithms. Bilinear interpolation is an operation that determines the intensity from weighted average of the four closest pixels to the specified input co-ordinates and then assigns that value to the output co-ordinate. Bilinear operation performs linear interpolation first in one direction and then again in the other direction. Thus it serves as an image restoring algorithm. The algorithm requires modest amount of memory. The target pixel $P_{(k,\ l)}$ can be calculated as

$$P_{(k,\ l)} = (1\text{-}dx) \times (1\text{-}dy) \times P_{(m,\ n)} + dx \times (1\text{-}dy) \times P_{(m+1,\ n)} + (1\text{-}dx) \times dy \times P_{(m,\ n+1)} + dx \times dy \times P_{(m+1,\ n+1)} \quad (3)$$

Where dx and dy are scale parameters in horizontal and vertical directions. $P_{(m,\ n)}$, $P_{(m+1,\ n)}$, $P_{(m,\ n+1)}$ and $P_{(m+1,\ n+1)}$ are the four nearest neighbor pixels of the source image.

The computation of output pixel requires eight multiply, four subtract and three addition operations. To reduce the silicon cost, algebraic manipulation is used to reduce the computation resource of bilinear interpolation. The simplifying procedure of bilinear interpolation is as follows.

$$P_{(k,\ l)} = [\,(1\text{-}dy) \times P_{(m+1,\ n)} + dy \times P_{(m+1,\ n+1)}\,] \times dx + [\,(1\text{-}dy) \times P_{(m,\ n)} + dy \times P_{(m,\ n+1)}\,]\,(1\text{-}dx) \quad (4)$$

$$= [\,P_{(m+1,\ n)} + dy \times (\,P_{(m+1,\ n+1)} - P_{(m+1,\ n)}\,)\,] \times dx + [\,P_{(m,\ n)} + dy \times (\,P_{(m,\ n+1)} - P_{(m,\ n)}\,)]\,(1\text{-}dx) \quad (5)$$

$$= \{\,[\,P_{(m+1,\ n)} + dy \times (\,P_{(m+1,\ n+1)} - P_{(m+1,\ n)}\,)\,] - [\,P_{(m,\ n)} + dy \times (\,P_{(m,\ n+1)} - P_{(m,\ n)}\,)\,]\,\} \times dx + [\,P_{(m,\ n)} + dy \times (\,P_{(m,\ n+1)} - P_{(m,\ n)}\,)\,] \quad (6)$$

By this simplification procedure, computing resources got reduced into two multiply, two subtract and two addition operations. The stage 3, 4, 5 and 6 in the Figure 6 represent the pipelined architecture of bilinear interpolation that can be directly mapped to the equation (6). The symmetrical circuit is the inverse T-model combined filter design that produce $P'_{(m,\ n+1)}$. The controller is used to generate the timing signals used to control the register bank and pipeline stages of combined filter and bilinear interpolation circuit.

4. Results and Discussion

The modules are modeled using Verilog HDL and simulated using ModelSim 6.3 III c to verify the functionality of the design. The image pixel values are obtained by using MATLAB R2009b. The pixel values are fed to the register bank. The register bank outputs are fed to the combined filter and bilinear interpolator to obtain the scaled output pixel values.

4.1 Simulation Result of Register Bank and Line Buffer Memory

The register bank is composed of ten shift registers. Along with the register bank one line buffer memory is used. The inputs to the module are clock (clk), reset (rst) and image data (im-data). The normal operation starts when clk = 1 and rst = 1. The output of lower five shift registers in register bank, are fed to the line buffer. The line buffer output again feeds the top five shift registers in the register bank. The outputs r40, r30, r20, r31, r21, r11, r00 and r10 serves as input to the combined filter. The simulated waveform of register bank and line buffer memory is shown in the Figure 8.

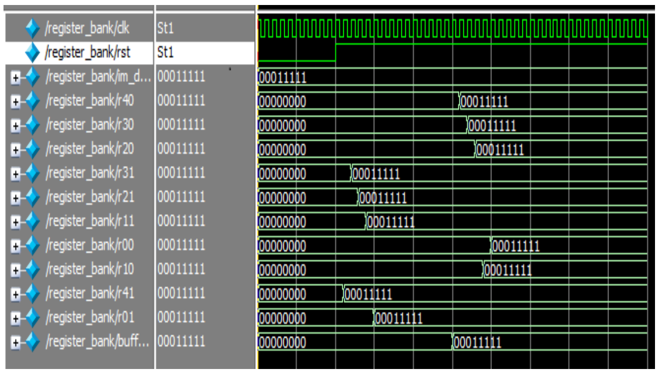
/register_bank/clk St1
/register_bank/rst St1
/register_bank/im_d... 00011111 00011111
/register_bank/r40 00011111 00000000 00011111
/register_bank/r30 00011111 00000000 00011111
/register_bank/r20 00011111 00000000 00011111
/register_bank/r31 00011111 00000000 00011111
/register_bank/r21 00011111 00000000 00011111
/register_bank/r11 00011111 00000000 00011111
/register_bank/r00 00011111 00000000 00011111
/register_bank/r10 00011111 00000000 00011111
/register_bank/r41 00011111 00000000 00011111
/register_bank/r01 00011111 00000000 00011111
/register_bank/buff... 00011111 00000000 00011111

Figure 8: Simulation result of register bank and line buffer memory

4.2 Simulation Result of Combined Filter and Bilinear Interpolation

To reduce the computation resources, the sharpening spatial filter and clamp filter that serves as pre-filters prior to bi-linear interpolation are combined to form the combined filter. Bi-linear interpolation is easy to implement in VLSI. Both these constitute the final stage in the scaling algorithm used. The inputs to this module are clock (clk), reset (rst), the pixel values that are obtained from the register bank, S and C parameters. The output is Pout. The simulated waveform of combined filter and bi-linear interpolation is shown in Figure 9.

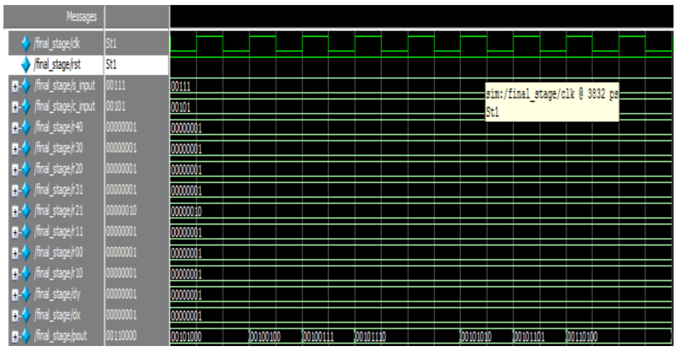

Figure 9: Simulation result of combined filter and bilinear interpolation

4.3 Simulation Result of Image Scaling processor

The register bank along with the line buffer, combined filter and bi-linear interpolation can be assembled to form the entire architecture of image scaling processor. The normal operation starts when clock (clk) = 1 and reset (rst) = 1. The other inputs to the module are the S parameter, C parameter, image data (im-data) that is given to register bank and the eight pixel values from the register bank. The scaled output is pout. Simulated waveform of image scaling processor is shown in Figure 10.

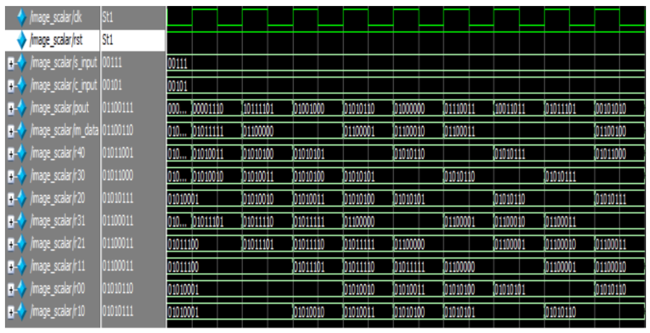

Figure 10: Simulated waveform of image scaling processor.

The RTL schematic view of the image scaling processor is shown in Figure 11.

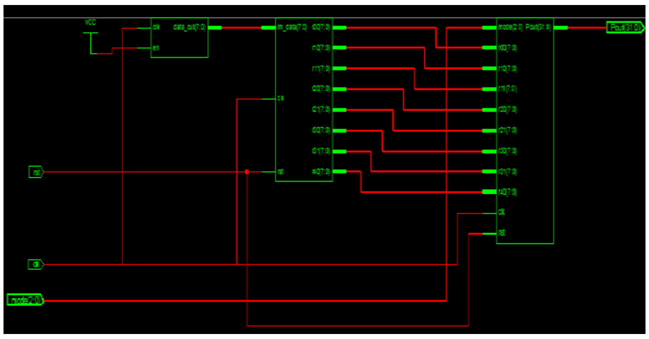

Figure 11: RTL-Schematic of image scaling processor

5. Conclusion

The project explores the design and simulation of the image scaling processor using VLSI technology. The architecture of the image scaling processor is of low cost, low memory requirement, high quality and high performance. The hardware cost had been reduced by the filter combining technique, hardware sharing and reconfigurable technique. The work also focuses to reduce the gate count and noise cancellation

References

[1] Shih-Lun Chen, “VLSI Implementation of a Low-Cost High- Quality Image Scaling Processor”, IEEE Transactions On Circuits And Systemsii: Express Briefs, Vol. 60, No. 1, January 2013.

[2] Vaishali Patel, Prof. Kinjal Mistree, “A Review on Different Image Interpolation Techniques for Image Enhancement”, International Journal of Emerging Technology and Advanced Engineering., Volume 3, Issue 12, December 2013.

[3] Shih- 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.

[4] S.L Chen, H. Y Huang, C. H Luo, “A low-cost high-quality adaptive scalar for real-time multimedia applications”, IEEE Trans. Circuits Syst.Video Technol., vol. 21, no. 11, pp. 16001611, Nov. 2011

[5] Sumit Vaidya, Deepak Dandekar, “Delay-Power Performance Comparison Of Multipliers In VLSI Circuit Design”, International Journal of Computer Networks Communications (IJCNC), Vol.2, No.4, July 2010.

[6] J. W. Han, J. H Kim, S. H Cheon, J. O Kim, S. J Ko, “A novel image interpolation method using the bilateral filter”, IEEE Trans.Consum. Electronics. , vol. 56, no. 1, pp. 175181, Feb. 2010.

[7] P.Y. Chen, C. Y. Lien and C. P. Lu, “VLSI implementation of an edge- oriented image scaling processor”, IEEE Trans. Very Large Scale Integr. (VLSI) Syst.. Vol.17, no.9, pp.1275-1284. Sep. 2009.

[8] C. H. Kim, S. M. Seong, J. A. Lee, L. S. Kim, “Winscale : An image scaling algorithm using an area pixel model”, IEEE Trans. Circuits Syst.Video Technol., vol. 13, no. 6, pp. 549553, Jun. 2003.

Licensed Under Creative Commons Attribution CC BY