ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)
On 21st & 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

VLSI Architecture of Pipelined Adaptive Edge-Enhanced Image Scalar for Image Processing Applications

G.Manoj kannan[#1],A.Manoj kumar[*2],R.Mareeswaran[#3], J.Kanimozhi[*4]

[# 1]Department of ECE, K.L.N College of Information Technology,Pottapalayam,Sivagangai District, Tamilnadu, India

[*2]Department of ECE, K.L.N College of Information Technology,Pottapalayam,Sivagangai District, Tamilnadu, India

[#3]Department of ECE, K.L.N College of Information Technology,Pottapalayam,Sivagangai District, Tamilnadu, India

[*4]Department of ECE, K.L.N College of Information Technology,Pottapalayam,Sivagangai District, Tamilnadu, India

ABSTRACT-In this paper, pipelining concepts is adopted to increase the processing speed of image scaling algorithms. The proposed algorithm consists of pipelining stages along with a linear space-variant edge detector, a low complicacy sharpening spatial filter, and a simplified bilinear interpolation. The linear space-edge detector is used to discover the image edges by a low-cost catching technique. The sharpening spatial filter serves as Pre filters in solving the blurring edges caused by the bilinear interpolation. An adaptive technology is used to enhance the edge contrast of image & improves the apparent sharpness effect of the edge detector by adaptively selecting the input pixels of the bilinear interpolation. Winscale method is used for the area pixel model. An algebraic manipulation and a hardware sharing techniques are used to simple the bilinear interpolation. Bilinear interpolation decreases the computing resources and silicon area in VLSI circuits. When compared with previous low complicacy techniques, this paper performs high speed, better quality, very good performance, less memory requirements, and a cheap hardware cost than other image scalar methods

KEYWORDS—Sharpening spatial filter, Edge detector, Bilinear interpolation, Pipelining**.**

I. INTRODUCTION

Image scalar Technique is process of resizing a digital Image that involves a trade-off between efficiency, smoothness and sharpness.Gradient of an edge has magnitude and direction. Adding laplacian to an image results in edge undershoot and overshoot. *k* factor tunes the degree of edge enhancement. It designs a low cost, high quality and high speed with a pipelined technique. The Nearest Neighbour algorithm is used for alleviation block and aliasing effects since it requires a low time complexity .The next complex method chosen is Bicubic Interpolation..Images resembled with bicubic interpolation are smoother and have fewer interpolation artifacts. [9]It is an extension of Bicubic Interpolation for Interpolating data points on a two dimensional regular grid. It has high quality but it need six line buffer memories.[9] Kim et al presented a Winscale method used an area-pixel model instead of the common point-pixel model, and uses a maximum of four pixels of the original image to calculate one pixel of a scaled image. [10]Lin et al. implemented if the edge detector obtains an efficient VLSI architecture design of winscale algorithm. It demonstrates High performance. It able to process DIS (Digital Image Scaling) for HD in Television and Real time applications. Lin et al suggested a low cost architecture based on the bicubic convolution interpolation algorithm by reducing complexity of generating weighting coefficients. Chen et al. suggested an edge-oriented area pixel scaling processor. It uses a low cost high quality image scalar and adaptive low-cost was suggested in which a sharpening spatial and a clamp 3×3 convolution filters were added as pre filters to reduce the blurring and aliasing effects produced by the bilinear Edge catching technique and seven stage pipeline architecture to achieve low cost and high performance. Line at all suggested a fast first-order polynomials convolution technique to execute an efficient image scaling design. Huang et al existent a high performance 2-D image scalar design based on an interpolation error theorem Interpolation .while the chip area and memory requirement were efficiently decreased by hardware sharing and filter combining technique, it required three multipliers, fifty adders, and a four-line-buffer memory. Thus, an adaptive edge-enhanced scaling algorithm that demands three multipliers, nineteen adders, and only a one-line-buffer memory is suggested in this paper. The suggested algorithm consists of a low-cost linear space-variant edge detector, a low-complex sharpening spatial filter, and a simplified bilinear interpolation. An algebraic manipulation technique and a hardware sharing technique are used to reduce the computing resource and hardware cost of the bilinear interpolation circuit. The sharpening spatial filter and low-cost edge detector are added to the proposed scaling circuit as pre filter to enhance the edges of the source images and decrease the blurring effects caused by the bilinear interpolation .In this pipelining concept is applied to increase the processing speed of the data .If we increase the pipelining stages, we can increase the speed of the processor.

II. Related Works

In this contrast to Image Scalar, edge detector which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. Sharpening spatial filter is used as a prefilter in solving the blurring edges caused by the Bilinear Interpolation. Bilinear Interpolation is mainly used for the Image zooming. The Register Bank is used to store the gray Scale Images. In this technique we use a pipelining technique at sharpening spatial filter, Edge detector and at multiplexer to perform the speed in a process and to reduce the critical path delay in the process .The Image is a given in a MATLAB and converted to a coding. The coding is given as a input in VLSI. So the edge detector identifies the outline and boundary of the Image. The bilinear interpolation up scaling the image ex: 64x64 to 128x128.The sharpening spatial filter is used as a prefilter to solving the blurring effects produced by the bilinear interpolation.

III. PROPOSED SYSTEM

A. Edge Detector

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed *edges*. The same problem of finding discontinuities in 1D signal is known as step detection and the problem of finding signal discontinuities over time is known as change detection. In edge detector we want to determine the edges of outline and boundary. In edges want to calculate the Intensity variations. In edge detector the pixels are determined by vertical and horizontal gradient calculation by kernel or masking.

The edge detected pictures..

The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world. It can be shown that under rather general assumptions for an image formation model, discontinuities in image brightness are likely to correspond to:

- discontinuities in depth,
- discontinuities in surface orientation,
- changes in material properties and
- Variations in scene illumination.

In the ideal case, the result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well as curves that correspond to discontinuities in surface orientation. Thus, applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified. However, it is not always possible to obtain such ideal edges from real life images of moderate complexity.

Edges extracted from non-trivial images are often hampered by fragmentation, meaning that the edge curves are not connected, missing edge segments as well as false edges not corresponding to interesting phenomena in the image – thus complicating the subsequent task of interpreting the image data.

Edge detection is one of the fundamental steps in image processing, image analysis, image pattern recognition, and computer vision techniques.

Simple Gradient Calculation

Let us represent an image by an array *A*, in which each element of the array corresponds to the gray level of an image. If the gray levels are in pixel counts, then the numbers might range from 0 to 255 for an eight-bit per pixel image.

The gradient is the change in gray level with direction. This can be calculated by taking the difference in value of neighboring pixels. Let us construct a new array *B* that contains the values of the gradient from *A*.

The horizontal gradient is formed by taking the differences between column values.

$$B(j,k) = A(j,k+1) - A(j,k)$$

This can be represented by a filter array as shown below:

-1	1

A problem with this filter is that the location of the gradient in the array *B* is shifted somewhat to the left. With an even number of pixels in the computation it is impossible to locate the result in the center of the cells used to produce it. It is therefore most common to use an odd number of cells. This can be accomplished by doing the calculation over cells that are separated by step.

$$B(j,k)=A(j,k+1)-A(j,k-1)$$

This can be represented by the array shown below:

-1	0	1

Note that the result pixel is centered between the left and right pixels used to calculate the gradient, so there is no shift in the location of the gradient result.

Horizontal edges would be detected by calculating the vertical gradient. The equation for the separated vertical difference is

$$B(j, k) = A(j+1, k)-A(j-1, k)$$

For an image in which the row coordinates are counted from the bottom edge upward, the corresponding filter array is

1
0
-1

Vertical Edges

Vertical edges can be detected by using a horizontal gradient operator followed by a threshold operation to detect the extreme values of the gradient. The gradient produces a doublet of extremes, positive-negative or negative-positive, depending on the direction of the transition.

We will investigate the detection of both vertical and horizontal edges using the image of the detail of a building shown in figure1. Detail of a building at 256 gray levels. This image has strong horizontal and vertical edges, and is therefore useful for the illustration of the method.

1

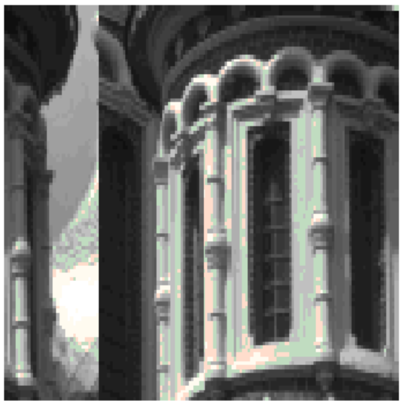


Figure 1:Detail of a building at 256 gray levels.

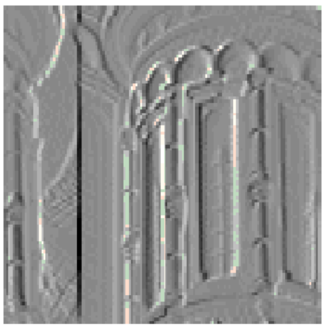


Figure 2:Image produced by the horizontal gradient calculation.

The horizontal gradient was calculated by taking differences in the image values between columns. Note that the column before and the column after *k* was used. Use of an odd number of pixels in a gradient calculation prevents a shift in location.

B (j, k) = A (j, k+1-A (j, k-1)

If A has gray values in the range 0 to 255, for example, then B may have values in the range -255 to 255. The values of B were renormalized to the range 0 to 255 by shifting and scaling. This can be done by the replacement

$$\left[\frac{B(j,k)-B\min}{B\max-B\min}\times 255\right]\rightarrow B(j,k)$$

Where the brackets [ ] indicate rounding to the nearest integer. Negative edges found by horizontal gradient detection with b=120. were detected by applying a threshold operation to the results of the horizontal gradient calculation. The strongest negative edge transitions were then detected with a threshold operation, in which the pixels of B that were less than a threshold b were detected figure3. Negative edges found by horizontal gradient detection with b=120. Shows those pixels for b=120.

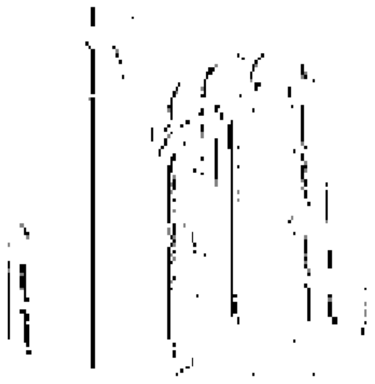


Figure 3:Negative edges found by horizontal gradient detection with b=120.

Negative edges found by horizontal gradient detection with b=120. were detected by applying a threshold operation to the results of the horizontal gradient calculation. The strongest negative edge transitions were then detected with a threshold operation, in which the pixels of B that were less than a threshold b were detected. Figure 3.Negative edges found by horizontal gradient detection with b=120. Shows those pixels for b=120.

Horizontal edges

Horizontal edges produce a vertical gradient in the image, and can be enhanced with a vertical gradient detector. A vertical gradient filter can be defined by

B(j,k)=A(j+1,k)-A(j-1,k)

The gradient values were shifted and normalized by

$$\left[\frac{B(j,k)-B\min}{B\max-B\min}\times 255\right]\rightarrow B(j,k)$$

When the gradient is calculated on the building image, the result is as shown in figure 4.Edge produced by vertical gradient calculation. The origin of the image is at the lower left corner, so that higher numbered rows are higher in the figure. An edge doublet that is formed by the gradient calculation in going from a darker area to a lighter area will be bright below dark. We wee this in the rounded "eyebrows" in the stone work.

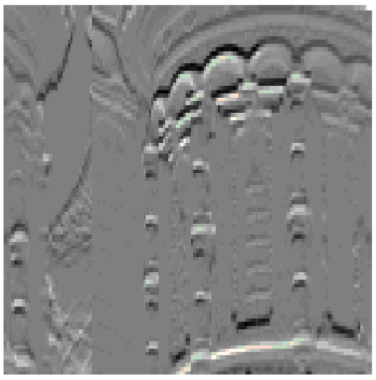


Figure 4:Edge produced by vertical gradient calculation.

The edges shown in figure 5. Positive edges found by vertical gradient detection with a=180. Were detected by applying a threshold operation to the results of the vertical gradient calculation. The strongest positive edge transitions were then detected with a threshold operation, in which the pixels of B that were greater than a threshold a were detected. Figure 5. Positive edges found by vertical gradient detection with a=180. Shows those pixels for a=180.

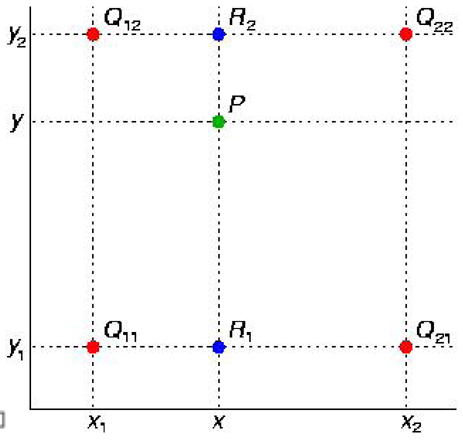
Figure 5 Positive edges found by vertical gradient detection with a=180.

The edges shown in figure 6.Negative edges found by vertical gradient detection with b=80. Were detected by applying a threshold operation to the results of the vertical gradient calculation. The strongest negative edge transitions were then detected with a threshold operation, in which the pixels of B that were less than a threshold *b* were detected. Figure 6 Negative edges found by vertical gradient detection with b=80. Shows those pixels for *b*=80.

Figure 6: Negative edges found by vertical gradient detection with b=80.

B. Bilinear Interpolation

Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value. An algorithm is used to map a screen pixel location to a corresponding point on the texture map. A weighted average of the attributes (color, alpha, etc.) of the four surrounding texts is computed and applied to the screen pixel. This process is repeated for each pixel forming the object being textured. This algorithm reduces some of the visual distortion caused by resizing an image to a non-integral zoom factor, as opposed to nearest neighbor interpolation, which will make some pixels appear larger than others in the resized image. It uses Bilinear Interpolation to extract the values from a grid into a point feature class. It used as a result of the distance weighted average for interpolating such values.



- P = DESIRED POINT
- Q = Known Points (four closest pixels)
- R = Point on the line with the known points

If we are given four (x,y) points such that

f(xl,yl) = zl f(xl,y2) = z2

f(x2,yl) = z3 f(x2,y2) = z4

and

$x_2 > x_l$
$y_2 > y_l$

then we can solve for a function in four unknown coefficients. There are an infinite number of such equations, but the most simple candidate is

$$z = ax + bx + cxy + d$$

so the system to solve for a, b, c and d is

ax1 + by1 + cx1y1 + d = z1

ax1 + by2 + cx1y2 + d = z2

ax2 + by1 + cx2y1 + d = z3

ax2 + by2 + cx2y2 + d = z4

We could solve this directly for a, b, c and d, but we can get a more simple solution by scaling x and y, like this

T=(x-x1)/(x2-x1) and U=(y-y1)/(y2-y1)

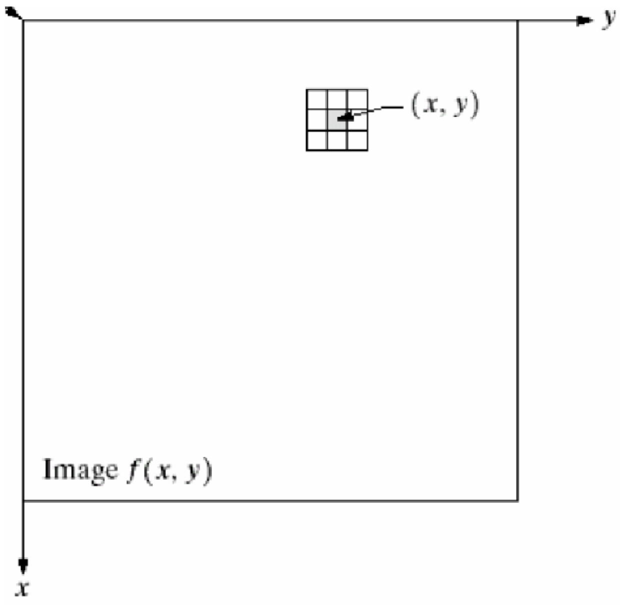
With this scaling, t = 0 when x = xl and t = l when x = x2. Similarly, u = 0 when y = yl and u = l when y = y2. This equation system to solve simplifies to this:

d = zl
b + d = z2   or   a = z3 - zl b = z2 - zl
a + d = z3
a + b + c + d = z4   c = zl - z2 - z3 + z4 d = zl

So the equation to estimate z in terms of t and u is
z = (z3 - zl)t + (z2 - zl)u + (zl - z2 - z3 + z4)ut + zl
We can expand this equation, collect on zl, z2, z3 and z4, and further factor that result to get
z = zl(l-u)(l-t) + z2u(l-t) + z3t(l-u) + z4ut

C. *Sharpening Spatial Filter*

Sharpening spatial filter which is like high pass filter .It is a form of FIR filtering. The filter is actually a mask of weights arranged in a rectangular pattern, The sharpening spatial filtering is mainly used for smoothing and sharpening. It Highlight transitions in intensities. The Edges are mainly used in image analysis for finding region boundaries. The Edges are pixels where values change abruptly. The Sharpening – Image details can be enhanced by adding the edges to the input image For image functions f (x; y) partial derivatives may be approximated by differences A change can be described by a gradient that pointing in the direction of the largest growth of the image function



From the above Image we explain the following points

- Define a center point (x; y).
- Perform an operation that involves only The pixels in a predefined Neighborhood.
- Result of the operation response of the Process at that point.
- Repeat the process for every pixel in the Image.
- Output is a function of a pixel value and its neighbours.
- Possible operations are: sum, weighted sum, average, weighted average, min, max, median,.

In the sharpening spatial filter the neighbourhood and kernel(mask) is used. the neighbourhood operation is to multiply each of the pixels in the neighbourhood by a weight and add them together .The local weights are sometimes called a mask or kernel .

The neighbourhood operation is done by the formula used over here

F(x-1,y-1)	F(x-1,y)	F(x-1,y+1)
F(x,y-1)	F(x,y)	F(x,y+1)
F(x+1,y-1)	F(x+1,y)	F(x,y+1)

W(-1,0)	W(-1,1)	W(-1,1))
W(0,0)	W(0,1)	W(0,1)
W(1,0)	W(1,1)	W(1,1)

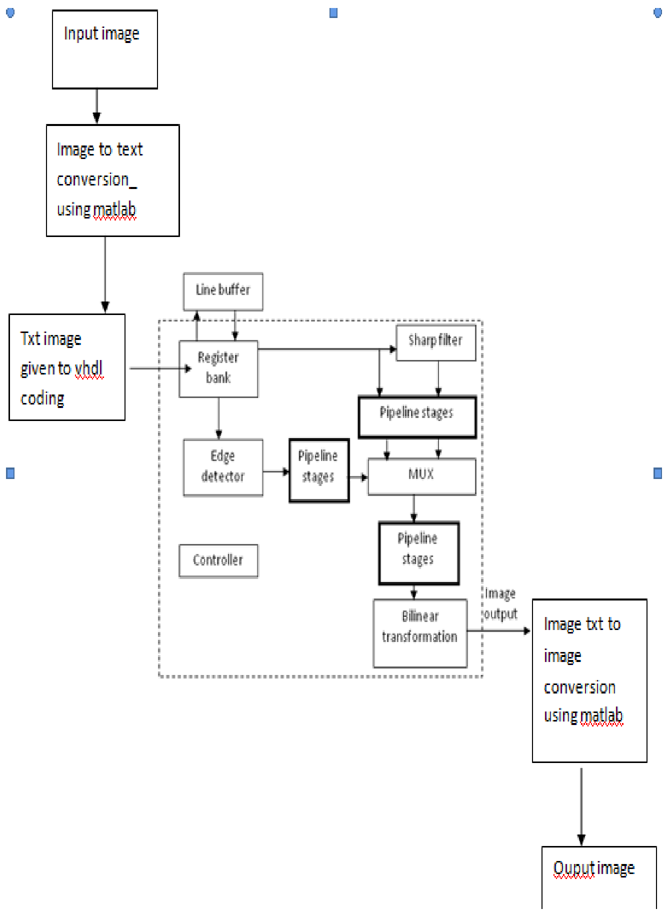
$$g(x,y)= \sum_{j=-k}^{k} \sum_{j=-k}^{k} w(i,j).f(x+i,y+j)$$

For Convolution - the same as correlation except that filter kernel W is rotated for 180 degrees. The convolution is indicated by (*).

The local neighbourhood is used for the Pixels in the neighbourhood are correlated. Rectangular neighbourhoods are often used with an odd number of pixels in rows and columns, enabling specification of the central pixel of the neighbourhood .Different neighbourhood sizes: 3x3, 5x5, 11x3,...,2n+1x2m+1. The choice of size and shape (rectangular, circular,...) of the neighbourhood depends on the size of the objects in the image. In spatial filter we use smoothing filter to give smoothness for the image.

In register bank we just store the gray scale image in a 8 bit register and we use further processs using linebuffer which is a data structure that holds a fixed amount of data in a serial fashion, The oldest data data gets discarded as new data is added.

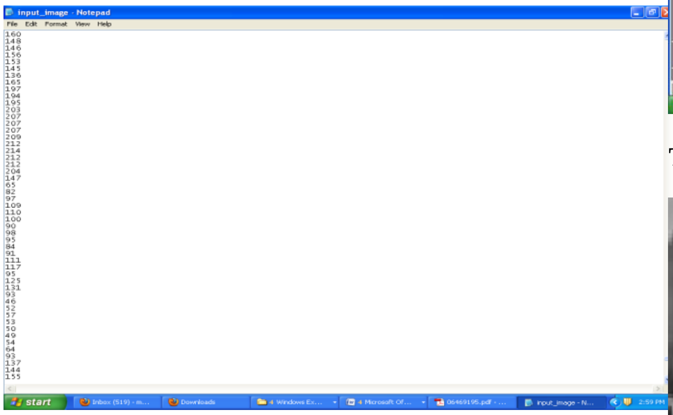
E.Block Diagram



The input gray scale image is given to the matlab from image to text. The text image is given to register bank to store the image. The coding is given to the sharp filter and edge detector. The edge detector identifies the horizontal and vertical gradient. The gradient gets added to form the edges in a image. The sharp filter actually a mask of weights arranged in a rectangular pattern, It is mainly used for smoothing and sharpening. The image is given to pipeline stages to speed up the process. The output is given to the multiplexer. The multiplexer combines the both two output. The multiplexed output is given to the bilinear interpolation. The bilinear interpolation is used to upscaling or image zooming. Finally the image is zoomed by bilinear interpolation. The image text is converter to image using matlab. Finally output image is obtained.

The Input Gray Scale Image:

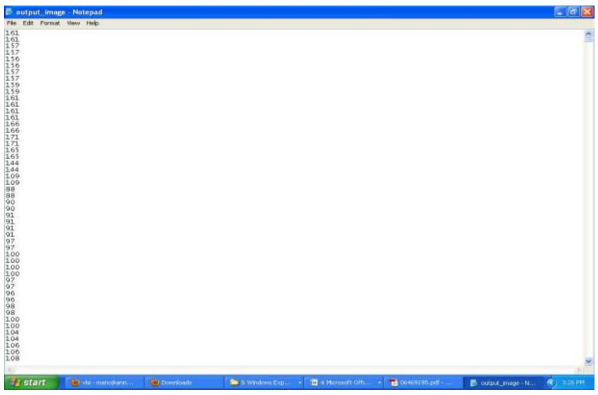
The Image is Converted to Text Using Matlab:



input_image - Notepad

```
160
148
146
156
153
145
136
165
197
194
195
203
207
207
207
209
212
214
212
212
204
147
65
82
97
109
110
100
90
98
95
84
91
111
117
95
125
131
93
46
52
57
53
50
49
54
64
93
137
144
155
```

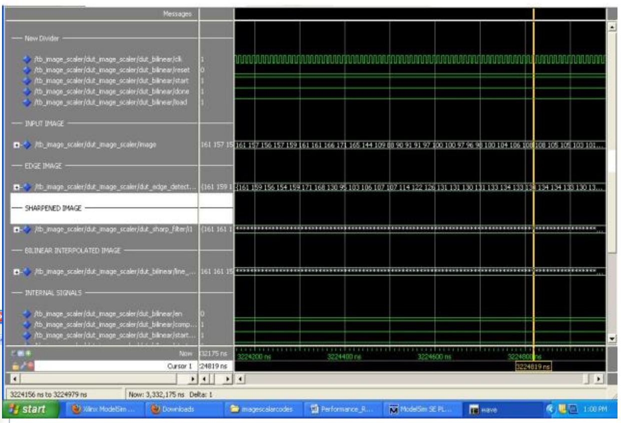
The output text is :



output_image - Notepad

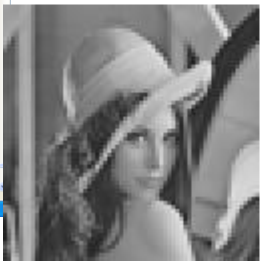
```
161
161
157
157
156
156
157
157
159
159
161
161
161
161
166
166
171
171
165
165
144
144
109
109
88
88
90
90
91
91
91
91
97
97
100
100
100
100
97
97
96
96
98
98
100
100
104
104
106
106
108
```

The Text Processed by VHDL:



```
New Divider
/tb_image_scaler/dut_image_scaler/dut_bilinear/clk        1
/tb_image_scaler/dut_image_scaler/dut_bilinear/reset      0
/tb_image_scaler/dut_image_scaler/dut_bilinear/start      1
/tb_image_scaler/dut_image_scaler/dut_bilinear/done       1
/tb_image_scaler/dut_image_scaler/dut_bilinear/load       1
INPUT IMAGE
/tb_image_scaler/dut_image_scaler/image                   161 157 15  161 157 156 157 159 161 161 166 171 165 144 109 88 90 91 91 97 100 100 97 96 98 100 104 106 108 108 105 105 103 101...
EDGE IMAGE
/tb_image_scaler/dut_image_scaler/dut_edge_detect...      {161 159 1  {161 159 156 154 159 171 168 130 95 103 106 107 107 114 122 126 131 131 130 131 133 134 133 134 134 134 133 130 13...
SHARPENED IMAGE
/tb_image_scaler/dut_image_scaler/dut_sharp_filter/i1     {161 161 1
BILINEAR INTERPOLATED IMAGE
/tb_image_scaler/dut_image_scaler/dut_bilinear/line_...   161 161 15
INTERNAL SIGNALS
/tb_image_scaler/dut_image_scaler/dut_bilinear/en         0
/tb_image_scaler/dut_image_scaler/dut_bilinear/comp...    1
/tb_image_scaler/dut_image_scaler/dut_bilinear/start...   1
Now       32175 ns
Cursor 1  24819 ns
3224156 ns to 3224979 ns      Now: 3,332,175 ns  Delta: 1
```

The output text is converted to image using matlab:



RESULT:

Image Scalar:

Module Name	Area	Delay
Bilinear transformation	6290 Logic gates	9.008ns
Edge detector	170 Logic gates	3.225ns
Sharpening filter	256 Logic gates	1.23ns
Image scalar	6716 Logic gates	9.008ns

Bilin
ear
Resu
lt:

Bilinear Result			
Logic Utilization	Used	Available	Utilization
Number of Slices	243	3584	6%
Number of Slice Flip Flops	327	7168	4%
Number of 4 input LUTs	476	7168	6%
Number of bonded IOBs	5	141	3%
Number of GCLKs	1	8	12%

Timing Summary:

Frequency	355 MHz
Minimum input arrival time	2.145 ns
Maximum input arrival time	2.053ns

V.CONCLUSION

In this work adaptive edge enhanced image scalar is developed, which offers low complexity, low power and low memory requirement and high speed. In this work extra pipelines are introduced at the end of gradient based edge detector, sharpening filer (2D-Convolution filter). These pipelining offers reduced critical path, and hence improve the speed of circuit operation. In this work the area and delay are calculated. Edge detector occupies 170 logic gates and offers a delay of 3.225ns, Sharpening filter occupies 256 logic gates and offers a delay of 1.23ns, and bilinear transform offers a delay of 9.008ns with 6290logic gates. The total area and delay are 6716 logic gates and delay of 9.008ns. The design was simulated using the industry leading simulator ModelSim 6.2c from Mentor Graphics, Inc. This design was synthesized to Xilinx Spartan series

REFERENCES

1.ENHANCED VLSI IMPLEMENTATION OF AN ADAPTIVE EDGE- IMAGE SCALAR FOR REAL-TIME MULTIMEDIA APPLICATIONS BY SHIH- LUN CHEN, MEMBER, IEEE BY SEP 2013

2.C. GONNER, AND K. SPITZER, "SURVEY: INTERPOLATION METHODS IN MEDICAL IMAGE T. M. LEHMANN PROCESSING," IEEE TRANS. MED. IMAG., VOL.18, NO. 11, PP. 1049–1075, NOV. 1999

3.C. T. LIN, K. W. FAN, H. C. PU, S. M. LU, AND S. F. LIANG, "AN HVSDIRECTENEURAL-NETWORK-BASED IMAGE RESOLUTION ENHANCEMENT SCHEME FOR IMAGE RESIZING," IEEE TRANS. FUZZY SYST., VOL. 15, NO. 4, PP. 605–615,AUG. 2007.

4.CASELLES, J. M. MOREL, AND C. SBERT, "AN AXIOMATIC APPROACH TO IMAGE INTERPOLATION," IEEE TRANS. IMAGE PROCESS., VOL. 7, NO. 3, PP. 376–386,MAR. 1998.

5.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," IN PROC. IEEE INT. CONF. FIELD-PROGRAMMABLE TECHNOL., 2008, PP. 381–384.

6.C. C. LIN, M. H. SHEU, H. K. CHIANG, Z. C. WU, J. Y. TU, AND C.H. CHEN, "A LOW-COST VLSI DESIGN OF EXTENDED LINEAR INTERPOLATION FOR REAL TIME DIGITAL IMAGE PROCESSING," IN PROC. IEEE INT CONF. EMBEDDEDSOFTWARE SYST., JUL. 2008, PP. 196–202.

7.D. DARIAN AND T. W. PARKS, "ADAPTIVELY QUADRATIC (AQUA) IMAGE INTERPOLATION,"IEEE TRANS. IMAGE PROCESS., VOL. 13, NO. 5, PP. 690–698,2004

8.vlsi implementation of an adaptive edge-enhanced image scalar for real-time multimedia applications by shihlun chen, member, ieee by sep 2013

9.C. GONNER, AND K. SPITZER, "SURVEY: INTERPOLATION METHODS IN MEDICAL IMAGE T. M. LEHMANN PROCESSING," IEEE TRANS. MED. IMAG., VOL.18, NO. 11, PP. 1049–1075, NOV. 1999

9.C. H. KIM, S. M. SEONG, J. A. LEE, AND L. S. KIM, "WINSCALE : AN IMAGESCALING ALGORITHM USING AN AREA PIXEL MODEL", IEEE TRANS. CIRCUITSSYSTEM VIDEO TECHNOL., VOL. 13, NO. 6, PP. 549–553, JUN. 2003.

10.U, W. K. TSAI, M. H. SHEU, AND H. K. CHIANG, "THEVLSI DESIGN OF WINSCALE FOR DIGITAL IMAGE SCALING," IN PROC. IEEE INT.CONF. INTELL. INFORM. HIDING MULTIMEDIA SIGNAL PROCESS., NOV. 2007, PP.511–514

11.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," IN PROC. IEEE INT. CONF. FIELD-PROGRAMMABLE TECHNOL., 20

12.D. DARIAN AND T. W. PARKS, "ADAPTIVELY QUADRATIC (AQUA) IMAGE INTERPOLATION,"IEEE TRANS. IMAGE PROCESS., VOL. 13, NO. 5, PP. 690–698,2004

13.C. C. LIN, Z. C. WU, W. K. TSAI, M. H. SHEU, AND H. K. CHIANG, "THEVLSI DESIGN OF WINSCALE FOR DIGITAL IMAGE SCALING," IN PROC. IEEE INT.CONF. INTELL. INFORM. HIDING MULTIMEDIA SIGNAL PROCESS., NOV. 2007, PP.511–514.