Abstract— Image scaling is the process of resizing a digital image. It is one of the most important methods used in various applications such as sharpening of an image, image zooming, preserving edge structures in an image and the image processing techniques described here can be implemented by both the hardware and software contained within image processing system. One method is to determine various functions which possess the similar shape in both the space domain and frequency domain. This strategy yields the result of various image interpolation algorithms. The higher resolution images yielded by this algorithm consisted of blocking artifacts. For the purpose of alleviating the blocking effects, the algorithm known as bilinear interpolation is introduced. The performance of an edge enhancement process is shown in above Figure (2). Here an efficient Image Scaling algorithm is proposed for the VLSI Image scaling processor using VERILOG code to the existing method. Simulation results shows, the code developed algorithm for proposed image scaling algorithm can reduce the memory requirement, and hardware cost and needs only one-line buffer memory compared to conventional schemes.

Keywords—Bilinear interpolation, Combined filter, controller, demosaicing, Prefilters, Reconfigurable calculation unit (RCU), Very Large Scale Integration (VLSI).

I. INTRODUCTION

Image Scaling is the process of resizing a digital image. Scaling is a process, which is non-trivial that involves a trade-off between efficiency, smoothness and sharpness. Image scaling has been widely applied in the fields of digital imaging devices such as digital cameras, digital video recorders, digital photo frame, high-definition television, mobile phone, tablet PC, etc. An obvious application of image scaling is to scale down the high-quality pictures or video frames to fit the mini size liquid crystal display panel of the mobile phone or tablet PC. As the graphic and video applications of mobile handset devices grow up, the demand and significance of image scaling are more and more outstanding.

The image scaling algorithms can be separated into polynomial-based and non-polynomial-based methods. The simplest polynomial-based method is a nearest neighbor algorithm. It has the benefit of low complexity, but the scaled images are full of blocking and aliasing artifacts. The most widely used scaling method is the bilinear interpolation algorithm [1], by which the target pixel can be obtained by using the linear interpolation model in both of the horizontal and vertical directions. Another popular polynomial-based method is the bicubic interpolation algorithm [9], which uses an extended cubic model to acquire the target pixel by a 2-D regular grid.

In recent years, many high-quality non-polynomial-based methods [2]–[4] have been proposed. Bilateral filter [2], interpolation [1], and autoregressive model [3] are the techniques which have proposed earlier to enhance the image quality by reducing the blurring and aliasing artifacts. The methods mentioned earlier efficiently enhance the image quality as well as reduce the artifacts of the blocking, aliasing, and blurring effects. However, these high-quality image scaling algorithms have the characteristics of high complexity and high memory requirement, which is not easy to be realized by VLSI technique. Thus, for real-time applications, low-complexity image processing algorithms are necessary for VLSI implementation [5]–[7].

II. PROPOSED SCALING ALGORITHM

Figure 1 shows the block diagram of the proposed scaling algorithm. It consists of a sharpening spatial filter, a clamp filter, and a bilinear interpolation. The sharpening spatial and clamp filters serve as pre filters to reduce blurring and aliasing artifacts produced by the bilinear interpolation. First, the input pixels of the original images are filtered by the sharpening spatial filter to enhance the edges and remove associated
noise. Second, the filtered pixels are filtered again by the clamp filter to smooth unwanted discontinuous edges of the boundary regions. Finally, the pixels filtered by both of the sharpening spatial and clamp filters are passed to the bilinear interpolation for up-/downscaling. To conserve computing resource and memory buffer, these two filters are simplified and combined into a combined filter. The details of each part will be described in the following sections.

**A. Low-Complexity Sharpening Spatial and Clamp Filters**

The sharpening spatial filter, a kind of high-pass filter, is used to reduce blurring artifacts and defined by a kernel to increase the intensity of a center pixel relative to its neighboring pixels. The principal objective of sharpening is to highlight fine details in an image or to enhance details that have been blurred either in error or as a natural effect of particular method for image acquisition. The applications of image sharpening range from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems. Sharpening Spatial Filter usually contains a single positive value at the center and is completely surrounded by negative values. The below kernel shows an example of a $3 \times 3$ array of sharpening spatial filter,

$$
\text{Kernel} = \begin{bmatrix}
-1 & -1 & -1 \\
-1 & S & -1 \\
-1 & -1 & -1 \\
\end{bmatrix}
$$

The clamp filter, a kind of low-pass filter, is a 2-D Gaussian spatial domain filter and composed of a convolution kernel array. It usually contains a single positive value at the center and is completely surrounded by negative values. The clamp filter is used to reduce aliasing artifacts and smooth the unwanted discontinuous edges of the boundary regions. The clamp filter can be represented by convolution kernels. The below kernel shows an example of a $3 \times 3$ array of an Clamp filter,

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

**B. Convolution Kernel**

A larger size of convolution kernel will produce higher quality of images. However, a larger size of convolution filter will also demand more memory and hardware cost. For example, a $6 \times 6$ convolution filter demands at least a five-line-buffer memory and 36 arithmetic units, which is much more than the two-line-buffer memory and nine arithmetic units of a $3 \times 3$ convolution filter. In our previous work, each of the sharpening spatial and clamp filters was realized by a 2-D $3 \times 3$ convolution kernel as shown in Fig. 2(a). It demands at least a four-line-buffer memory for two $3 \times 3$ convolution filters. For example, if the image width is 1920 pixels, $4 \times 1920 \times 8$ bits of data should be buffered in memory as input for processing.

![Figure 2](image)

Figure 2: Weights of the convolution kernels. (a) $3 \times 3$ convolution kernel, (b) Cross-model convolution kernel, (c) T-model and inversed T-model convolution kernels.

To reduce the complexity of the $3 \times 3$ convolution kernel, a Cross-model formed is used to replace the $3 \times 3$ convolution kernel, as shown in Fig. 2(b). It successfully cuts down on four of nine parameters in the $3 \times 3$ convolution kernel. Furthermore, to decrease more complexity and memory requirement of the cross-model convolution kernels, T-model and inversed T-model convolution kernels are proposed for realizing the sharpening spatial and clamp filters. As shown in Fig.2(c), the T-model convolution kernel is composed of the lower four parameters of the cross-model and the inversed T-model convolution kernel is composed of the upper four parameters.

In the proposed scaling algorithm, both the T-model and inversed T-model filters are used to improve the quality of the images simultaneously. The T-model or inversed T-model filter is simplified from the $3 \times 3$ convolution filter of the previous work, which not only efficiently reduces the complexity of the convolution filter but also greatly decreases the memory requirement from two to one line buffer for each convolution filter. The T-model and the inversed T-model provide the low-complexity and low memory-requirement convolution kernels for the sharpening spatial and clamp...
filters to integrate the VLSI chip of the proposed low-cost image scaling processor.

C. Combined Filter
In proposed scaling algorithm, the input image is filtered by a sharpening spatial filter and then filtered by a clamp spatial filter again. Although the sharpening spatial and clamp filters are simplified by T-models and inversed T-models, it still needs two line buffers to store input data or intermediate values for each T-model or inversed T-model filter. Thus, to be able to reduce more computing resource and memory requirement, sharpening spatial and clamp filters, which are formed by the T-model or inversed T-model, should be combined together into a combined filter as,

\[
\mathbf{p}'_{mn} = \left[ \begin{array}{cccc}
S-C & S-C & S-C & S-C
\end{array} \right] \mathbf{p}_{mn}^{-1} \quad \rightarrow (1)
\]

Where S and C are the sharp and clamp parameters and \( p'_{mn} \) is the filtered result of the target pixel \( p_{mn} \) by the combined filter. A T-model sharpening spatial filter and a T-model clamp filter have been replaced by a combined T-model filter as shown in (1). To reduce the one-line-buffer memory, the only parameter in the third line, parameter \(-1\) of \( p_{mn-2} \), is removed, and the weight of parameter \(-1\) is added into the parameter \( S-C \) of \( p_{mn-1} \) by \( S-C-1 \) as shown in (2). The combined inversed T-model filter can be produced in the same way. In the new architecture of the combined filter, the two T-model or inversed T-model filters are combined into one combined T-model or inversed T-model filter. By this filter-combination technique, the demand of memory can be efficiently decreased from two to one line buffer, which greatly reduces memory access requirements for software systems or hardware memory costs for VLSI implementation.

D. Simplified Bilinear Interpolation
In the proposed scaling algorithm, the bilinear interpolation method is selected because of its characteristics with low complexity and high quality. The bilinear interpolation is an operation that performs a linear interpolation first in one direction and, then again, in the other direction. The output pixel \( p_{k+1} \) can be calculated by the operations of the linear interpolation in both x- and y-directions with the four nearest neighbor pixels. The target pixel \( p_{k+1} \) can be calculated by

\[
p_{k+1} = \frac{(1 - dx) \times p_{m,n} + dx \times (1 - dy) \times p_{m,n+1} + (1 - dy) \times p_{m,n+1} + dy \times p_{m,n+1}}{(1 - dx) \times dy \times p_{m,n+1} + dx \times dy \times p_{m,n+1+1}} \quad \rightarrow (4)
\]

Where \( p_{m,n}, p_{m+1,n}, p_{m,n+1} \) and \( p_{m+1,n+1} \) are the four nearest neighbor pixels of the original image and \( dx, dy \) are scale parameters in the horizontal and vertical directions. By (2), we can easily find that the computing resources of the bilinear interpolation cost eight multiply, four subtract, and three addition operations. It costs a considerable chip area to implement a bilinear interpolator with eight multipliers and seven adders. Thus, an algebraic manipulation skill has been used to reduce the computing resources of the bilinear interpolation. The original equation of bilinear interpolation is presented in (2), and the simplifying procedures of bilinear interpolation can be described from (4)–(6). Since the function of \( dy \times (p_{mn} + p_{mn+1}) \) appears twice in (6), one of the two calculations for this algebraic function can be reduced

\[
p_{k+i,j} = \left[ (1 - dy) \times p_{m+i,n+j} + dy \times p_{m+i,n+j+1} \right] \times dx + \left[ (1 - dy) \times p_{m+i,n+j} + dy \times p_{m+i,n+j+1} \right] \times (1 - dx) \quad \rightarrow (5)
\]

Thus, the function \( dy \times (p_{mn} + p_{mn+1}) \) can be replaced by the previous result of \( p_{mn} + p_{mn+1} \) as shown in (6). The simplifying procedures successfully reduce the computing resource from eight multiply, four subtract, and three add operations to two multiply, two subtract, and two add operations.

III. PROPOSED SCALING ALGORITHM
The proposed scaling algorithm consists of two combined pre filters and one simplified bilinear interpolator. For VLSI implementation, the bilinear interpolator can directly obtain two input pixels \( p'_{mn} \) and \( p'_{m,n+1} \) from two combined pre filters without any additional line-buffer memory. Figure 3 shows the block diagram of the VLSI architecture for the proposed design. It consists of four main blocks: a register bank, a combined filter, a bilinear interpolator, and a controller. The details of each part will be described in the following sections.
A. Architecture of the Register Bank
In this brief, the combined filter is filtering to produce the target pixels of $p'_{(m,n)}$ and $p'_{(m,n+1)}$ by using ten source pixels. The register bank is designed with a one-line memory buffer, which is used to provide the ten values for the immediate usage of the combined filter. Figure.4 shows the architecture of the register bank with a structure of ten shift registers.

When the shifting control signal is produced from the controller, a new value of $p_{(m+3,n)}$ will be read into Reg1, and each value stored in other registers belonging to row $n+1$ will be shifted right into the next register or line-buffer memory. The Reg10 reads a new value of $P_{(m+2,n)}$ from the line-buffer memory, and each value in other registers belonging to row $n$ will be shifted right into the next register.

B. Combined Filter
The combined T-model or inversed T-model convolution function of the sharpening spatial and clamp filters is discussed in Section II, and the equation is represented in (1). Figure.7 shows the six-stage pipelined architecture of the combined filter and bilinear interpolator, which shortens the delay path to improve the performance by pipeline technology.

The stages 1 and 2 in Figure.7 show the computational scheduling of a T-model combined and an inverse T-model filter. The T-model or inversed T-model filter consists of three reconfigurable Calculation units (RCUs), one multiplier–adder (MA), three adders (+), three subtractors (−), and three shifters (S). The hardware architecture of the T-model combined filter can be directly mapped with the convolution equation shown in (1). The values of the ten source pixels can be obtained from the register bank mentioned earlier.

The symmetrical circuit, as shown in stages 1 and 2 of Figure.5, is the inversed T-model combined filter designed for producing the filtered result of $p'_{(m,n+1)}$. Obviously, the T-model and the inversed T-model are used to obtain the values of $p'_{(m,n)}$ and $p'_{(m,n+1)}$ simultaneously. The architecture of this symmetrical circuit is a similar symmetrical structure of the T-model combined filter, as shown in stages 1 and 2 of Figure.5. Both of the combined filter and symmetrical circuit consist of one MA and three RCUs. The MA can be implemented by a multiplier and an adder. The RCU is designed for producing the calculation functions of $(S-C)$ and $(S-C-1)$ times of the source pixel value, which must be implemented with C and S parameters.

The C and S parameters can be set by users according to the characteristics of the images. The architecture of the proposed low-cost combined filter can filter the whole image with only a one-line-buffer memory, which successfully decreases the memory requirement from four to one line buffer of the combined filter in our previous work [15].
Figure 6: Architecture of the Reconfigurable calculation Unit

Table I lists the parameters and computing resource for the RCU. With the selected C and S values listed in Table I. The gain of the clamp or sharp convolution function is {8, 16, 32} or {4, 8, 16}, which can be eliminated by a shifter rather than a divider. Figure 6 shows the architecture of the RCU (Reconfigurable Calculation Unit). It consists of four shifters, three multiplexers (MUX), three adders, and one sign circuit. By this RCU design, the hardware cost of the combined filters can be efficiently reduced.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>S-C</td>
</tr>
<tr>
<td>S-C-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,13,29</td>
</tr>
<tr>
<td>7,11,19</td>
</tr>
<tr>
<td>2,-6,-22,6,-2,-18,14,6,-10</td>
</tr>
<tr>
<td>1,-7,-23,5,-3,-19,13,5,-11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computing Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add and Shift</td>
</tr>
<tr>
<td>Add and Shift</td>
</tr>
<tr>
<td>Add, Shift and Sign</td>
</tr>
<tr>
<td>Add, Shift and Sign</td>
</tr>
</tbody>
</table>

C. Bilinear Interpolator and Controller

Bilinear interpolation is an extension of linear interpolation for interpolating functions of two variables on a regular 2D grid. The principle of Bilinear interpolation is to perform linear interpolation first in one direction, and then again in the other direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location. The architecture of bilinear interpolation is shown in Figure 9.

IV. SIMULATION RESULT

A low-cost and high-quality image scalar is proposed for previous work. It successfully improves the image quality by adding sharpening spatial and clamp filters as pre-filters with an adaptive technique based on the algorithm of bilinear interpolation. As per the six stage pipeline architecture, the image is initially passed through the combined filter, where the aliasing and blurring effects are removed. After that bilinear interpolation is performed on the filtered output to obtain a scaled image. The input image used for the image scaling process is Text image. The image matrix is represented by 95 x 89 pixels, which totally contains 25365 bytes pixel values.

Figure 7: Architecture of Bilinear Interpolator & Controller.

The stages 3, 4, 5 and 6 in Figure 7 show the four-stage pipelined architecture, and two-stage pipelined multipliers are used to shorten the delay path of the bilinear interpolator. The input values of \( p'(m,n) \) and \( p'(m,n+1) \) are obtained from the combined filter and symmetrical circuit. By the hardware sharing technique, as shown in (6), the temperature result of the function \( p'(m,n) + dy \times (p'(m,n+1) - p'(m,n)) \) can be replaced by the previous result of \( p'(m+1,n) + dy \times (p'(m+1,n+1) - p'(m+1,n)) \). It also means that one multiplier and two adders can be successfully reduced by adding only one register. The controller is implemented by a finite-state machine circuit. It produces control signals to control the timing and pipeline stages of the register bank, combined filter, and bilinear interpolator.

Figure 8: Simulation waveform for the combined filter.
Figure 8 shows the combined filter output waveform. T model and inversed T model matrix is produced as the output of combined filter. The filtered pixels $p'_{(m,n)}$ and $p'_{(m,n+1)}$ are given as input to the bilinear interpolator. The Figure 9 shows the bilinear interpolation output waveform with new scaled pixels. The scaled output image is displayed using the Mat lab software.

**Figure 9:** Out put image for bilinear interpolator in Text

The output image of size 190x178 is obtained. The total estimated power consumption of this image scaling algorithm is 303mW at a 115 MHz operating frequency and uses about 3.74-k gate counts with peak memory usage of about196MB. The design of a register bank is utilized to provide 10 values for the consumption of the T-model, which requires a single line buffer memory and it is coded and simulated by the synthesis of Xilinx ISE Simulator is depicted in Figure 8.

**V. CONCLUSION**

An image scaling technique was developed with low memory requirement, high quality and high performance VLSI scalar processor for real-time image zooming applications. The Sharpening Spatial filter and Clamp filter were used to reduce the smoothness and artifacts produced by the Bilinear Interpolator. Further, T-model and Inversed T-model were used to minimize the memory buffers and computing resources for the proposed image processor algorithm. Two T-model or inversed T-model filters were combined into a combined filter which requires only a one-line-buffer memory. Moreover, the algebraic manipulation and the hardware sharing techniques performed by Reconfigurable calculation unit (RCU) would reduce the calculation complexity and hardware cost of design.

**REFERENCES**


