JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 26, 631-648 (2010)

An Efficient Architecture of Extended Linear Interpolation for Image Processing

CHUNG-CHI LIN[1], MING-HWA SHEU[2], HUANN-KENG CHIANG[2], CHISHYAN LIAW[1], ZENG-CHUAN WU[3] AND WEN-KAI TSAI[2]
[1]*Department of Computer Science*
Tung Hai University
Taichung, 407 Taiwan
[2]*Graduate School of Engineering Science and Technology*
[3]*Department of Electronic Engineering*
National Yunlin University of Science and Technology
Yunlin, 640 Taiwan

This paper presents a novel image interpolation method, extended linear interpolation, which is a low-cost and high-speed architecture with interpolation quality compatible to that of bi-cubic convolution interpolation. The method of reducing computational complexity of generating weighting coefficients is proposed. Based on the approach, the efficient hardware architecture is designed under real-time requirement. Compared to the latest bi-cubic hardware design work, the architecture saves about 60% of hardware cost. The architecture is implemented on the *Virtex-II FPGA*, and the high-speed VLSI has been successfully designed and implemented with TSMC 0.13μm standard cell library. The simulation results demonstrate that the efficient VLSI of extended linear interpolation at 267MHz with 25980 gates in a $450 \times 450 \mu m^2$ chip is able to process digital image scaling for HDTV in real-time.

Keywords: interpolation, scaling, image reconstruction, bi-cubic convolution, VLSI

1. INTRODUCTION

Digital image scaling has a variety of applications; for instance, scaling up is used to enlarge images for HDTV or medical image display and scaling down is applied to shrink images to fit mini-size LCD panel in portable instruments. Numerous digital image scaling techniques have been presented [1-15]. The most popular methods are nearest neighbor, bi-linear, cubic, and bi-cubic, in which the simplest approaches are nearest neighbor and bi-linear interpolation. These simple methods are easy to be implemented in hardware; however, they have undesirable blocks effect and blurring effect. Winscale [8, 9] interpolation is a linear scaling algorithm that was implemented on an FPGA. Its hardware cost is low, but the scaling ratio is limited to the range of [0.5, 2] only and the resulting images involve undesirable blocks effect and blurring effect.

A better quality of interpolation is achieved by using higher order models; the typical method of this category is bi-cubic convolution interpolation. Nevertheless, it requires a lot of computation efforts and as a result, the software implementation is hard to achieve real-time. To reduce computational complexity, Nuno-Maganda [7] decomposed bi-cubic method as two 1-dimension interpolations on a *Xilinx Virtex II Pro FPGA* for

Received March 17, 2008; revised August 20, 2008 & February 5, 2009; accepted April 9, 2009.
Communicated by Tong-Yee Lee.

real-time process. It has good image quality but the hardware cost is high and number of memory access times is heavy.

Bi-cubic convolution interpolation does not cause scaled images distortion; however, it needs a lot of computation efforts. This paper presents a novel image interpolation method, extended linear interpolation. The scheme has the advantages of low operation complexity and hardware cost with the interpolation quality compatible to that of bi-cubic convolution interpolation. Based on the approach, the efficient hardware architecture is designed under the real-time requirement.

2. THE RELATED WORKS

In this section, first- and third-order polynomial interpolation kernels, which provide for linear and cubic convolution interpolations, are introduced.

The kernel of linear interpolation is built up of first-order polynomials and approximates the ideal sinc-function in interval [– 1, 1]. The weighting coefficients are given by

$$h_L(s) = \begin{cases} 1-|s|, & 0 \le |s| < 1 \\ 0, & 1 \le |s| \end{cases}. \tag{1}$$

The kernel of cubic convolution interpolations is built up of third-order polynomials [5] and approximates the ideal sinc-function in interval [– 2, 2]. The weighting coefficients are given by

$$h_C(s) = \begin{cases} 1-(c+3)|s|^2+(c+2)|s|^3, & 0 \le |s| < 1 \\ -4c+8c|s|-5c|s|^2+c|s|^3, & 1 \le |s| < 2 \\ 0, & 2 \le |s| \end{cases}. \tag{2}$$

Keys [5] determined the constant c by forcing the Taylor series expansion of the sampled sinc-function with the resulting in $c = -0.5$. The weighting coefficients of Keys [5] interpolation can be obtained by

$$h_{Keys}(s) = \begin{cases} 1-2.5|s|^2+1.5|s|^3, & 0 \le |s| < 1 \\ 2-4|s|+2.5|s|^2-0.5|s|^3, & 1 \le |s| < 2 \\ 0, & 2 \le |s| \end{cases}. \tag{3}$$

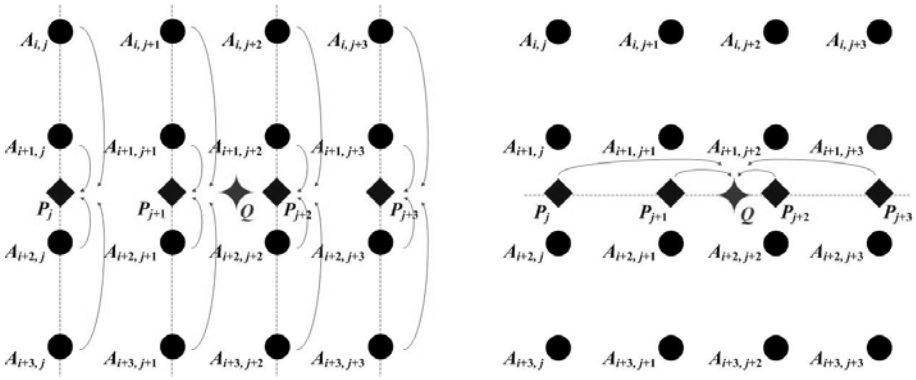
Bi-cubic interpolation uses the constant $c = -1$ and matches the slope of the ideal sinc-function at $s = 1$ [3]. The weighting coefficients of bi-cubic interpolation is given by

$$h_{Bicubic}(s) = \begin{cases} 1-2|s|^2+|s|^3, & 0 \le |s| < 1 \\ 4-8|s|+5|s|^2-|s|^3, & 1 \le |s| < 2 \\ 0, & 2 \le |s| \end{cases}. \tag{4}$$

Eq. (4) indicates that bi-cubic convolution interpolation has less computational complex-

ity than [5]. Thus, bi-cubic interpolation method is widely used as a compromise approximation to the ideal sinc-function interpolation.

Bi-cubic interpolation, a 2-dimensional interpolation method, is usually decomposed as two 1-dimension operations (vertical interpolation and horizontal interpolation), to reduce the computational complexity. Fig. 1 shows the interpolated point, Q, and its 4×4 neighbor source pixels. Interpolation is performed in vertical direction first in our method. Four virtual pixels (P_j, P_{j+1}, P_{j+2}, and P_{j+3}) can be obtained from their corresponding source pixels and the distances between virtual and source pixels. These virtual pixels are reused for horizontal interpolation that is to obtain the interpolated pixel Q.



(a) (b)

Fig. 1. Decomposition of 2-dimension 4×4 interpolations as (a) vertical interpolation and (b) horizontal interpolation.

3. THE PROPOSED METHOD

Basically, the operation of bi-cubic convolution interpolation needs the coordinates of 16 neighboring source pixels of each interpolated point. According to Eq. (4), the operation of 2-dimensional bi-cubic convolution interpolation requires the calculation of 16 weighting coefficients generated along from 16 neighboring pixels of a source image. For each interpolating point, the generation of 16 weighting coefficients is tedious since it may require 80 multiplications and 32 additions. A high-performance scaling method, extended linear interpolation, with reducing computation complexity is presented.

3.1 The Kernel of Extended Linear Interpolation

The extended linear interpolation has the advantage of low complexity as the linear interpolation algorithm and moreover, it improves the quality of linear interpolation. The number of sampling points of third-order polynomial interpolation [11] is applied in extended linear interpolation. In addition, the sampling points of extended linear interpolation uses four points as illustrated in Fig. 2. The proposed method approximates the ideal sinc-function in the interval [– 2, 2]. The weighting coefficients are given by

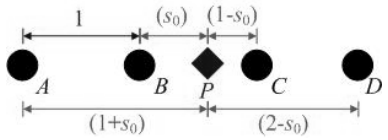


Fig. 2. 1-dimension of third-order polynomial interpolation.

$$h_{EL}(s) = \begin{cases} c_{00} + c_{10}|s|, & 0 \le |s| < 1 \\ c_{01} + c_{11}|s|, & 1 \le |s| < 2 \\ 0, & 2 \le |s| \end{cases}. \tag{5}$$

In Eq. (4), bi-cubic interpolation has some features [11] as following to derive the kernel of extended linear interpolation; since $h_{Bicubic}(s)$ is an even function, only value for $s \ge 0$ are discussed.

(i) The values of $h_{Bicubic}(0) = 1$ and $h_{Bicubic}(1) = 0$.
(ii) Separated bi-cubic interpolation is 1-dimension of third-order polynomial interpolation; the interpolated pixel (P) can be obtained from their corresponding source pixels as shown in Fig. 2. The feature of weighting coefficients of separated bi-cubic interpolation in both directions can be obtained as:

$$h_{Bicubic}(s_0) + h_{Bicubic}(1 + s_0) = 1 - s_0, \tag{6}$$
$$h_{Bicubic}(1 - s_0) + h_{Bicubic}(2 - s_0) = s_0. \tag{7}$$

According to (i), substituting $s = 0$ into Eq. (5), then $c_{00} = 1$; thus, replacing $s = 1$ into Eq. (5), then $c_{11} = -c_{01}$. Assume $c_{01} = \beta$, then $c_{11} = -\beta$. Based on (ii), while $s = s_0$, then $h_{EL}(s_0) + h_{EL}(1 + s_0) = (1 + c_{10} \times s_0) + [\beta + (-\beta) \times (1 + s_0)] = 1 - s_0$; that is, $c_{10} = \beta - 1$. Therefore, Eq. (5) can be expressed as

$$h_{EL}(s) = \begin{cases} 1 + (\beta - 1) \times |s|, & 0 \le |s| < 1 \\ \beta - \beta \times |s|, & 1 \le |s| < 2 \\ 0, & 2 \le |s| \end{cases}, \tag{8}$$

where β is sharpness factor.

3.2 Sharpness Factor Decision

In order to obtain the sharpness factor which has the best interpolation quality, the sum of standard deviation is applied to determine the value of β, whose spectrum is the most approximate the ideal sinc-function. Fig. 3 depicts the standard deviation curve of the spectrum for β in the range between 0 and 0.5 and that of the sinc-function. It shows that the minimum standard deviation is at $\beta = 0.1216$. After the value of β that has the best quality is obtained, we now choose $\beta = 0.125$, which is close to 0.1216. The weighting coefficients of extended linear interpolation equation is

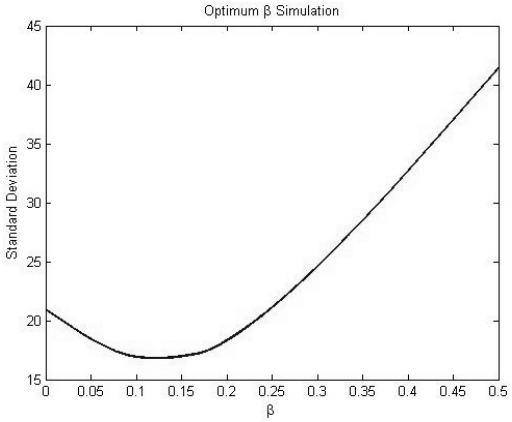
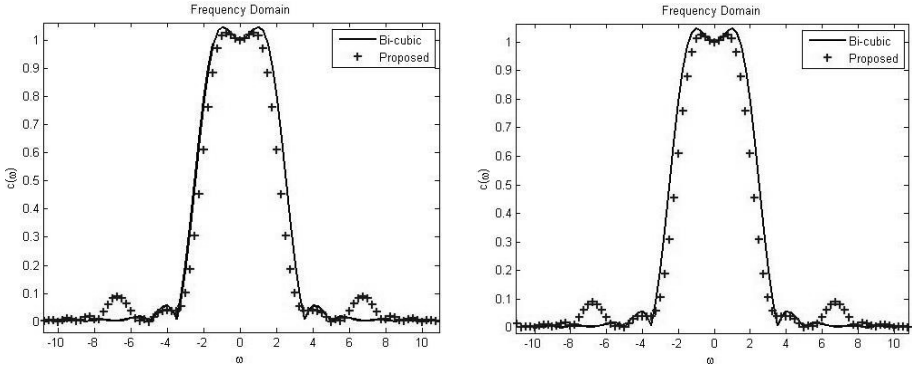


Fig. 3. The standard deviation curve for $\beta = 0$ to 0.5.

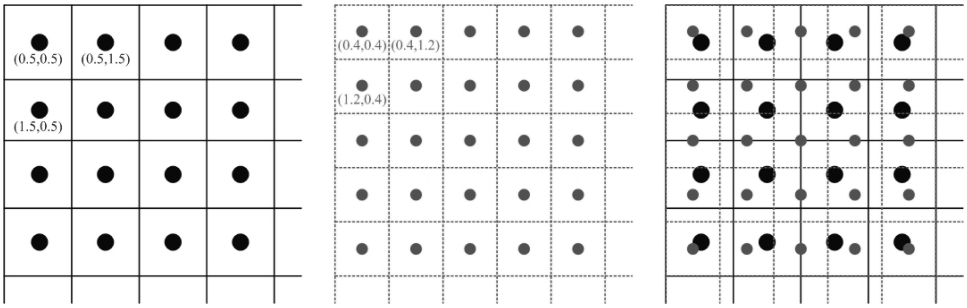


(a) (b)

Fig. 4. Frequency domain of the proposed method vs. bi-cubic interpolation for (a) $\beta = 0.1216$ (b) $\beta = 0.125$.

$$h_{EL_{0.125}}(s) = \begin{cases} -0.875 \times |s| + 1, & 0 \le |s| < 1 \\ -0.125 \times |s| + 0.125, & 1 \le |s| < 2 \\ 0, & 2 \le |s| \end{cases}. \tag{9}$$

The frequency response on the spectrum for $\beta = 0.1216$ and $\beta = 0.125$ are compared with bi-cubic interpolation, respectively, and illustrate in Fig. 4. The low-frequency response is very close to that of bi-cubic interpolation except that the high-frequency response is a bit higher when $\beta = 0.1216$. Thus, the interpolation quality for $\beta = 0.1216$ and $\beta = 0.125$ is approximately equal to that of bi-cubic interpolation. From the hardware implementation point of view, $\beta = 0.125$ is an optimal point to obtain good interpolation quality. The advantage of $\beta = 0.125$ is that the multiplication, as shown in Eq. (9), with distance s can be accomplished by shifting s to the right for 3 bits, instead of using a multiplier, in hardware implementation. Besides, the value is approximately equal to 0.1216 and the interpolation effect is similar to the effect of the ideal sinc-function.



(a) (b) (c)

Fig. 5. (a) The coordinates of a source image; (b)The interpolated coordinates at scale ratio = 1.25; (c)The interpolated pixels are mapped into source-image coordinates.

3.3 Coordinate Orientation and the Interval Calculation

For image scaling, the coordinate system of interpolated pixels is different from that of a source image. The coordinates of an interpolated pixel depend on scale ratio and thus, they require precise calculations of integer and fraction. Fig. 5 shows an example of scaled coordinate for scale ratio 1.25. In our approach, the origin of a source image in source coordinates is (0.5, 0.5); nevertheless, its location in memory is at (1, 1). For an interpolated image, the interpolated pixel has to be re-located in the scaled coordinates at first. The first interpolated pixel in scaled coordinates begins at $(\frac{1}{\text{Scale Ratio}} \times \frac{1}{2}, \frac{1}{\text{Scale Ratio}} \times \frac{1}{2})$. Thus, the next interpolated coordinates can be determined by increasing an interval value, which is 1/*Scale Ratio*, in x-direction or y-direction. All the interpolated coordinates can be obtained row by row; the coordinate of the position of the *n*th (*m*th) interpolated pixel in x-direction (y-direction) is

$$x_n = x_{n-1} + \frac{1}{Scale\ Ratio}, \tag{10}$$

$$y_m = y_{m-1} + \frac{1}{\text{Scale Ratio}}, \tag{11}$$

where x_{n-1} (y_{m-1}) is the coordinate of the previous interpolated pixel in x-direction (y-direction).

After the location (x_n, y_m) of an interpolated pixel Q is obtained, the row addresses of the source pixels in off-chip memory can be calculated according to the vertical coordinate (y_m) of the interpolated pixel. The row address (r_addr_i) of the source pixel $A_{i,j}$ in off-chip memory has to be determined first as in Eq. (12).

$$r\_addr_i = \begin{cases} \lfloor y_m \rfloor, & \text{fraction of } y_m \geq 0.5 \\ \lfloor y_m \rfloor - 1, & \text{fraction of } y_m < 0.5 \end{cases} \tag{12}$$

Afterward, r_addr_{i+1}, r_addr_{i+2}, and r_addr_{i+3}, which are the neighbors of r_addr_i, can be calculated by adding a constant q as:

$$r\_addr_{i+q} = r\_addr_i + q, \tag{13}$$

where q = 1, 2, and 3. Likewise, column addresses (c_addr_j, c_addr_{j+1}, c_addr_{j+2}, and c_addr_{j+3}) can be obtained by the same manners of Eqs. (12) and (13) except replacing y_m by x_n.

To simplify the weighting coefficient generation, the distance in vertical direction, s_v, has to be found before the calculation of vertical weighting coefficients. The distance s_v can be obtained from Eq. (14); similarly, the distance in horizontal direction, s_h, can be decided from Eq. (15).

$$s_v = y_m - (r\_addr_i + 0.5) \tag{14}$$
$$s_h = x_n - (c\_addr_j + 0.5) \tag{15}$$

3.4 Reduction of the Computation of Weighting Coefficients

In 1-dimensional interpolation, vertical and horizontal weighting coefficients have to be determined. The method of calculating vertical weighting coefficients is identical to the way of obtaining horizontal weighting coefficients. According to Fig. 6, Eq. (9) can be modified to find all the vertical weighting coefficients.

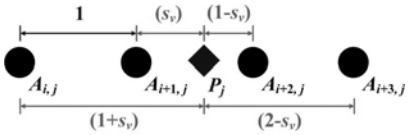


Fig. 6. Vertical interpolation.

$$\begin{aligned}
vw_i &= -0.125\times(1+s_v)+0.125 = -0.125\times s_v,\\
vw_{i+1} &= -0.875\times s_v + 1,\\
vw_{i+2} &= -0.875\times(1-s_v)+1 = 0.875\times s_v + 0.125,\\
vw_{i+3} &= -0.125\times(2-s_v)+0.125 = 0.125\times s_v - 0.125,
\end{aligned} \tag{16}$$

where vw_i, vw_{i+1}, vw_{i+2}, and vw_{i+3} are the vertical weighting coefficients of the corresponding source pixels $A_{i,j}$, $A_{i+1,j}$, $A_{i+2,j}$, and $A_{i+3,j}$; s_v is the distance between the source pixel $A_{i+1,j}$ and the virtual interpolated pixel P_j. Eq. (16) can be represented as Eq. (17), which requires only two adders and two subtractors. Thus, Eq. (17) involves shifting s_v to the right for 3 bits instead of using a multiplier to obtain $0.125s_v$.

$$\begin{aligned}
vw_i &= -0.125\times s_v\\
vw_{i+1} &= -0.875\times s_v + 1 = ((-vw_i) - s_v) + 1\\
vw_{i+2} &= 0.875\times s_v + 0.125 = s_v - (0.125\times s_v - 0.125) = s_v + (-vw_{i+3})\\
vw_{i+3} &= 0.125\times s_v - 0.125 = -(0.125 - (-vw_i))
\end{aligned} \tag{17}$$

Similarly, all the horizontal weighting coefficients can be found by

$$\begin{aligned} hw_j &= -0.125 \times s_h, \\ hw_{j+1} &= ((-hw_j) - s_h) + 1, \\ hw_{j+2} &= s_h + (-hw_{j+3}), \\ hw_{j+3} &= -(0.125 - (-hw_j)). \end{aligned} \tag{18}$$

3.5 Decomposition of 2-dimension 4 × 4 Interpolation

Fig. 1 shows the interpolated point, Q, and its 4 × 4 neighboring source pixels. Interpolation is performed in vertical direction first; from Eq. (19), four virtual pixels (P_j, P_{j+1}, P_{j+2}, and P_{j+3}), can be calculated from their vertical weighting coefficients (vw_i, vw_{i+1}, vw_{i+2}, and vw_{i+3}) by four 1-dimensional interpolations.

$$P_j = \sum_{q=0}^{3} A_{i+q,\,j} \times vw_{i+q}, \tag{19}$$

where i is the row positions of the corresponding virtual pixel P_j, $A_{i,j}$ is one of 4 × 4 neighboring source pixels, and vw_{i+q} is the vertical weighting coefficient of $A_{i+q,j}$. These virtual pixels are created for horizontal interpolation. Therefore, the interpolated pixel then can be calculated from horizontal interpolation:

$$Q = \sum_{q=0}^{3} P_{j+q} \times hw_{j+q}, \tag{20}$$

where hw_{j+q} is the horizontal weighting coefficient of P_{j+q}.

Observing Eqs. (19) and (20), all the interpolated points within the same row of Q have the same vertical weighting coefficients. Thus, these same vertical weighting coefficients are calculated only one time for each row and then proceeds the calculations of horizontal weighting coefficients of each interpolated pixel in that row.

In vertical interpolation, each set of source pixels is read once to create a virtual point that is used for interpolation. The virtual point may be used repeatedly for interpolating at the same row and all interpolated pixels are calculated row by row. Therefore, the number of memory access times, α, for interpolating a pixel by our method can be determined according to the following equation.

$$\alpha = \begin{cases} 4/\textit{Scale Ratio}, & \textit{Scale Ratio} > 0.25 \\ 16, & \textit{Scale Ratio} \le 0.25 \end{cases}. \tag{21}$$

4. THE PROPOSED HARDWARE ARCHITECTURE

The block diagram of the proposed hardware architecture, depicted in Fig. 7, includes the Coordinate Calculation Unit, the Weighting Coefficient Generator, the Vertical Interpolation Unit, the Virtual Pixel Buffer, and the Horizontal Interpolation Unit.

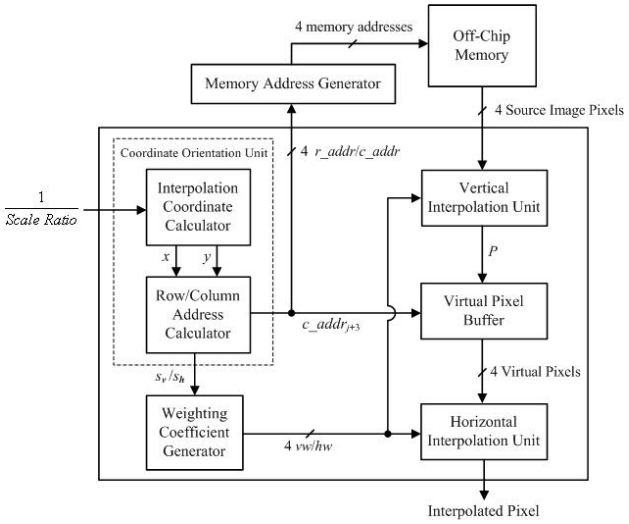


Fig. 7. The block diagram of the proposed hardware architecture.

4.1 The Coordinate Calculation Unit

The extended linear interpolation requires 16 source pixels that around an interpolated point of a source image. According to Eqs. (10) and (11), the coordinate of the interpolated point $Q(x_n, y_m)$ is obtained in the Interpolation Coordinate Calculator shown in Fig. 7. The row addresses and column addresses of its 16 neighboring source pixels then can be determined based on $Q(x_n, y_m)$. Afterward, the values of the 16 source pixels in off-chip memory can be obtained and extended linear interpolation can proceed.

The proposed method of extended linear interpolation decomposes the process as vertical and horizontal interpolations. For instance, in Fig. 1 the row addresses of $A_{i,j}$, $A_{i+1,j}$, $A_{i+2,j}$, and $A_{i+3,j}$, have to be found in order to interpolate where P_j is. The pixels in horizontal direction, which is the column address of the first virtual pixels, have to be determined in order to read the virtual pixels correctly from the Virtual Pixel Buffer and then process the horizontal interpolation. The above operations do not have to be executed in the circuit simultaneously; therefore, only one row/column addresses calculation circuit, Row/Column Address Calculator, is needed.

The circuit of Row/Column Address Calculator is depicted in Fig. 8. The operation of vertical or horizontal address orientation is controlled by the signal of vertical/horizontal; in other words, the signal determines the vertical (y_m) and the horizontal (x_n) coordinates. The data format of coordinate includes 10 bits of integer and other 10 bits of fraction. The 9th bit in the coordinate is the most significant bit of fraction and it determines whether the fraction of the coordinate is larger than 0.5 of decimal. The multiplexer (MUX_1) controlled by the 9th bit is used to output the correct memory address. If the signal vertical/horizontal is vertical, then the row addresses r_addr_i, r_addr_{i+1}, r_addr_{i+2}, r_addr_{i+3}, and the vertical interval s_v can be obtained according to Eqs. (12) to (14); otherwise, the column addresses c_addr_j, c_addr_{j+1}, c_addr_{j+2}, c_addr_{j+3}, and the horizontal interval s_h can be found from Eqs. (12), (13), and (15).

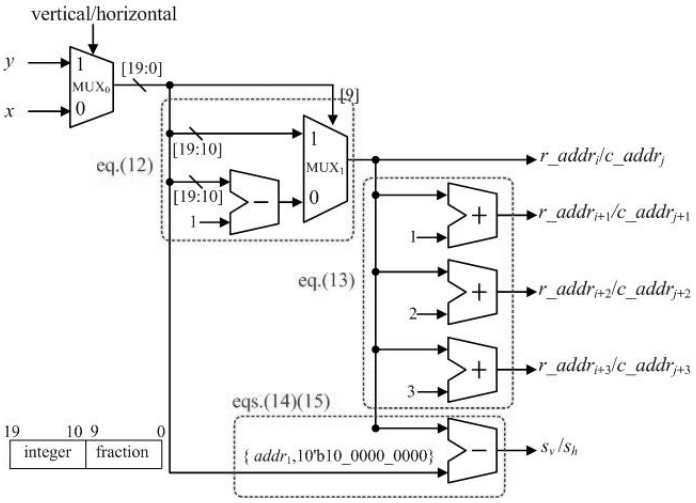


Fig. 8. Row/column address calculator.

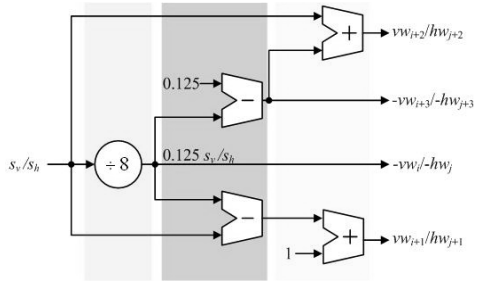


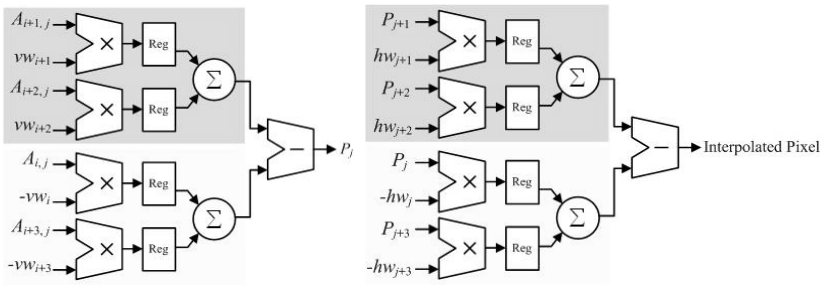
Fig. 9. The coefficient generator.

4.2 The Weighting Coefficient Generator

In the designed architecture, the calculations of vertical weighting coefficients and horizontal weighting coefficients are not generated at the same time although they both apply Eqs. (17) and (18) to determine all the coefficients. Therefore, the Weighting Coefficient Generator, as depicted in Fig. 9, is designed for producing vertical and horizontal weighting coefficients. The final outputs are $-vw_i$, vw_{i+1}, vw_{i+2}, and $-vw_{i+3}$ if the control signal, vertical/horizontal, is vertical; otherwise, the control is horizontal and the outputs are $-hw_j$, hw_{j+1}, hw_{j+2}, and $-hw_{j+3}$. The most computation effort in extended linear interpolation is the calculation of interpolation weighting coefficients. The complex operation is simplified in the generator, which includes only two adders and two subtractors.

4.3 The Vertical and Horizontal Interpolation Units

According to Eqs. (19) and (20), the vertical and horizontal interpolations have the same operation, but they have to execute in parallel to accelerate scaling speed. The logic circuits of the Vertical and Horizontal Interpolation Units are depicted in Fig. 10. The



(a) (b)

Fig. 10. (a) The vertical interpolation unit; (b) The horizontal interpolation unit.

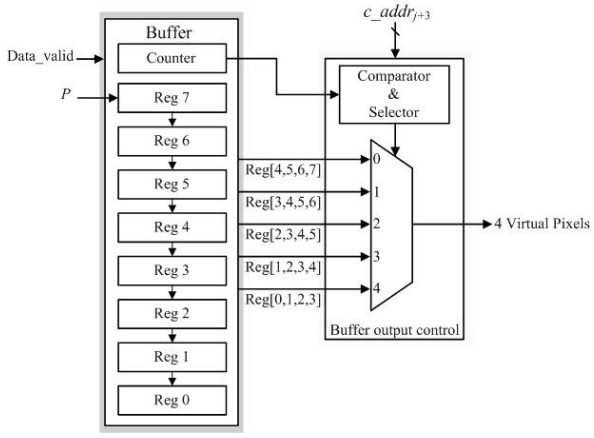


Fig. 11. The virtual pixel buffer.

multipliers and adders can efficiently perform the convolution of four input data ($A_{i,j}$, $A_{i+1,j}$, $A_{i+2,j}$, and $A_{i+3,j}$ / P_j, P_{j+1}, P_{j+2}, and P_{j+3}) and the corresponding 4 weighting coefficients and then generate the interpolated result quickly.

4.4 The Virtual Pixel Buffer

The virtual pixels created from vertical interpolation are stored in the Virtual Pixel Buffer, as shown in Fig. 7, where to be accessed in the process of horizontal interpolation. In general, scale ratio will determine both input and output data rates of this buffer. While scaling up, each virtual pixel may be reused for the horizontal interpolation such that the data rate of vertical interpolator is less than that of the horizontal interpolator in a period of time. Conversely, in scaling down, the virtual pixels generated by vertical interpolation are more than those required for the horizontal interpolation. In order to adjust the different data rates for various scale ratios, Virtual Pixel Buffer is designed and its circuit is demonstrated in Fig. 11, which includes a counter, 8 registers, and the circuit to control buffer output. In the process of scaling up, the buffer will temporarily suspend the operation of vertical interpolation, which cease the generation of virtual pixels, while the number of virtual pixels is more than certain amount that required by horizontal in-

terpolation. In the operation of scaling down, the horizontal interpolation will be halted if the buffer detects that the amount of valid virtual pixels in the Virtual Pixel Buffer approximately approaches the quantity needed for horizontal interpolation until more virtual pixels are ready. The counter records the column address of the newest generated virtual pixel stored in Reg7. The comparator and selector compares the counter and c_addr_{j+3} to determinate the outputs of the buffer. Obviously, the buffer along with the operation mechanism is uncomplicated.

5. EXPERIMENTAL RESULTS

To evaluate the performance of extended linear interpolation, we compared and analyzed the simulation of the proposed architecture with those of Winscale [8] and Nuno-Maganda [7]. The kernel of the proposed method is built up of first-order polynomial interpolation; it presents a lower computation complexity of interpolation than Nuno-Maganda [7]. Nuno-Maganda [7] uses 12 multipliers and 12 adders to generate weighting coefficients and 20 multipliers and 12 adders for each convolution operation. With compatible image quality, the proposed method applies 4 adders to generate weighting coefficients and 8 multipliers and 6 adders for each convolution operation. Our approach has reduced about 70% of hardware cost. In addition, Peak Signal to Noise Ratios (PSNR) is used to measure the quality of reconstruction images. The simulation results, shown in Tables 1 to 2 and Figs. 12 to 13, demonstrate that the proposed method presents a quality

Table 1. PSNRs of various scaling methods for 3/2 upscaling after 2/3 downscaling.

Method / Test images	3/2 → 2/3 (up/down)		
	Winscale [8]	Bi-cubic [7]	Proposed
Tank	30.35	35.63	35.29
Sailboat	27.05	32.29	32.12
Boat	27.36	31.95	32.11
Bridge	24.11	29.17	29.14
Goldhill	28.50	34.20	34.00
Lena	29.62	35.17	35.21
Peppers	29.71	34.62	34.41
Airplane	28.56	34.06	34.21

Table 2. PSNRs of various scaling methods for 2/3 downscaling after 3/2 upscaling.

Method / Test images	2/3 → 3/2 (down/up)		
	Winscale [8]	Bi-cubic [7]	Proposed
Tank	29.74	33.06	33.33
Sailboat	26.34	30.39	30.75
Boat	26.35	30.76	31.03
Bridge	23.39	26.91	27.27
Goldhill	27.81	31.75	31.95
Lena	29.02	33.73	33.97
Peppers	28.98	32.88	33.22
Airplane	28.01	33.05	33.12

(a) (b) (c) (d)

Fig. 12. (a) Original image of Lena; The image of 3/2 upscaling after 2/3 downscaled by (b) Winscale; (c) Bi-cubic; (d) Proposed.

(a) (b) (c) (d)

Fig. 13. (a) Original image of Lena; The image of 2/3 downscaling image after 3/2 upscaled by (b) Winscale (c) Bi-cubic (d) Proposed.

Table 3. Comparison of operations per interpolated pixel.

	Kim [8]	Nuno-Maganda [7]	Proposed
Approach	Winscale	Bi-cubic	Extended Linear
Pixel moving	6 adds	2 adds	2 adds
Weighting factor calculation	6 muls 2 adds	12 muls 12 adds	0 mul 4 adds
Pure filter operations	4 muls 3 adds	20 muls 12 adds	8 muls 6 adds
Memory access	4 reads 1 write	16 reads 1 write	α reads 1 write

$^{*}\alpha = \begin{cases} 4/\textit{Scale Ratio}, & \textit{Scale Ratio} > 0.25 \\ 16, & \textit{Scale Ratio} \le 0.25 \end{cases}$.

that is compatible to bi-cubic convolution interpolation. For example, Winscale [8] produces line-crawling effect in Figs. 12 (b) and 13 (b), whereas the proposed method presents the similar quality with bi-cubic interpolation in Figs. 12 (c), (d), 13 (c), and (d).

The simplification of the weighting coefficient functions demonstrated in Eqs. (17) and (18) decreases the hardware complexity and the number of memory access times substantially. Compared with other designs, the hardware costs were shown in Table 3. It shows that the hardware cost of ours is lower than the others. When scale ratio is greater than 0.25, number of memory access times to produce the virtual pixels of a row is fixed although the number of memory read times increases with the scale ratio in column.

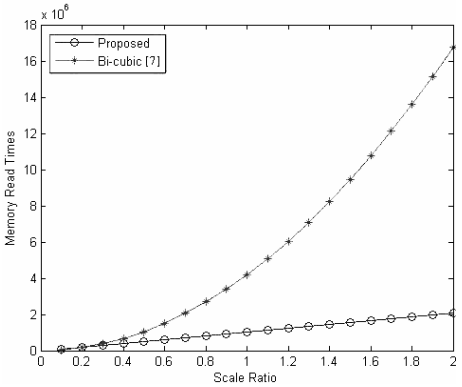
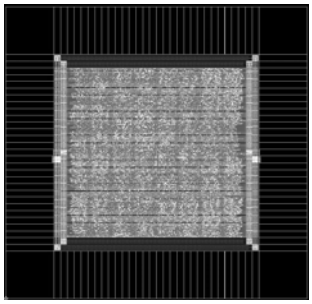


Fig. 14. Memory read times vs. scale ratio using a source image (512 × 512) for the proposed method and the bi-cubic interpolation.



Process	TSMC 0.13-μm
Frequency	267MHz
Gates count	25980
Power consumption	18.07 mW
Chip area	$450 \times 450 \mu m^2$

Fig. 15. The chip layout and specifications.

Therefore, the memory read times increase linearly when scale ratio is greater than 0.25. The comparison between the proposed method and the bi-cubic method for interpolating an image was shown in Fig. 14. It depicts the number of memory read times vs. scale ratio that varies from 0.1 to 2 for a source image with 512 × 512. The memory read times of the bi-cubic method increases with the scale ratio in row and column directions and obviously, it needs much more read times than our method when the scale ratio is greater than 0.25.

The number of adders and subtractors used to generate weighting coefficients in the proposed method are much less than the bi-cubic algorithm. In addition, the number of multipliers and adders for convolution operation has decreased substantially. Consequently, the proposed architecture has solved the problem of computation complexity of interpolation and furthermore, simplified the circuit and reduced chip area. The proposed architecture was implemented on the *Virtex-II FPGA*, and VLSI architecture that was synthesized by Synopsys Design Compiler with TSMC 0.13μm CMOS standard cell

Table 4. Comparison of FPGA Implementation.

	Kim [8]	Nuno-Maganda [7]	Proposed
Approach	Winscale	Bi-cubic	Extended Linear
Implementation technique	FPGA	FPGA/Virtex-II	FPGA/Virtex-II
CLBs	NA	890	376
Gates count	29000	NA	25980
Frequency	65MHz	100MHz	104.3MHz

library. The chip implementation of the proposed method with 10 fraction bits is illustrated in Fig. 15. With compatible image quality, the simulation results demonstrate that the proposed architecture has 25980 gates in a $450 \times 450 \mu m^2$ chip. In addition, the performance of hardware architecture was compared to that of Winscale [8] and Nuno-Maganda [7] in Table 4. The results indicate that the proposed architecture is more efficient than the others.

6. CONCLUSIONS

In this paper we proposed an efficient VLSI architecture of extended linear interpolation for digital image processing. The operation of extended linear interpolation requires 16 weighting coefficients generated from 16 neighboring pixels of source image. In other words, the operation of interpolation needs to read 16 neighboring source pixels from off-chip memory and generate 16 weighting coefficients of each interpolated point. Therefore, the number of memory read times for interpolating in the bi-cubic interpolation varies with scale ratio; the weighting coefficient is created with heavy computation. In our proposed method, the number of memory access times for interpolating a row is fixed and not related to the scale ratio. In addition, the number of adders and subtractors used to generate weighting coefficients in the proposed method are much less than the bi-cubic algorithm. Consequently, the proposed architecture has solved the problem of computation complexity of interpolation and furthermore, simplified the circuit and reduced chip area. The simulation results have shown that our high performance architecture presents a lower hardware cost than other interpolation methods to obtain a quality that is compatible to bi-cubic convolution interpolation. Besides, based on our technique, the high-speed VLSI architecture has been designed and implemented and it is able to process digital image scaling for HDTV [16] in real-time.

REFERENCES

1. T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: interpolation methods in medical image processing," *IEEE Transactions on Medical Imaging*, Vol. 18, 1999, pp. 1049-1075.
2. P. Thevenaz, T. Blu, and M. Unser, "Interpolation revisited," *IEEE Transactions on Medical Imaging*, Vol. 19, 2000, pp. 739-758.
3. S. S. Rifman, "Digital rectification of ERTS multispectral imagery," in *Proceedings of Symposium on Significant Results Obtained from BRTS-1*, Vol. 1, 1973, NASA

SP-327, pp. 1131-1142
4. S. K. Park and R. A. Schowengerdt, "Image reconstruction by parametric cubic convolution," *Computer Vision, Graphics, and Image Processing*, Vol. 23, 1983, pp. 258-272.
5. R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Signal Processing*, Vol. 29, 1981, pp. 1153-1160.
6. E. Meijering and M. Unser, "A note on cubic convolution interpolation," *IEEE Transactions on Image Processing*, Vol. 12, 2003, pp. 477-479.
7. M. A. Nuno-Maganda and M. O. Arias-Estrada, "Real-time FPGA-based architecture for bicubic interpolation: an application for digital image scaling," in *Proceedings of International Conference on Reconfigurable Computing and FPGAs*, 2005.
8. C. H. Kim, S. M. Seong, J. A. Lee, and L. S. Kim, "Winscale: An image-scaling algorithm using an area pixel model," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, 2003, pp. 549-553.
9. E. Aho, J. Vanne, K. Kuusilinna, and T. D. Hamalainen, "Comments on Winscale: an image-scaling algorithm using an area pixel model," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, 2005, pp. 454-455.
10. J. W. Hwang and H. S. Lee, "Adaptive image interpolation based on local gradient features," *IEEE Signal Processing Letters*, Vol. 11, 2004, pp. 359-362.
11. E. H. W. Meijering, K. J. Zuiderveld, and M. A. Viergever, "Image reconstruction by convolution with symmetrical piecewise *n*th-order polynomial kernels," *IEEE Transactions on Image Processing*, Vol. 8, 1999, pp. 192-201.
12. T. Blu, P. Thevenaz, and M. Unser, "Complete parameterization of piecewise-polynomial interpolation kernels," *IEEE Transactions on Image Processing*, Vol. 12, 2003, pp. 1297-1309.
13. T. Blu, P. Thevenaz, and M. Unser, "Linear interpolation revitalized," *IEEE Transactions on Image Processing*, Vol. 13, 2004, pp. 710-719.
14. S. E. Reichenbach and F. Geng, "Two-dimensional cubic convolution," *IEEE Transactions on Image Processing*, Vol. 12, 2003, pp. 857-865.
15. J. Shi and S. E. Reichenbach, "Image interpolation by two-dimensional parametric cubic convolution," *IEEE Transactions on Image Processing*, Vol. 15, 2006, pp. 1857-1870.
16. S. C. Hsia, B. D. Liu, J. F. Yang, aad C. H. Huang, "An NTSC to HDTV video conversion system by using the block processing concept," *IEEE Transactions on Consumer Electronics*, Vol. 40, 1994, pp. 216-224.

Chung-Chi Lin (林正基) received the M.S. degree in Computer Science from University of Houston, Texas, U.S.A., in 1983, and the Ph.D. degree in Engineering Science and Technology from National Yunlin University of Science and Technology, Taiwan, in 2009, respectively. Since 1984, he has been a Lecturer with the Department of Computer Science, Tunghai University, Taiwan. His current research interests include image processing, digital signal processing, and system-on-chip design.

Ming-Hwa Sheu (許明華) received the B.S. degree in Electronics Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 1986, and the M.S. and Ph.D. degrees in Electrical Engineering from National Cheng Kung University, Tainan, in 1989 and 1993, respectively. From 1994 to 2003, he was an Associate Professor at National Yunlin University of Science and Technology, Touliu, Taiwan. Currently, he is a Professor in the Department of Electronic Engineering, National Yunlin University of Science and Technology. His research interests include CAD/VLSI, digital signal process, algorithm analysis, and system-on-chip design.

Huann-Keng Chiang (江煥鏗) received the M.S. and Ph.D. degrees in Electrical Engineering from the National Cheng Kung University, Taiwan, in 1987 and 1990, respectively. He is a Professor in the Department of Electrical Engineering, National Yunlin University of Science and Technology, Taiwan. His research interests include automatic control, digital control, grey theory and motor servo control.

Chishyan Liaw (廖啟賢) received the B.S. and M.S. degrees in Electrical Engineering from Florida Institute of Technology, Florida, U.S.A., in 1984 and 1985, respectively. Since 1989, he has been a Lecturer with the Department of Computer Science and Information Engineering, Tunghai University, Taiwan. His current research interests include image processing, computer graphic, artificial neural networks, and logic design.

Zeng-Chuan Wu (吳曾傳)received the B.S. and M.S. degrees in Electronics Engineering from National Yunlin University of Science and Technology, Taiwan, in 2004 and 2006, respectively. His research interests include digital IC design, image processing, and associated FPGA prototyping verification.

Wen-kai Tsai (蔡文凱) is a Ph.D. candidate at Graduate School of Engineering Science and Technology, National Yunlin University of Science and Technology, Taiwan. He received the B.S. and M.S. degrees in Electronics Engineering from National Yunlin University of Science and Technology, Taiwan, in 2003 and 2006, respectively. His research interests include digital signal processing and image processing.