Hindawi Publishing Corporation
VLSI Design
Volume 2014, Article ID 872501, 10 pages
http://dx.doi.org/10.1155/2014/872501

Review Article

VLSI Architectures for Image Interpolation: A Survey

C. John Moses,[1] D. Selvathi,[2] and V. M. Anne Sophia[1]

[1] *Department of ECE, St. Xavier's Catholic College of Engineering, Nagercoil 629003, India*
[2] *Department of ECE, Mepco Schlenk Engineering College, Sivakasi 626005, India*

Correspondence should be addressed to C. John Moses; jofjef@yahoo.com

Received 18 October 2013; Revised 20 April 2014; Accepted 5 May 2014; Published 19 May 2014

Academic Editor: Marcelo Lubaszewski

Image interpolation is a method of estimating the values at unknown points using the known data points. This procedure is used in expanding and contrasting digital images. In this survey, different types of interpolation algorithm and their hardware architecture have been analyzed and compared. They are bilinear, winscale, bi-cubic, linear convolution, extended linear, piecewise linear, adaptive bilinear, first order polynomial, and edge enhanced interpolation architectures. The algorithms are implemented for different types of field programmable gate array (FPGA) and/or by different types of complementary metal oxide semiconductor (CMOS) technologies like TSMC 0.18 and TSMC 0.13. These interpolation algorithms are compared based on different types of optimization such as gate count, frequency, power, and memory buffer. The goal of this work is to analyze the different very large scale integration (VLSI) parameters like area, speed, and power of various implementations for image interpolation. From the survey followed by analysis, it is observed that the performance of hardware architecture of image interpolation can be improved by minimising number of line buffer memory and removing superfluous arithmetic elements on generating weighting coefficient.

1. Introduction

In digital image scaling, image interpolation algorithms are used to convert an image from one resolution to another resolution without losing the visual content in the image. In the colour, image interpolation is the process of estimating the missing colour samples to reconstruct a full colour image [1]. Image scaling is widely used in many fields, ranging from consumer electronics, such as digital camera, mobile phone, tablet, display devices and medical imaging like computer assisted surgery (CAS) and digital radiographs [2]. In many applications, from consumer electronics to medical imaging, it is desirable to improve the restructured image quality and processing performance of hardware implementation [3]. For example, a video source with a 640 × 480 video graphics arrays (VGA) resolution may need to fit the 1920 × 1080 resolution of a high definition multimedia interface (HDMI).

Image up scaling [4] methods are implemented for a variety of computer equipments like printers, digital television, media players, image processing systems, graphics renderers, and so on. On the other hand, high resolution image may need to be scaled down to a small size in order to fit the lower resolution of small liquid crystal display panels. That is, the image scaling is a challenging and very significant issue in digital image processing [5]. The problem is to attain a digital image to be displayed on a large bitmap from unique data sample in a smaller grid, and this image should appear like it had been attained with a sensor having the resolution of the up-scaled image or, as a minimum, present a "natural" texture. Methods that are normally used to solve the problem (i.e., pixel replication, bilinear, or bi-cubic interpolation) do not realize these requirements, producing images with visual artifacts like pixelization, jagged contours, and over smoothing. Therefore, a set of advanced adaptive methods have been presented [4].

The interpolation algorithms can be classified as adaptive and nonadaptive algorithms [3]. Nonadaptive algorithms perform interpolation in a fixed pattern for every pixel, that is, by averaging neighbouring pixels. This causes an artifact, the zipper effect in the interpolated image. This leads to a blur across the edges. This technique is fixed irrespective of the input image features. Adaptive algorithms use both spectral and spatial features present in the neighbourhood pixel in order to interpolate the missed pixel as close to

the original as possible [6]. To restore more accurate and visually pleasing results, image spatial or spectral correlation or both have been exploited. Adaptive algorithms can detect local spatial features present in the pixel neighbourhood and then make effective choices as to which predictor to use for that neighbourhood. The result is a reduction or elimination of zipper-type artifacts.

Further, the interpolation methods are generally classified into two classes as [7] spatial domain and frequency domain. Spatial domain technique is adopted frequently for real-time applications because it has low complexity. The other method uses proper transform such as discrete cosine transform (DCT), discrete fourier transform (DFT), or wavelet transform to scale images in the frequency domain. Due to its higher computational complexity and memory requirements, the frequency domain technique is not appropriate for real-time or low cost applications.

In many practical real-time applications, the scaling process is included in end user equipment and it can be implemented by VLSI. VLSI architecture is implemented using either FPGA or application specific integrated circuit (ASIC). FPGA is designed to be configured by a customer or a designer often manufacturing. Thus, FPGAs are used to construct reconfigurable computing systems with high performance and low cost. But ASIC is not reconfigurable. Recent FPGA platforms have millions of gates, up to 500 MHz clock frequency, reasonably large on chip memory and fast input output interface [8]. FPGAs are widely used for real-time implementation of various image processing algorithms such as motion detection, image enhancement, image correlation, and image compression. In this paper, different hardware architectures of interpolation methods are discussed and analysed their performance.

Section 2 explains the hardware architecture of image interpolation, Section 3 provides the important characteristics of different hardware implementations of image interpolation algorithms using FPGA/TSMC. Section 4 deals with a detailed analysis of different optimization parameters of VLSI architecture like gate count, speed, and power. Finally, Section 5 concludes the analysis.

2. Hardware Architecture of Image Interpolation

The hardware architecture of image interpolation includes the coordinate accumulator, line buffer, weighting coefficients generator, vertical interpolator, and horizontal interpolator. Each of the units performs specified operations for the interpolation [14]. The hardware architecture with its different units is shown in Figure 1.

2.1. Co-Ordinate Accumulator. The coordinate accumulator calculates the corresponding coordinate of the current target pixel to be interpolated. Its inputs are two scaling factors: one is for the horizontal direction scale_x and the other is for the vertical direction scale_y. These factors provide the scaling

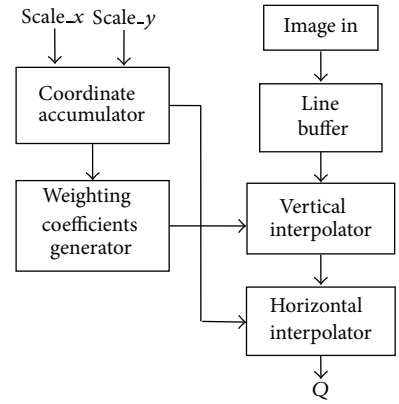

Figure 1: Block diagram of the hardware architecture of image interpolation.

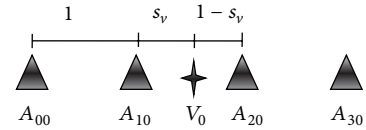

Figure 2: Vertical interpolation.

ratio of each direction. The current target pixel's coordinate (x_c, y_c) is calculated as [7]

$$\begin{aligned} x_c &= x_{c-1} + \text{scale\_}x, \\ y_c &= y_{c-1} + \text{scale\_}y. \end{aligned} \tag{1}$$

Equation (1) represents the coordinate of the previous target pixel.

2.2. Line Buffers (LB). The line buffers are used to store source image data.

2.3. Weighting Coefficient Generator. In 1D 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. The way of calculating vertical and horizontal weighting coefficients is shown in Figures 2 and 3, respectively [14].

According to Figure 2 the vertical weighting coefficients can be calculated as

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

where vw_i, vw_{i+1}, vw_{i+2} and vw_{i+3} are the vertical coefficients of the corresponding source pixels A_{00}, A_{10}, A_{20}, and A_{30}. s_v is the distance between the source pixel $A_{i+1,j}$ and the vertical interpolation pixel v_0.

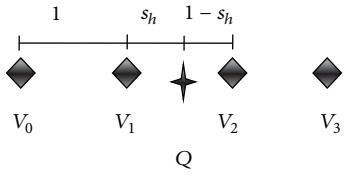

Figure 3: Horizontal interpolation.

Similarly, based on Figure 3, all the horizontal weighting coefficients can be found by

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

where hw_j, hw_{j+1}, hw_{j+2}, and hw_{j+3} are the horizontal coefficients of the corresponding virtual pixels V_0, V_1, V_2, and V_3. S_h is the distance between the vertical interpolation pixel V_0 and target pixel Q.

2.4. Vertical Interpolator. After the vertical weighting coefficients are obtained, the vertical interpolation operation is performed by using (2). For the 16 source image pixels, four new pixel values are obtained from vertical interpolation. These pixels are called "virtual pixels."

2.5. Horizontal Interpolator. The vertical interpolated pixels are used to interpolate the target pixel through the horizontal direction. The horizontal weighting coefficients are processed by using (3) to obtain the target pixel Q.

3. Interpolation Methods

This chapter briefs a review of different image interpolation algorithms such as winscale, bi-cubic, linear, polynomial convolution, bilinear, and adaptive scaling algorithms. Various algorithms are discussed by their hardware characteristics and visual quality. The hardware characteristics such as area utilization and speed and power consumptions are mainly focused upon. The quality of the interpolation algorithms can be expressed in dB (decibel) with peak signal-to-noise ratio (PSNR) of scaled image and the original image.

3.1. Digital Image Scaling Algorithm. Andreadis and Amantiadis proposed an image interpolation algorithm for both grayscale and colour images of any resolution in any scaling factor [5]. This algorithm uses a mask of maximum four pixels and calculates the final luminosity of each pixel combining two factors such as the percentage of area that mask covers from each source pixel and the difference in luminosity between the source pixels. This interpolation operates on linear area domain and uses continuous area filtering. It can perform both upscale and downscale processes for fast real-time implementation. This method is implemented on Quartus II FPGA with the operating frequency of 55 MHz. The hardware architecture of this interpolation uses 20 additions and 13 multiplications. The root mean square error (RMSE) of this method is very less than the RMSE of other interpolation scheme like nearest neighbour, bilinear, and winscale.

3.2. Winscale Image Interpolation. Image scaling using winscale [9] is proposed by Kim et al. The aim of this scaling algorithm is to improve the quality and to reduce the computational complexity by reducing number of operations per pixel. This algorithm can perform both scale up and scale down transform using an area pixel model rather than a point pixel model. To provide low complexity, it uses a maximum of four pixels of an original image to calculate one pixel of a scaled image. Also, it can provide better quality by the characteristics such as fine-edge and changeable smoothness. The hardware of winscale is implemented using an FPGA for displaying scenes in a liquid crystal display panel. winscale has good scale property with low complexity. It preserves the edge characteristic of an image well and handles streaming data directly and requires only small amount of memory. Thus, image interpolation can be done by four-line buffers in spite of up/down ratio. Further, this work tells that the winscale algorithm has low RMSE and it has better image quality than the bilinear algorithm. This implementation utilizes 29000 gates at operating frequency of 65 MHz. winscale can be used in various digital display devices that need image scaling, especially in applications that require good image quality with low hardware cost.

3.3. VLSI Design of Winscale for Digital Image Scaling. Lin et al. proposed a VLSI design of winscale algorithm [10] for digital image scaling. This winscale technique uses an area pixels model to evaluate a scaled pixel. The original image and the scaled image are treated as rectangular and the intensity pixel is evenly scattered in rectangular area. The scaled image could be obtained by source image with scaling up/down in various ratios. The scale ratio that comprises horizontal scale ratio (HSR) and vertical scale ratio (VSR) is greater than 1.0 for scale up and less than 1.0 for scale down. The VLSI architecture of this scaling algorithm is implemented by UMC 0.18 μm CMOS standard cell library. The total gate count is 17414 with the operating frequency of 130.24 MHz. This architecture occupies $450 \times 450\ \mu m^2$ core area of chip and total power consumption is 19.41 mW. Further, this can be used for processing image interpolation for high definition television (HDTV) in real time.

3.4. Bi-Cubic Convolution Interpolation. An efficient VLSI design of bi-cubic convolution interpolation [11] for digital image processing algorithm is presented by Lin et al. The architecture of reducing the computational complexity of generating coefficients and decreasing the number of memory access times is proposed. The hardware architecture of this method is based on Figure 1. In this method, the number of memory access time, for interpolating a row is

fixed and does not vary with the scale ratio. This method generates weighting coefficients as shown in Figures 2 and 3. The number of multipliers and adders used to generate coefficients is less than the original bi-cubic algorithm. It provides a simple hardware architecture design and low computation cost and it is also easy to implement. Based on this technique, the high-speed VLSI architecture has been successfully designed and implemented with TSMC 0.13 μm standard cell library. The simulation results demonstrate that the high performance architecture of bi-cubic convolution interpolation at 279 MHz with 30643 gates in a 498×498 μm^2 chip is able to process digital image scaling for HDTV, in real time. The hardware of this algorithm is also analyzed by using FPGA. The FPGA implementation demonstrates that the algorithm requires only about 437 logic blocks but the bi-cubic algorithm which is proposed by Nuño-Maganda and Arias-Estrada [16] requires about 890 logic blocks.

3.5. Real-Time FPGA Linear Convolution Interpolation. A linear interpolation [12] is presented by Lin et al., which is low-cost hardware architecture with digital image scaling for real-time requirements. This architecture is also based on Figure 1 and this generates weighting coefficients based on Figures 2 and 3. The scheme has the advantage of low operation complexity which reduces the coefficient generating effort and hardware cost with the interpolation quality, compatible to that of bi-cubic convolution interpolation [16]. The operation of linear convolution interpolation requires 16 weighting coefficients, generated from 16 neighbouring pixels of source image. Therefore, the number of adders and subtractors used to generate weighting coefficients in this method are much less than the bi-cubic algorithm [16]. The hardware architecture of this algorithm is implemented on Virtex-II FPGA. Consequently, this architecture has solved the problem of computational complexity of interpolation and simplified the hardware circuit. The high performance architecture of extended linear interpolation can be simulated at 104 MHz with 379 LBs (Logic Blocks) which is able to process digital image scaling. This architecture requires about 26200 gates but the architecture which is proposed by Kim et al. [9] requires about 29000 gates. In addition, this method achieves that higher PSNR (33.12). But the winscale and bi-cubic achieve lower PSNR as 28.1 and 33.05, respectively for the 3/2 upscaling image after 2/3 downscaling.

3.6. Extended Linear Interpolation. An extended linear interpolation for real-time digital image processing is presented by Lin et al. [13]. This image interpolation is a low cost architecture with the interpolation quality compatible to that of bi-cubic convolution interpolation [22]. This method has the similar hardware architecture as shown in Figure 1. Further, this method has similar visual quality (PSNR) as the bi-cubic interpolation algorithm and higher PSNR than the winscale algorithm [9]. The architecture is capable of reducing the computational complexity of generating weighting coefficients. A 2-dimensional interpolation method is decomposed as two 1-dimensional operations (vertical interpolation and horizontal interpolation) as shown in Figures 2 and 3, respectively. This is to solve the problem of computational complexity of interpolation and furthermore to simplify the circuit, and to reduce chip area. This architecture is implemented on the Virtex-II FPGA, and the VLSI architecture has been successfully designed and implemented with TSMC 0.13 μm standard cell library. The high performance architecture of extended linear interpolation that can be simulated at 267 MHz with 26200 gates in a 452×452 μm^2 chip is able to process digital image scaling for HDTV in real time.

3.7. Efficient Architecture of Extended Linear Interpolation (EAELI). An efficient extended linear image interpolation is proposed by Lin et al. [14]. The hardware architecture of this interpolation is shown in Figure 1. The extended linear interpolation requires 16 weighting coefficients produced from 16 neighbouring pixels of source image. This method reduces computational complexity of generating weighting coefficients. The principles of generating coefficients are shown in Figures 1 and 2 and by (2) and (3). The number of arithmetic elements, such as adders and subtractors for generating weighting coefficients, are less than the bi-cubic [16] and winscale [9] image interpolations. This architecture is designed on Virtex-II FPGA and with TSMC 0.13 technologies. This utilizes 25980 gates at 267 MHz to process image interpolation on HDTV in real time. The FPGA implementation tells that this architecture utilizes only about 379 configurable logic blocks (CLBs) but the bi-cubic architecture [16] requires about 890 CLBs. Furthermore, this method provides higher PSNR than winscale [9] and bi-cubic [16] algorithms. The achieved PSNR of this method is 35.29 for the test image (tank) for 3/2 upscaling after 2/3 downscaling.

3.8. Piecewise Linear Convolution Interpolation. A piecewise linear convolution interpolation, with third-order approximation, for real-time image processing, is presented by Lin et al. [20]. This work presents a high-performance architecture of a piecewise linear convolution interpolation for digital image. The hardware architecture of this interpolation is based on Figure 1. The kernel of this method, built up of piecewise linear polynomial, approximates the ideal sinc-function in interval $[-2, 2]$. The number of arithmetic elements like adders and subtractors used for generating weighting coefficients are very less than the bi-cubic algorithm. The number of CLBs utilized by Virtex-II FPGA for various algorithms such as bi-cubic [11, 16] and this algorithm are 890, 437, and 393, respectively. This is able to process various-range image scaling for HDTV in real time. Thus, the architecture reduces the computational complexity of generating weighting coefficients and provides a simple hardware architecture design and low computation cost and it is easy to meet real-time requirement. The architecture is implemented on the Virtex-II FPGA, and the VLSI architecture has been designed and implemented with TSMC 0.13 μm standard cell library. The simulation results indicate that the interpolation quality of the proposed architecture is better than cubic convolution interpolations [11, 16]. This implementation has higher PSNR (49.53) than the keys (49.29) and bi-cubic (46.38) for the test

image (airplane) of scaled from 3/4 downscaling after 4/3 upscaling.

3.9. An Edge-Oriented Image Scaling Algorithm. An edge-oriented area-pixel scaling algorithm [15] has been presented by Chen et al. This algorithm aims to achieve low cost; the area-pixel scaling technique is implemented with low-complexity VLSI architecture in this design. A simple edge catching technique is adopted to preserve the image edge features effectively so as to achieve better image quality. This scaling processor can support floating-point magnification factor and preserve the edge features efficiently by taking into account the local characteristic that existed in those available source pixels around the target pixel. Furthermore, it handles streaming data directly and requires only small amount of memory like one-line buffer rather than a full frame buffer. The FPGA implantation of this method is obtained by Xilinx Virtex-II XC2VP50. The FPGA implementation utilizes only about 581 CLBs but the bi-cubic [16] algorithm utilizes about 890 CLBs. Thus, the seven stage pipelined VLSI architecture of this image scaling processor contains 10.4 K gate counts and yields a processing rate of about 200 MHz by using TSMC 0.18 μm technology, but the bi-cubic architecture [16] utilizes about 29 K gates. The quality comparison shows that this method has higher PSNR as 38.16 than the other methods such as nearest neighbour, bilinear, bi-cubic [16], area-pixel scaling [9], and winscale [5] algorithms. The PSNR is obtained for the test image (crowd) for image reduction from the size of 600×600 to 512×512.

3.10. A Novel Interpolation Algorithm (NICRMA). Huang et al. presented a novel scaling algorithm for the implementation of (2D) image scalar [7]. This interpolation method is based on the interpolation error theorem. It solves the blurring and checkerboard effects caused by linear interpolation, and efficiently enhance the edge features of scaled images. This method can be conveniently applied to image zooming using both integer and noninteger magnification factors. The hardware architecture of this method is based on Figure 1 and it uses four-line-memory buffer. In the VLSI architecture, the cooperation and hardware sharing techniques have been used, which greatly reduce hardware cost requirements. The number of arithmetic components like adders and subtractors is very less than winscale [9, 15] algorithms. Using a nine-stage pipeline, the scaling circuit contains 13 K gate counts and yields a processing rate of approximately 278 MHz using TSMC 0.13 μm technology. Simulation results indicate a clock period of 3.6 ns and a processing rate of 278 megapixels per second. This implementation can also provide better PSNR than the winscale algorithms.

3.11. Bicubic Interpolation. One of the most extended algorithms for image scaling is bi-cubic interpolation [16]. In this work, hardware architecture for bi-cubic interpolation (HABI) is proposed. The HABI is integrated by three main blocks: the first block generates the coefficients, which implements the bi-cubic function to be used in HABI; the second block performs the interpolation process and the third one is a control unit that synchronizes the processing and the pipeline stages. The architecture works with monochromatic image, but it can be extended for working with RGB colour images. This design description is implemented on Xilinx Virtex-II pro FPGA. The system runs 10 times faster than an Intel Pentium 4-based PC at 2.4 GHz. The hardware architecture requires about 890 CLBs, 28 BlockRAMs at 100 MHz operating frequency, and 3.50 ms processing time. The architecture is a compact and efficient module that can be combined with other high performance architecture to create more robust application like, tracking system, robotic, and mobile applications. This architecture can also be updated to more parallelism without changing control and memory.

3.12. First-Order Polynomial Convolution Interpolation. Fast first-order polynomial convolution interpolation (FFOPCI) for real-time digital image reconstruction [19] is given by Lin et al. This work presents high-performance architecture of a novel first-order polynomial convolution interpolation for digital image scaling. The architecture is based on Figure 1. Generally, a better quality of image interpolation is achieved by using higher order model but that requires complex computations. This work tries to reduce the computational effort by using first-order polynomial convolution with compatible quality. The kernel of the proposed method is built up of first-order polynomials and it approximates the ideal sinc-funtion in the interval $[-2, 2]$. This method reduces the computational complexity of generating weighting coefficients and provides a simple hardware architecture design and low computation cost, and it easily meets real-time requirements. The architecture is implemented on the Virtex-II FPGA, and the VLSI architecture has been designed and implemented with the TSMC 0.13 μm standard cell library. The number of adders and subtractors used to generate weighting coefficients in the architecture is much less than that of bi-cubic interpolation [11]. Consequently, this architecture has solved the problem of computational complexity of interpolation and furthermore simplified the circuit, and reduced chip area. The Virtex-II FPGA utilizes 414 CLBs for this algorithm, but the Virtex-II utilizes 483 CLBs for bi-cubic [11] algorithm.

3.13. Bilinear Interpolation. A novel approach to real-time bilinear interpolation is proposed by Gribbon and Bailey [22]. This work presents a real-time bilinear interpolation, which is useful in applications, such as lens distortion correction where the input coordinates follow a curved path that spans multiple rows. Gribbon and Bailey aim to improve the quality of interpolated image with high speed. To improve the speed of operation a special caching is proposed to retrieve the required pixels in a single clock cycle. Also, they use the three point interpolation instead of using the normal four point interpolation. The hardware architecture of this interpolation is implemented on Spartan-II FPGA. The FPGA implementation utilizes about 329 CLBs out of 1172 CLBs and three numbers of BlockRAM out of 14 BlockRAMs. The major drawback of this system is that the use of the three-point interpolation instead of using the normal four-point

interpolation generates additional errors into the measured pixel values. Bilinear interpolation can be used for lens distortion correction, but it is complicated by the truth that input coordinates do not emerge on straight lines.

3.14. Parallel Bilinear Interpolation. Generalised parallel bilinear interpolation architecture for vision systems [21] is given by Fahmy. Bilinear image interpolation is extensively used in computer vision for generating pixel values for locations that lie off the pixel grid in an image. For every subpixel, the values of four neighbours are utilized to calculate the interpolated value. This work improves memory access by parallelism in embedded memories and increases the speed of interpolation. This method attains performance of 250 M samples per second in a modern device. The memory requirements of this method differ linearly with image size. For larger images that would need external memory, the resulting speed would depend on the greatest access speeds for the external memory. The hardware architecture of this method is implemented on Xilinx Virtex 4 XC4VSX55 FPGA. This architecture utilizes 362 Slices, 256 numbers of 18 K bit Block memories, and three numbers of DSP48 blocks. This system can operate at 250 MHz and can process 512×512 pixel images. This system is above 30 times faster as compared with the software implementation. This also suggested that Virtex-5 device can be used for processing 1024×1024 pixel images. This system can be used for computer vision. The major drawback of this system is that it requires a large memory.

3.15. Adaptive Scalar Bilinear Interpolation. To implement a 2D image scalar, a novel scaling algorithm [3] has been proposed by Chen et al. Bilinear interpolation, a clamp filter, and a sharpening spatial filter are involved in this algorithm. For its low complexity and high quality, the bilinear interpolation algorithm is used. Generally, bilinear interpolation produces blurring and aliasing effects. To reduce these effects, prefilters such as clamp and sharpening spatial filter are added. By using 5×5 combined convolution filter instead of 3×3 matrix coefficients, the memory requirement and computation complexity are reduced. The VLSI architecture is optimized by cooperation and hardware sharing. The hardware architecture of this interpolation scheme is implemented on FPGA and designed by using TSMC 0.18 μm and TSMC 0.13 μm CMOS libraries. The FPGA implementation of this scheme utilizes about 9.28 K gates, but the winscale [9, 10] algorithms utilize about 29 K gates and 17.4 K gates, respectively. The bi-cubic algorithms [11] and real-time FPGA [12] also utilize a large amount of gate as 30.6 K and 26.2 K, respectively, by comparing this method. This VLSI architecture of bilinear interpolation algorithm can achieve 280 MHz and its chip area is 46418 μm^2 synthesized by a 0.13 μm CMOS process. Furthermore, this method has higher PSNR (32.07) for the test image (splash) than the winscale [5, 9, 10], bi-cubic [11, 12] and other bilinear algorithms. This higher PSNR is achieved by using combined filter, adaptive technology, and bilinear interpolation. Thus, this algorithm provides a low cost, low power consumption, low memory demand, high performance,and good quality VLSI architecture for many video and image scaling applications.

3.16. SL Chen's Image Scaling Algorithm. Low cost, high quality image scaling algorithm [17] is proposed by Chen. This image scaling algorithm consists of a sharpening spatial filter, a clamp filter, and a bilinear interpolation. Bilinear interpolation is used to reduce the blurring and aliasing artifacts. The sharpening spatial and clamp filters are added as prefilters to minimize the memory buffers and computing resources. A T-model and inversed T-model convolution kernels are produced for realizing the sharpening spatial and clamp filters. In addition, the combined filter is designed by combining two T-models or inversed T-model filters which requires only a one-line buffer memory. Chen introduces sharing of computing resources and hardware to reduce the computational effort and hardware cost. The number of multiplication for this scheme is about four but the other algorithms [12, 17, 19] require 13, 10, and 32 multiplications, respectively. Also, this work requires only one-line buffer but winscale [19] and bi-cubic [12] require four- and six-line buffers, respectively. The VLSI architecture of this method contains 6.08 K gate counts, and the chip area is 30378 μm^2 synthesized by TSMC 0.13 μm CMOS process. It consumes 6.9 mW at 280 MHz operating frequency with a 1.1 V supply voltage. The PSNR of this method is 28.78 for the test image (Doub.) by using combined filter and bilinear interpolation. This PSNR is higher than the PSNR of other methods such as bilinear, winscale [19], bi-cubic [12], edge-enhanced [17], and other adaptive [5] image interpolation algorithms.

3.17. Edge-Enhanced Adaptive Scalar. A less computation complexity adaptive edge-enhanced scalar has been proposed [18] by Chen for 2D image interpolation applications. In this work, Chen aims to increase the interpolation quality, to reduce the computational complexity, and to reduce the hardware cost. This method includes a linear space-variant edge detector, a less complexity spatial filter, and a bilinear interpolation. The edge detector is used to find the image edges. The spatial filter is used as a prefilter which reduces the blurring effect of bilinear interpolation. An adaptive method enhances the edge detector by choosing pixels of the bilinear interpolation. Furthermore, an algebraic manipulation and a technique of hardware sharing are utilized to optimize bilinear interpolation, which reduce the computation complexity and chip area. This algorithm uses eight 8-bit registers and it can process streaming data directly and needs a one-line-buffer memory. The hardware of this work is analyzed by using Altera Cyclone II FPGA. The FPGA implementation of this method utilizes 6.65 K gates, but the other algorithms utilize a large number of gates. For example, winscale [10], edge-oriented [15], and an adaptive [3] algorithms utilize 17.4 K, 10.4 K, and 9.27 K gates, respectively. The VLSI circuit of this method contains 6.67 K gate count and operates at 280 MHz by utilizing TSMC 0.13 μm CMOS technology. This method achieves higher PSNR as 42.08 by using sharp spatial filter and 41.91 by using bilinear interpolation for the test image (splash) for image scaling down from the size of

Table 1: Comparison of computing resources.

Interpolation	Multiplication (numbers)	Addition (numbers)
Digital image scaling [5]	13	20
Winscale [9]	10	11
VLSI design of Winscale [10]	9	12
Bi-cubic [11]	32	53
Real time FPGA [12]	32	53
Extended linear [13]	4	3
EAELI [14]	8	12
Edge-oriented [15]	13	14
NICRMA [7]	11	14
HABI [16]	20	N/A
Adaptive bilinear [3]	3	50
SL Chen's image scaling [17]	4	36
Edge-enhanced adaptive scalar [18]	3	19

512 × 512 to 720 × 480. Based on PSNR, this method can give better image quality than other methods such as bilinear, winscale [6], bi-cubic [5], adaptive [3], and edge-enhanced [15] algorithms.

4. Performance Analysis

The performance analysis can be given in terms of parameters such as line buffer, gate count, frequency, and power. Figures 4, 5, and 6 and Tables 1, 2, 3, and 4 show the comparison of those parameters for various interpolation algorithms using VLSI architectures. These architectures are implemented by using different FPGA devices or by using TSMC 0.18 μm/TSMC 0.13 μm CMOS libraries for different image processing purposes.

Table 1 describes that the number of arithmetic elements such as multiplication and addition are used for various interpolation techniques. Generally, the number of arithmetic element is a major factor that affects the computational complexity of VLSI circuits. Table 1 demonstrates two interpolation algorithms such as bi-cubic [11] and real-time FPGA [12] that require a large number of multiplications and additions as 32 and 53, respectively. Also, it shows that the other two interpolation algorithms such as edge-enhanced adaptive scalar [18] and an adaptive bilinear [3] require less number of multiplications as three but the adaptive linear requires a large number of addition as 50. Furthermore, it describes that the extended linear [13] requires less number of multiplications as four and number of additions are only three. Therefore, the computation complexity of extended linear interpolation [13] is very less as compared with other interpolation algorithms.

Table 2 gives the comparison of different interpolation architectures in terms of number of memory buffer, gate count, frequency, and number of configurable logic blocks. The different types of FPGA used for designing these interpolation schemes are Virtex II, Virtex IV, Quartus II, and Spartan II devices. Among the various FPGA devices,

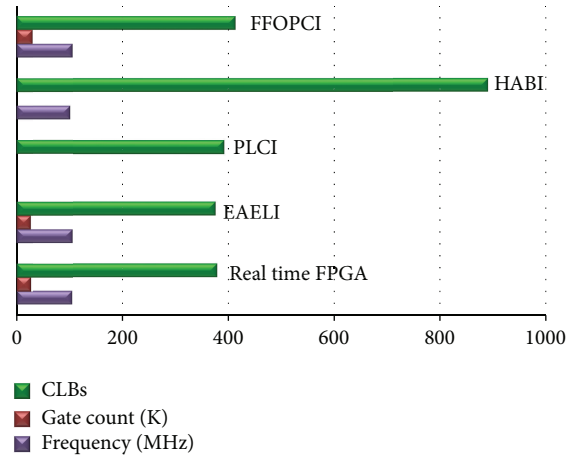

Figure 4: CLBs, gate count, and frequency of FPGA (Virtex-II) based interpolation methods.

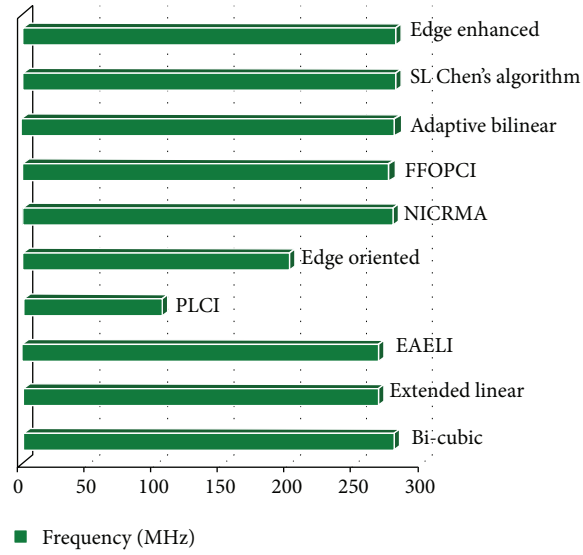

Figure 5: Frequency of TSMC 0.13 μm based interpolation methods.

Xilinx Virtex-II is mostly used. By comparing Virtex-II based interpolation methods, the HABI [16] system requires a large number of CLBs as 890. But the EAELI [14] requires a less number of CLBs as 376 and the gate count of this method is only 25.98 K. Also, based on Table 1, EAELI [14] requires a less number of multiplications as eight and a less number of additions as 12 as compared with HABI [16]. Therefore, among the various implementations on Virtex-II, the EAELI [14] is an area efficient method. Further, by comparing adaptive bilinear [3] and edge-enhanced adaptive scalar [18], the work in [18] uses less number of line buffer memory as only one and it requires less gate count as 6.65 K. But, [3] requires four-line buffer memory and 9.28 K. Therefore, [18] is an area efficient implementation on Quartus-II FPGA.

Figure 4 tells that the EAELI [14] uses less number of CLBs than other methods [12, 16, 19, 20]. Further, it illustrates that the operating frequency is almost similar for all the methods except PLCI [20].

Table 2: Comparison of FPGA based interpolation method.

Interpolation algorithms	FPGA	Parameters			
		Memory buffer (numbers)	Gate count (K)	Frequency (MHz)	CLBs (numbers)
Winscale [9]	FPGA	4 line	29	65	N/A
Bi-cubic [11]	FPGA	4 line	N/A	N/A	437
HABI [16]	Virtex-II	N/A	N/A	100	890
FFOPCI [19]	Virtex-II	N/A	28.9	104.3	414
PLCI [20]	Virtex-II	N/A	N/A	N/A	393
Real time FPGA [12]	Virtex-II	4 line	26.2	104	379
EAELI [14]	Virtex-II	N/A	25.98	104.3	376
Parallel bilinear [21]	Virtex-IV	N/A	N/A	250	362
Adaptive bilinear [3]	Quartus-II	4 line	9.28	67	N/A
Edge-enhanced adaptive scalar [18]	Quartus-II	1 line	6.65	72.57	N/A
Digital image scaling [5]	Quartus-II	4 line	N/A	55	N/A
Bilinear [22]	Spartan-II	N/A	N/A	N/A	329

Table 3: Comparison of TSMC 0.18 μm based interpolation methods.

Interpolation algorithms	Parameters				
	Memory buffer (numbers)	Gate count (K)	Frequency (MHz)	Throughput (pixels/second) Frames/sec	Power (mW)
NICRMA [7]	4 line	12.9	200	N/A	11.69
Adaptive bilinear [3]	4 line	9.27	200	200 M/30 f/sec	14.7
SL Chen's image scaling [17]	1 line	6.08	200	200 M/30 f/sec	6.45
Edge-enhanced adaptive scalar [18]	1 line	6.65	200	200 M/30 f/sec	10.23

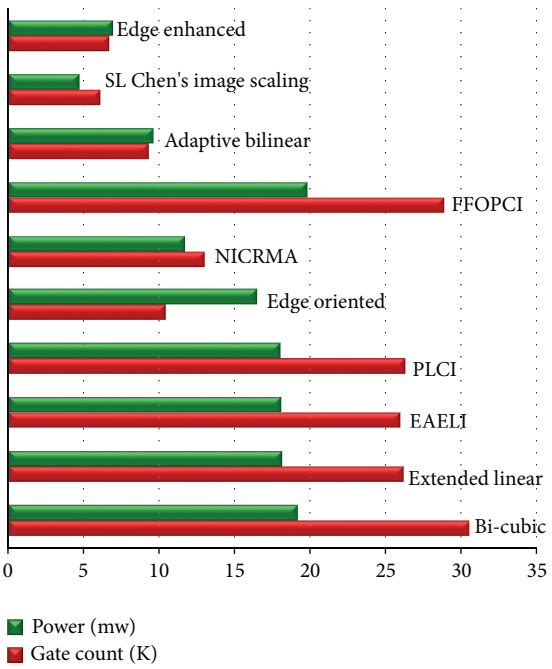

Figure 6: Power and gate count of TSMC 0.13 μm based interpolation methods.

Figure 4 also shows the comparison of various interpolation methods such as FFOPCI [19], HABI [16], PLCI [20], EAELI [14], and real-time FPGA [12] in terms of their number of CLBs used, gate count, and frequency on Virtex-II based implementation.

Table 3 illustrates the different VLSI architectures of image interpolation in terms of line-buffer memory, gate count, throughput, and power consumption on TSMC 0.18 μm technology. Based on Tables 1 and 3, as adaptive bilinear [3] requires three multiplications and a large number of additions as 50, this method utilizes 9.27 K gate count with four-line-memory buffer on TSMC 0.18 μm technology. But, the SL Chen's image scaling [17] consumes a low power as 6.45 mW and this method requires only a less amount of gate count as 6.08 K with only one-line-memory buffer and four multiplications and 36 additions. Table 3 also illustrates the interpolation methods such as adaptive bilinear [3], SL Chen's image scaling [17], and edge-enhanced adaptive scalar [18] provide the same throughput as 200 Mega pixels per second to operate a HDMI video resolution of wide quad super extended graphics array (WQSXGA) (3200 × 2048) at 30 frames per second in real-time multimedia image processing.

Table 4 lists the different parameters like memory buffer, gate count, frequency, throughput, and power of different

Table 4: Comparison of TSMC 0.13 μm based interpolation methods.

Interpolation algorithms	Parameters				
	Memory buffer (numbers)	Gate count (K)	Frequency (MHz)	Throughput (pixels/sec) Frames/sec	Power (mW)
FFOPCI [19]	NA	28.9	275	NA	19.80
Bi-cubic [11]	4 line	30.6	279	NA	19.17
Extended linear [13]	4 line	26.2	267	NA	18.14
EAELI [14]	NA	25.98	267	NA	18.07
PLCI [20]	NA	26.3	104.3	NA	18.02
Edge-oriented [15]	1 line	10.4	200	NA	16.47
NICRMA [7]	4 line	13	278	278 M/30 f/sec	11.69
Adaptive bilinear [3]	4 line	9.28	280	280 M/30 f/sec	9.6
Edge-enhanced adaptive scalar [18]	1 line	6.67	280	280 M/30 f/sec	6.9
SL Chen's image scaling [17]	1 line	6.08	280	280 M/30 f/sec	4.68

image interpolations on TSMC 0.13 μm VLSI technology. Based on Tables 1 and 4, the SL Chen's image scaling [17] requires a less number of arithmetic elements and therefore this method utilizes less power as 4.68 mW and less gate count as 6.08 K on TSMC 0.13 μm technology with only one-line-memory buffer. But, the bi-cubic algorithm [11] uses a large number of arithmetic elements (32 multiplications and 53 additions) and therefore this method requires a large amount of gate as 30.6 K and consumes a large amount of power as 19.17 mW with four-line-memory buffer. Therefore, the SL Chen's image scaling [17] can be implemented as an area efficient interpolation. By comparing the throughput of different interpolation methods on TSMC 0.13 μm technology, the methods such as adaptive bi-linear [3], SL Chen's image scaling [17], and edge-enhanced adaptive scalar [18] achieve the 280 Mega pixels per second or 30 frames per second. But the method NICRMA [7] achieves only 27 Mega pixels per second with four-line-memory buffer.

Figure 5 compares the frequency of various interpolation methods on TSMC 0.13 μm technology. This shows that except for PLCI [20] and edge oriented [15], all other methods operate on similar range of frequency between 260 MHz and 280 MHz.

Figure 6 demonstrates power utilization and gate count of various interpolation methods on TSMC 0.13 μm technologies. This shows the various interpolation methods such as PLCI [20], EAELI [14]. Extended-linear [13] and bi-cubic architecture [11] utilize a power ranges between 18 mW and 20 mW. Also, they require a large gate count ranges between 25 K and 31 K. The methods such as NICFMA [7] and edge oriented [15] utilize a moderate range of power and gates. But, the methods such as edge enhanced [18], SL Chen's image scaling [17], and adaptive bilinear [3] utilize a less amount of power ranges between four mW and 10 mW. Also, they require a less amount of gate ranges between 6 K and 10 K.

5. Conclusion

Digital image interpolation algorithms are widely used in many fields of digital image and video applications. These applications require improved image quality and high processing performance of hardware requirements. This survey focuses on different interpolation algorithms and their hardware performance. The algorithms such as Adaptive bilinear, SL Chen's image scaling algorithm, Extended linear interpolation algorithm provide low cost, low power consumption, low memory demand, and high performance. Among the various interpolation algorithms, SL Chen's image scaling consumes less power as 4.68 mW and requires a less gate count as 6.08 K with higher operating frequency of 280 MHz and one-line-memory buffer by TSMC 0.13 μm VLSI technologies, whereas the NICRMA algorithm which requires moderate power of 11.69 mW contains 13 K gate counts, yields a processing rate of approximately 278 MHz using TSMC-0.13 μm technology, and provides better image quality with four-line-memory buffer. Based on the survey, it can be concluded that the performance of hardware architecture of image interpolation can be improved by reducing number line buffer memory and by removing redundant arithmetic elements on generating weighting coefficient.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] I. Pekkucuksen and Y. Altunbasak, "Multiscale gradients-based colour filter array interpolation," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 157–165, 2013.

[2] T. M. Lehmann, C. Gönner, and K. Spitzer, "Survey: interpolation methods in medical image processing," *IEEE Transactions on Medical Imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[3] S.-L. Chen, H.-Y. Huang, and C.-H. Luo, "A low-cost high-quality adaptive scalar for real-time multimedia applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 11, pp. 1600–1611, 2011.

[4] A. Giachetti and N. Asuni, "Real-time artifact-free image upscaling," *IEEE Transactions on Image Processing*, vol. 20, no. 10, pp. 2760–2768, 2011.

[5] I. Andreadis and A. Amanatiadis, "Digital image scaling," in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference (IMTC '05)*, pp. 2028–2032, Ottawa, Canada, May 2005.

[6] S. P. Jaiswal, V. Jakhetiya, A. Kumar, and A. K. Tiwari, "A low complex context adaptive image interpolation algorithm for real time applications," in *Proceedings of the IEEE International Conference on Instrumentation and Measurement Technology (MTC '12)*, pp. 969–972, 2012.

[7] C.-C. Huang, P.-Y. Chen, and C.-H. Ma, "A novel interpolation chip for real-time multimedia applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 10, pp. 1512–1525, 2012.

[8] L. Deng, K. Sobti, Y. Zhang, and C. Chakrabarti, "Accurate area, time and power models for FPGA-based implementations," *Journal of Signal Processing Systems*, vol. 63, no. 1, pp. 39–50, 2011.

[9] 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, no. 6, pp. 549–553, 2003.

[10] C.-C. Lin, Z.-C. Wu, W.-K. Tsai, M.-H. Sheu, and H.-K. Chiang, "The VLSI design of winscale for digital image scaling," in *Proceedings of the 3rd International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP '07)*, vol. 2, pp. 511–514, November 2007.

[11] C.-C. Lin, M.-H. Sheu, H.-K. Chiang, C. Liaw, and Z.-C. Wu, "The efficient VLSI design of BI-CUBIC convolution interpolation for digital image processing," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 480–483, May 2008.

[12] C.-C. Lin, M.-H. Sheu, H.-K. Chiang, W.-K. Tsai, and Z.-C. Wu, "Real-time FPGA architecture of extended linear convolution for digital image scaling," in *Proceedings of the International Conference on Field-Programmable Technology (FPT '08)*, pp. 381–384, Taipei, Taiwan, December 2008.

[13] 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 *Proceedings of the International Conference on Embedded Software and Systems (ICESS '08)*, pp. 196–202, July 2008.

[14] C.-C. Lin, M.-H. Sheu, H.-K. Chiang, C. Liaw, Z.-C. Wu, and W.-K. Tsai, "An efficient architecture of extended linear interpolation for image processing," *Journal of Information Science and Engineering*, vol. 26, no. 2, pp. 631–648, 2010.

[15] P.-Y. Chen, C.-Y. Lien, and C.-P. Lu, "VLSI implementation of an edge-oriented image scaling processor," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 9, pp. 1275–1284, 2009.

[16] M. A. Nuño-Maganda and M. O. Arias-Estrada, "Real-time FPGA-based architecture for bicubic interpolation: an application for digital image scaling," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '05)*, pp. 1–8, September 2005.

[17] S.-L. Chen, "VLSI implementation of a low cost high quality image scaling processer," *IEEE Transactions on Circuit and System: Express Briefs*, vol. 60, no. 1, pp. 31–35, 2013.

[18] S.-L. 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, pp. 1510–1522, 2013.

[19] C.-C. Lin, M.-H. Sheu, C. Liaw, and H.-K. Chiang, "Fast first-order polynomials convolution interpolation for real-time digital image reconstruction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 9, pp. 1260–1264, 2010.

[20] C.-C. Lin, C. Liaw, and C.-T. Tsai, "A piecewise linear convolution interpolation with third-order approximation for real-time image processing," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '10)*, pp. 3632–3637, October 2010.

[21] S. A. Fahmy, "Generalised parallel bilinear interpolation architecture for vision systems," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '08)*, pp. 331–336, December 2008.

[22] K. T. Gribbon and D. G. Bailey, "A novel approach to real-time bilinear interpolation," in *Proceedings of the 2nd IEEE International Workshop on Electronic Design, Test and Applications (DELTA '04)*, pp. 126–131, January 2004.