

# VLSI Architecture for Image Scaling Based On the Edge Adaptive Area Pixel Model

**Jitendra Jugele**

ECE ME (VLSI)

SSCET, Bhilai, India

[jitendra.jugele19@gmail.com](mailto:jitendra.jugele19@gmail.com)

**Chandrabas Sahu**

Sr. Asst. prof. (E&I DEPT)

SSCET, Bhilai, India

[chandrahasscet@gmail.com](mailto:chandrahasscet@gmail.com)

**Abstract**—Image scaling is a very important technique and has been widely used in many image processing applications. In this paper, we present an edge-adaptive area-pixel scaling model. To achieve the goal of low cost, the area-pixel scaling technique is implemented with a low-complexity VLSI architecture in our design. A simple edge catching technique is adopted to preserve the image edge features effectively so as to achieve better image quality. Compared with the previous low-complexity techniques, our method performs better in terms of both quantitative evaluation and visual quality. The seven-stage VLSI architecture of our image scaling processor contains 10.4-K gate counts and yield a processing rate of about 200 MHz by using TSMC 0.18- $\mu$ m technology.

**Keywords:** Image scaling, edge catching, pipeline architecture, VLSI.

## I. INTRODUCTION

IMAGE scaling is widely used in many fields [1]–[4], ranging from consumer electronics to medical imaging. It is indispensable when the resolution of an image generated by a source device is different from the screen resolution of a target display. For example, we have to enlarge images to fit HDTV or to scale them down to fit the mini-size portable LCD panel. The most simple and widely used scaling methods are the nearest neighbor [5] and bilinear [6] techniques.

According to the required computations and memory space, we can divide the existing scaling methods [5] into two classes: lower complexity [5]–[8] and higher complexity [9]–[10] scaling techniques. The complexity of the former is very low and comparable to conventional bilinear method. The latter yields visually pleasing images by utilizing more advanced scaling methods. In many practical real-time applications, the scaling process is included in end-user equipment, so a good lower complexity scaling technique, which is simple and suitable for low-cost VLSI implementation, is needed. In this paper, we consider the lower complexity scaling techniques [5]–[8] only.

To achieve the goal of lower cost, we present an edge adaptive area-pixel scaling processor model in this paper. The area-pixel scaling technique is approximated and implemented with the proper and low-cost VLSI circuit in our design. The proposed scaling processor can support floating-point magnification factor and preserve the edge features efficiently by taking into account the local characteristic existed in those available source pixels around the target pixel. Furthermore, it handles streaming data directly and requires only small amount of memory: one line buffer rather than a full frame buffer. The experimental results demonstrate that the proposed design performs better than other lower

complexity image scaling methods [5]–[8] in terms of both quantitative evaluation and visual quality.

The seven-stage VLSI architecture for the proposed design was implemented and synthesized by using Verilog HDL and synopsys design compiler, respectively. In our simulation, the circuit can achieve 200 MHz with 10.4-K gate counts by using TSMC 0.18- m technology. Since it can process one pixel per clock cycle, it is quick enough to process a video resolution of WQXGA (3200 2048) at 30 f/s in real time.

## II AREA-PIXEL SCALING TECHNIQUE

### A. An Overview of Area-Pixel Scaling Technique.

Assume that the source image represents the original image to be scaled up/down and target image represents the scaled image. The area-pixel scaling technique performs scale-up scale-down transformation by using the area pixel model instead of the common point model. Each pixel is treated as one small rectangle but not a point; its intensity is evenly distributed in the rectangle area. Fig. 1 shows an example of image scaleup process of the area-pixel model where a source image of 4×4 pixels is scaled up to the target image of 5×5 pixels. Obviously, the area of a target pixel is less than that of a source pixel. A window is put on the current target pixel to calculate its estimated luminance value. As shown in Fig. 1(c), the number of source pixels overlapped by the current target pixel window is one, two, or a maximum of four. Let the luminance values of four source pixels overlapped by the window of current target pixel at coordinate (k,l) be

denoted as  $F_S(m,n)$ ,  $F_S(m+1,n)$  and  $F_S(m+1,n+1)$  respectively. The estimated value of current target pixel, denoted  $F_T((k,l))$  can be calculated by weighted averaging the luminance values of four source pixels with area coverage ratio as

$$F_T((k,l)) = \sum_{i=0}^1 \sum_{j=0}^1 [F_S(m+i,n+j) \times W(m+i,n+j)]$$

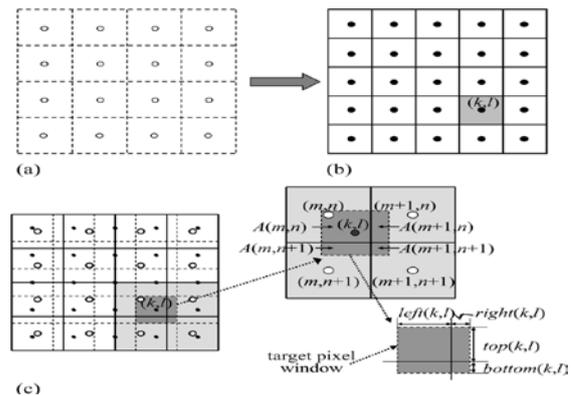


Fig 1 (a) source image of 4×4 pixels. (b) A target image of 5×5 pixels. (c) Relations of the target pixel and source pixels

## III THE PROPOSED LOW COMPLEXITY ALGORITHM

we know that the direct implementation of area-pixel scaling requires some extensive floating-point computations for the current target pixel at (k,l) to determine the four parameters,  $left'(k,l)$ ,  $top'(k,l)$ ,  $right'(k,l)$  and  $bottom'(k,l)$ . In the proposed processor, we use an approximate technique suitable for low-cost VLSI implementation to achieve that goal properly. We modify (3) and implement the calculation of areas of the overlapped regions as

$$[A'(m,n), A'(m+1,n), A'(m,n+1), A'(m+1,n+1)] = [left'(k,l) \times top'(k,l), right'(k,l) \times top'(k,l), left'(k,l) \times bottom'(k,l), right'(k,l) \times bottom'(k,l)].$$

Those  $left'(k,l), top'(k,l)$  ,  $right'(k,l)$  and  $bottom'(k,l)$  are all 6-b integer and given as  $[left'(k,l), top'(k,l)$  ,  $right'(k,l)$  ,  $bottom'(k,l)] = Appr [left'(k,l), top'(k,l)$  ,  $right'(k,l)$  ,  $bottom'(k,l)]$

Where Appr represents the approximate operator adopted in our design and will be explained in detail later. To obtain better visual quality, a simple low-cost edge catching technique is employed to preserve the edge features effectively by taking into account the local characteristic existed in those available source pixels around the target pixel. The final areas of the overlapped regions are given as

$$[A''(m,n), A''(m+1,n), A''(m,n+1), A''(m+1,n+1)] = \Gamma ([A'(m,n), A'(m+1,n), A'(m,n+1), A'(m+1,n+1)])$$

(6) where we adopt a  $\Gamma$  tuning operator to tune the areas of four overlapped regions according to the edge features obtained by our edge-catching technique. By applying (6) to (1) and (2),

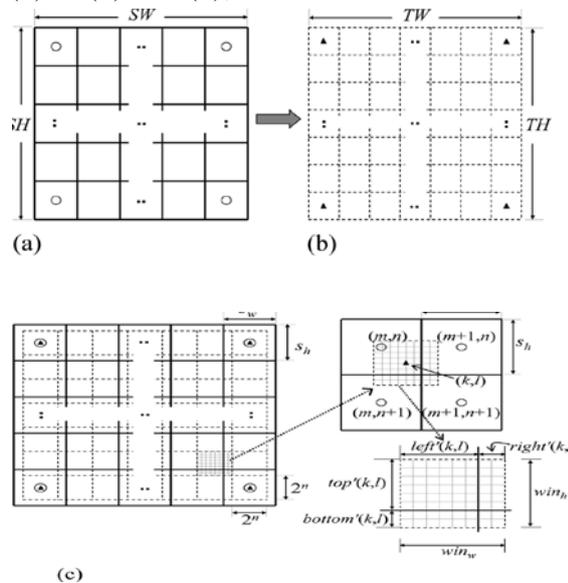


Fig 2. Example of image enlargement for our method (a) A source image of  $SW \times SH$  pixels (b) A target image of  $TW \times TH$

pixels (c) Relations of the target pixels and source pixels.

### The Approximate Technique

Fig. 2 shows an example of our image scale up process where a source image of  $SW \times SH$  pixels is scaled up to the target image of  $TW \times TH$  pixels and every pixel is treated as one rectangle. As shown in Fig. 2(c), we align those centers of four corner rectangles (pixels) in the source and target images. For simple hardware implementation, each rectangular target pixel is treated as  $2^n \times 2^n$  grids with uniform size ( $n$  is 3 for Fig. 2). Assume that the width and the height of the target pixel window are denoted as  $win_w$  and  $win_h$ , then the area of the current target pixel window ( $A_{sum}$ ) can be calculated as  $win_w$

### B. The Low-Cost Edge-Catching Technique

In the design, we take the sigmoidal signal [15] as the image edge model for image scaling. Fig. 3(a) shows an example of the 1-D sigmoidal signal. Assume that the pixel to be interpolated is located at coordinate  $k$  and its nearest available neighbors in the image are located at coordinate  $m$  for the left and  $m+1$  for the right. Let  $s = k - m$  and  $E(k)$  represent the luminance value of the pixel at coordinate  $k$ . If  $t$  he estimated value  $E(k)$  of the pixel to be interpolated is determined by using linear interpolation, it can be calculated as  $E(k) = (1-s) \times E(m) + s \times E(m+1)$

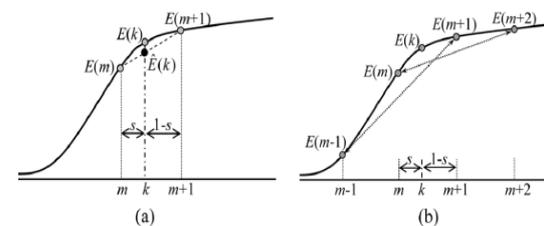


Fig 3 Local characteristic of the data in the neighbourhood of  $k$ . (a) An image edge model (b) Local  $c/s$

#### IV VLSI ARCHITECTURE

Our scaling method requires low computational complexity and only one line memory buffer, so it is suitable for low-cost VLSI implementation. Fig. 4 shows block diagram of the seven stage VLSI architecture for our scaling method. The architecture consists of seven main blocks: approximate module (AM), register bank (RB), area generator (AG), edge catcher (EC), area tuner (AT), target generator (TG), and the controller. Each of them is described briefly in the following subsections.

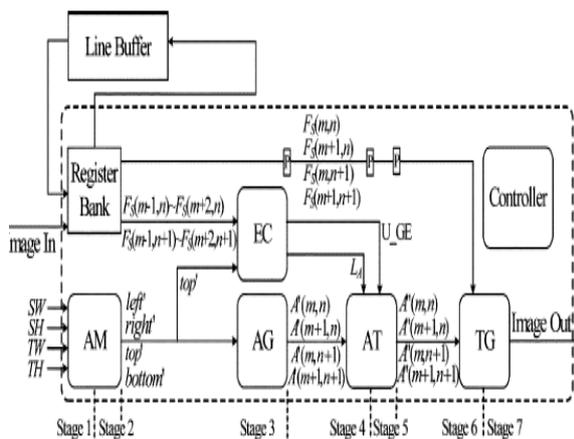


Fig 4 Block diagram of VLSI architecture for our scaling methods

A. AM is composed of two-stage pipelined architecture. In the first stage, the coordinate  $(k,l)$  of the current target pixel and the coordinate  $(m,n)$  of the top-left source pixel overlapped by the current window are determined. In the second stage, AM first calculates  $win_{left}(k,l)$ ,  $src_{right}(m,n)$ ,  $win_{top}(k,l)$  and  $src_{btm}(m,n)$  according to (10)–(11) and (13)–(14), and then generates  $left'(k,l)$ ,  $right'(k,l)$ ,  $top'(k,l)$

, and  $bottom'(k,l)$  according to (7)–(9) and (12).

#### B. Register Bank

In our design, the estimated value of the current target pixel  $F_T(k,l)$  is calculated by using the luminance values of  $2 \times 4$  neighboring source pixels  $F_S(m-1,n)$ ,  $F_S(m,n)$ ,  $F_S(m+1,n)$ ,  $F_S(m+2,n)$ ,  $F_S(m-1,n+1)$ ,  $F_S(m,n+1)$ ,  $F_S(m+1,n+1)$ , and  $F_S(m+2,n+1)$ . The register bank, consisting of eight registers, is used to provide those source luminance values at exact time for the estimated process of current target pixel. Fig. 5 shows the internal connections of RB where every four registers are connected serially in a chain to provide four pixel values of a row in current pixel window, and the line buffer is used to store the pixel values of one row in the source image. When the controller enables the shift operation in RB, two new values are read into RB (Reg3 and Reg7) and the rest 6-pixel values are shifted to their right registers one by one. The 8-pixel values stored in RB will be used by EC for edge catching and by TG for target pixel estimating.

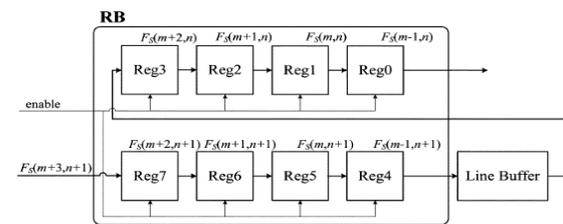


Fig 5 Architecture of register bank

#### C. Area Generator

For each target pixel, AG calculates the areas of the overlapped regions  $A'(m,n)$ ,  $A'(m,n+1)$ ,  $A'(m+1,n)$  and  $A'(m+1,n+1)$  according to (4). Fig. 11 shows the architecture of AG where represents the pipeline register and MULT is the  $4 \times 4$  integer multiplier

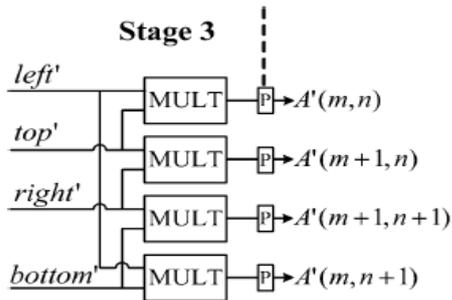


Fig 6 Architecture of area generator.

#### D. Edge Catcher

EC implements the proposed low-cost edge-catching technique and outputs the evaluating parameter  $L_A$ , which represents the local edge characteristic of current pixel at coordinate  $(k,l)$ . Fig. 12 shows the architecture of EC where SUB unit generates the difference of two inputs and  $|SUB|$  unit generates the two inputs' absolute value of difference. The comparator CMP outputs logic 1 if the input value is greater than or equal to  $win_h/2$ . The binary compared result, denoted as  $U_{GE}$ , is used to decide whether the upper row (row  $n$ ) in current pixel window is more important than the lower row (row  $n+1$ ) in regards to catch edge features. According to (22) and (24), EC produces the final result  $L_A$  and sends it to the following AT.

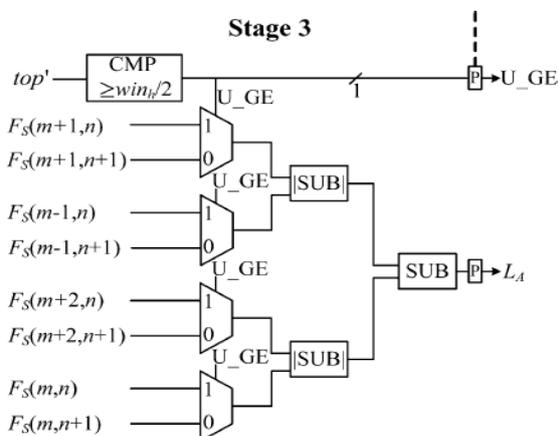


Fig 7 Architecture of edge catcher.

#### E. Area Tuner

AT is used to modify the areas of the four overlapped regions based on the current local edge information ( $L_A$  and  $U_{GE}$  provided by EC). Fig. 13 shows the two-stage pipeline architecture of AT. If  $U_{GE}$  is equal to 1, the upper row (row  $n$ ) in current pixel window is more important, so  $A'(m,n)$  and  $A'(m+1,n)$  are modified according to (23). On the contrary, if  $U_{GE}$  is equal to 0, the lower row (row  $n+1$ ) is more important, so  $A'(m,n+1)$  and  $A'(m+1,n+1)$  are modified according to (25). Finally, the tuned areas  $A''(m,n)$ ,  $A''(m+1,n)$ ,  $A''(m,n+1)$  and  $A''(m+1,n+1)$  are sent to TG.

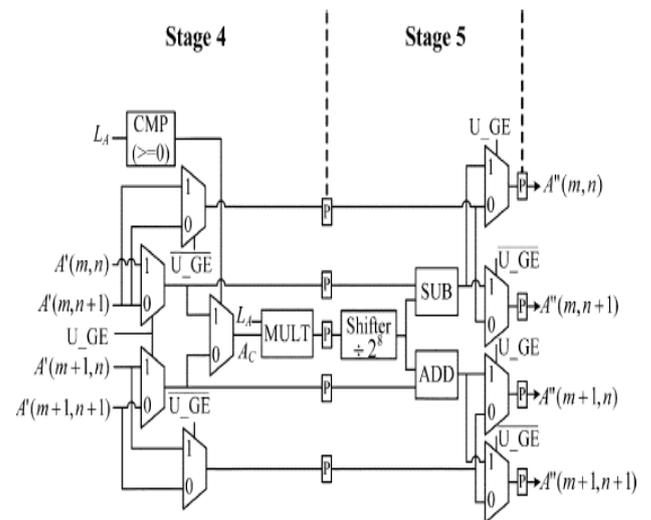


Fig 8 Architecture of area tuner

#### F. Target Generator

By weighted averaging the luminance values of four source pixels with tuned-area coverage ratio, TG implements (1) and (2) to determine the estimated value  $F_T(k,l)$ . Fig. 9 shows the two-stage pipeline architecture of TG. Four MULT units and three ADD units are used to perform (1). Since the value of  $A_{sum}$  is equal to the power of 2, the division operation in (2) can be implemented by the shifter easily.

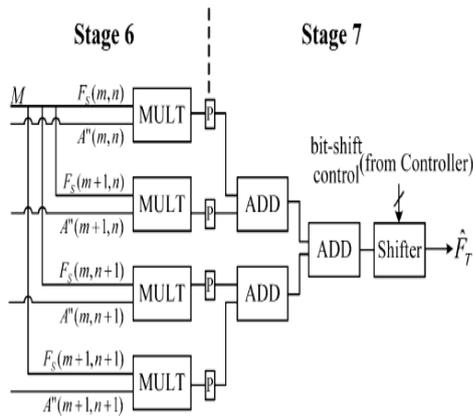


Fig 9 Architecture of target generator .

### G. Controller

The controller, realized with a finite-state machine, monitors the data flow and sends proper control signals to all other components. In the design, AM, AT, and TG require two clock cycles to complete their functions, respectively. Both AG and EC need one clock cycle to finish their tasks, and they work in parallel because no data dependency between them exists. For each target pixel, seven clock cycles are needed to output the estimated value  $F_T(k,l)$ .

## V SIMULATION RESULTS AND CHIP IMPLEMENTATION

To evaluate the performance of our image-scaling algorithm, we use 12 gray-scale test images of 512x512 8 b. For each single test image, we reduce/enlarge the original image by using the well-known bilinear method, and then employ various approaches to scale up/down the bilinear-scaled image back to the size of the original test image. Thus, we can compare the image quality of the reconstructed images for various scaling methods. Three well-known scaling methods, nearest neighbor (NN) [5], bilinear (BL) [6], and bicubic (BC) [9], two area-pixel scaling

methods, Win (winscale in [7]) and M Win (the modified winscale in [8]), and our method are used for comparison in terms of computational complexity, objectivetesting (quantitative evaluation), and subjective testing (visual quality), respectively. To reduce hardware cost, we adopt the low-cost technique suitable for VLSI implementation to perform area-pixel scaling. To verify the computational complexity, all the six scaling methods are implemented in C language on the 2.8-GHz Pentium 4 processor with 512-MB memory and the 520-MHz INTEL XScale PXA270 with 64-MB memory, respectively. To explore the performance of quantitative evaluation for image enlargement and reduction, first we scale the twelve 512 512 test images to the size of 400 400, 600 600, and 256 256, respectively, by using the bilinear method. Then, we scale up/down these images back to the size of 512 512. Here, the output images of our scaling method are generated by the proposed VLSI circuit after post-layout transistor-level simulation. The output images of other scaling methods are all generated with software C programs. Simulation results show that our design achieves better quantitative quality than the previous low-complexity scaling methods [5]–[8]. However, the exact degree of improvement is dependent on the content of different images processed.

The proposed VLSI architecture of the proposed design was implemented by using Verilog HDL. We used SYNOPSIS Design Vision to synthesize the design with TSMC's 0.18- m cell library.

## VI. CONCLUSION

A low-cost image scaling processor model is proposed in this paper. The experimental results demonstrate that our design achieves better performances in both objective and subjective image quality than other low-complexity scaling methods. Furthermore, an efficient VLSI architecture for the proposed method is presented. In our simulation, it operates with a clock period of 5 ns and achieves a processing rate of 200 megapixels/second. The architecture works with monochromatic images, but it can be extended for working with RGB color images easily.

## REFERENCES

- [1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1992.
- [2] W. K. Pratt, *Digital Image Processing*. New York: Wiley-Interscience, 1991.
- [3] T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: Interpolation methods in medical image processing," *IEEE Trans. Med. Imag.*, vol. 18, no. 11, pp. 1049–1075, Nov. 1999.
- [4] C. Weerasnghe, M. Nilsson, S. Lichman, and I. Kharitonenko, "Digital zoom camera with image sharpening and suppression," *IEEE Trans. Consumer Electron.*, vol. 50, no. 3, pp. 777–786, Aug. 2004.
- [5] S. Fifman, "Digital rectification of ERTS multispectral imagery," in *Proc. Significant Results Obtained from Earth Resources Technology Satellite-1*, 1973, vol. 1, pp. 1131–1142.
- [6] J. A. Parker, R. V. Kenyon, and D. E. Troxel, "Comparison of interpolation methods for image resampling," *IEEE Trans. Med. Imag.*, vol. MI-2, no. 3, pp. 31–39, Sep. 1983.
- [7] C. Kim, S. M. Seong, J. A. Lee, and L. S. Kim, "Winscale: An image scaling algorithm using an area pixel model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 549–553, Jun. 2003.
- [8] I. Andreadis and A. Amanatiadis, "Digital image scaling," in *Proc. IEEE Instrum. Meas. Technol. Conf.*, May 2005, vol. 3, pp. 2028–2032.
- [9] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-26, no. 6, pp. 508–517, Dec. 1978.
- [10] J. K. Han and S. U. Baek, "Parametric cubic convolution scalar for enlargement and reduction of image," *IEEE Trans. Consumer Electron.*, vol. 46, no. 2, pp. 247–256, May 2000.