



Design Of High Performance And Low Power Vlsi Architecture For Image Scaling

¹P.Kranthi kumari, ²M.V.S.Abhinaya Reddy, ³K.Bhanu Prakash, ⁴Y.Chakrapani

¹B.Tech, ²B.Tech, ³B.Tech, ⁴B.Tech.

¹Electronics and Communication Engineering,

¹Gudlavaluru Engineering College, Gudlavaluru, India

Abstract: In multimedia systems, high-performance image scaling is essential to provide high-quality images on various devices. This paper introduces a new VLSI architecture for high-performance image scaling. The architecture utilizes optimized bilinear interpolation algorithms and hardware parallelism to improve throughput and minimize latency to provide fast and correct image scaling. Key aspects of the design are a pipelined architecture for ongoing data processing, bilinear interpolation methods to ensure image quality, and minimal power usage to maximize device battery life. The VLSI design compares favourably with previous methods in speed, resource usage, and power consumption. The architecture is ideal for inclusion within present-day multimedia systems and provides an extensible solution that addresses the needs of modern visual processing applications.

The suggested architecture is developed using Verilog HDL and implemented using MATLAB AND Xilinx Vivado software.

Index Terms – Image Scaling , Bilinear Interpolation.

I. INTRODUCTION

Image scaling is the process of resizing an image from one resolution to another, commonly used in digital image processing for applications such as medical imaging, video processing, and image zooming. With the increasing demand for high-quality images across various multimedia applications, efficient image scaling has become a crucial aspect of modern visual processing. Image scaling is widely used in display systems, video streaming, medical imaging, and mobile devices to adapt image resolutions while preserving visual quality. However, traditional software-based scaling techniques often suffer from high computational complexity and latency, making them unsuitable for real-time applications. To address these challenges, this paper presents a high-performance and low-power **VLSI (Very Large-Scale Integration) architecture** for image scaling. The proposed design leverages **hardware parallelism, pipelined processing, and adaptive interpolation techniques** to achieve faster computation while maintaining image quality. Unlike conventional methods that rely on general-purpose processors or DSPs (Digital Signal Processors), the dedicated VLSI implementation significantly improves efficiency, reduces power consumption, and enhances scalability for modern multimedia systems.

The key contributions of this work include:

- A **pipelined processing architecture** that ensures continuous data throughput with minimal latency.
- **Optimized interpolation techniques** to enhance image quality while reducing computational complexity.
- **Efficient resource utilization** to minimize area overhead and power consumption, making it suitable for battery-operated devices.
- **Implementation in Verilog HDL**, verified using MATLAB, and synthesized on Xilinx ISE to evaluate performance in terms of speed, power, and resource usage.

This paper compares the proposed VLSI design with existing methods to demonstrate its superiority in terms of **processing speed, power efficiency, and hardware utilization**. The architecture is well-suited for real-time multimedia applications, offering a scalable and efficient solution for modern visual processing needs.

II. PROPOSED IMAGE SCLING BLOCK DIAGRAM

Image scaling is a very important field of study, particularly in consideration of the need for real-time processing and power-efficient operation in embedded systems, mobile phones, and other computationally demanding applications. Such designs have a methodology that takes many steps, which guarantee that the system is capable of performing image processing operations such as scaling with minimal power expenditure and maximum throughput.

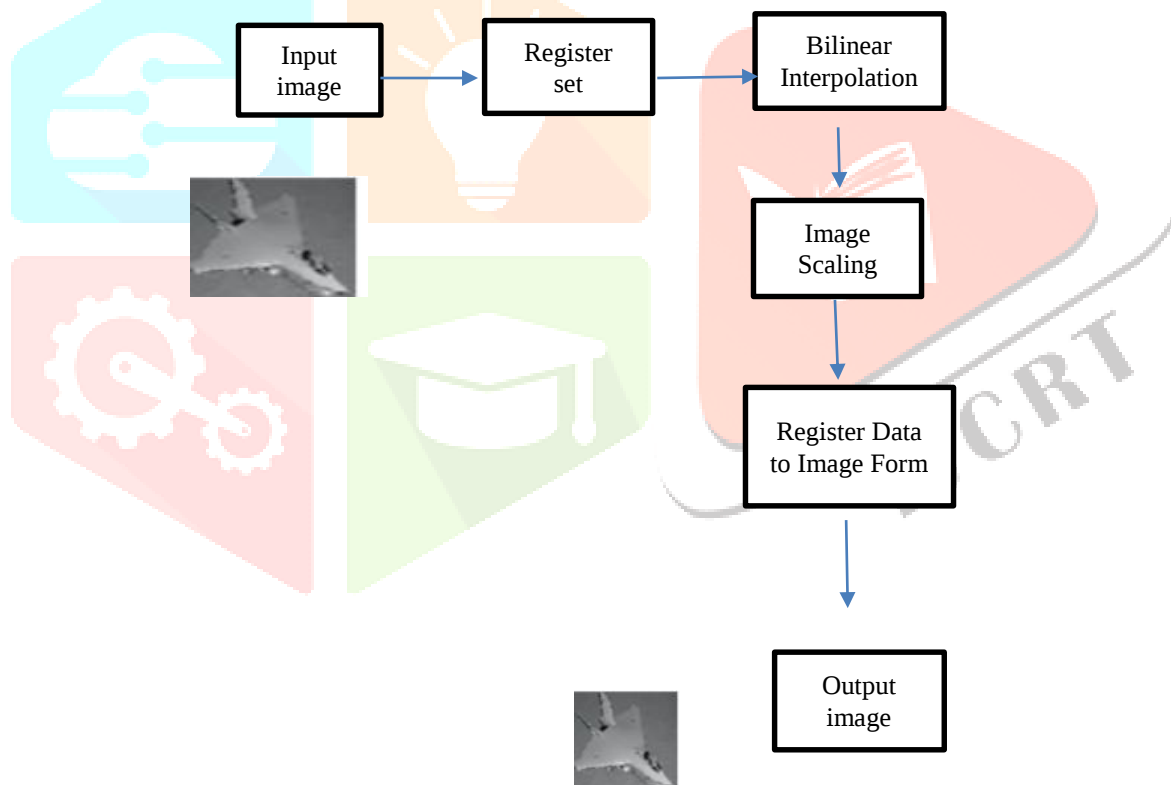


Fig.1 Block Diagram of Image Scaling

Figure 1. shows the Block diagram of proposed image scaling architecture. It consists of Register Set, Nearest Neighbour Interpolation, Image scaling. The following is the explanation of the various blocks and their functions:

1. Input Image

The operation begins with an input image, which is input to the system to be scaled. The input image may be in memory and be processed pixel by pixel.

2.Register Set

The "Register Set" in block diagram is a temporary storage device that stores pixel information during the image processing phases. Its primary operations are:

1. **Data Storage:** The register set stores pixel values from the input image prior to processing by the bilinear interpolation unit.
2. **Synchronization:** It makes data available in a structured format, avoiding loss or misalignment during processing.
3. **Effective Data Transfer:** The register set serves as a buffer to facilitate smooth data transfer between various stages of processing, including bilinear interpolation and image scaling.
4. **Optimizing Speed:** By temporarily storing data, the register set increases processing speed and reduces latency, enhancing the overall effectiveness of image scaling.

This phase is crucial for ensuring data integrity and proper interpolation before creating the scaled image.

3. Bilinear Interpolation

The "Bilinear Interpolation" block in the figure is an important block in image scaling, which interpolates the intensity of new pixels from the neighbouring pixels in the original image. This includes:

1. **Choosing Neighbouring Pixels:** For every new pixel to be interpolated, bilinear interpolation examines the four closest pixels from the original image (top-left, top-right, bottom-left, bottom-right).
2. **One-Dimensional Linear Interpolation:** The algorithm starts by performing linear interpolation in the x-direction to find intermediate values.
3. **Linear Interpolation in the Second Direction:** Then it interpolates in the y-direction based on the intermediate values calculated above.
4. **New Pixel Value Assignment:** The new pixel's final intensity value is determined by the weighted average of the four pixels around it.

This approach guarantees that the scaled image appears smoother with fewer pixelations and is therefore more aesthetically pleasing than using less complex techniques such as nearest-neighbour interpolation.

4. Image Scaling

After performing bilinear interpolation to scale an image, the output is a scaled image with smoother edges and less pixelation than the use of a basic method such as nearest-neighbour interpolation. Each new scaled pixel is calculated as a weighted average of the four closest original image pixels.

This guarantees that edges and textures look smoother.

Artifacts such as blackness or jagged edges are minimized.

The image feels more natural, particularly when scaling up.

Nevertheless, though bilinear interpolation enhances quality, it can blur fine details, particularly in big upscaling. For greater-quality requirements, more sophisticated techniques such as bicubic interpolation or deep learning-based super-resolution could be used.

5. Register Data to image form

The "Register Data to Image Form" in the block diagram is the process wherein the processed pixel data in bilinear interpolation and scaling processes are transformed back to a structured form of an image. It encompasses:

1. **Storing Pixel Values:** Computed pixel intensity values are being stored in structured form with accurate spatial relations.
2. **Data Formatting:** The processed data is formatted in standard image forms (like BMP, JPEG, or PNG) for subsequent utilization.

3. Image Reconstruction: The data is organized in rows and columns to produce a visually understandable output image.

4. Handling Final Output: The formatted image data is displayed or stored in memory for subsequent processing.

This process ensures that the resized image is maintained in its quality and is available for visualization or additional processing.

6. Output image

The "Output Image" block is the final output after the image has gone through the bilinear interpolation and scaling phases. It is the scaled version of the input image, produced with smoother transitions and better quality.

Key Features of the Output Image:

1. Resized Representation: The output image is either enlarged or shrunk depending on the scaling needs while ensuring visual coherence.
2. Smooth Transitions Between Pixels: Bilinear interpolation makes intensity changes between pixels look natural and avoids jagged edges and pixilation.
3. Image of Processed Data: The resultant image is put together from processed pixel values and displayed or stored in a required format.
4. Ready for Use: Now, the resulting image can be saved, output, or put to further process in applications such as computer vision, graphics, or machine learning.

This block concludes the image scaling pipeline with a high-quality, correctly scaled version of the input image.

III. ARCHITECTURE DIAGRAM OF IMAGE SCALING

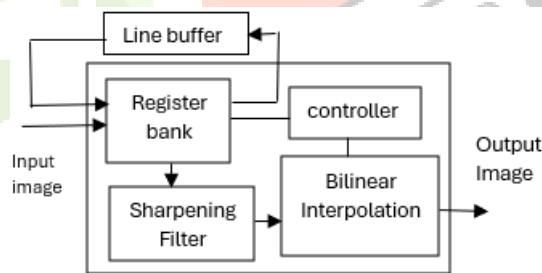


Fig.2 Proposed Architecture Diagram of Image Scaling

Register Bank

The Register Bank is an important part of image processing hardware, particularly in filtering and interpolation operations. It holds pixel data temporarily and provides easy access for sharpening and interpolation operations.

The Register Bank communicates between the Input Image and the Sharpening Filter. It supplies the pixel window required to the Sharpening Filter, and it tightens the image.

The filtered information is sent to the Bilinear Interpolation block in order to resize or smooth.

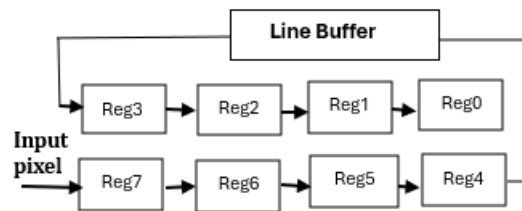


Fig.3 Architecture of Register Bank

The Register Bank consists of 8 registers (Reg 0 to Reg 7). Reg 0 through Reg 3 get data from the Line Buffer, which stores previous rows of pixels. Reg 4 through Reg 7 get data from the current input pixels directly. This arrangement works as efficiently to store a 3×3 pixel window in the Register Bank, which is required to perform filtering operations like convolution or detection of edges.

Sharpening Filter

To enhance images scaled with bilinear interpolation, a sharpening filter can be applied post-interpolation to reduce blurring and enhance edges, often combined with a clamp filter to mitigate artifacts.

Sharpening filters are useful for enhancing the outlines and details of images to add clarity and definition. It emphasizes the high-frequency parts of the image, specifically the regions where pixels change intensity rapidly, for example, edges and texture.

Controller

The controller generates the shift commands to control the shift registers in the register bank and to generate the scaling factors as inputs to the simplified bilinear interpolation. The controller was designed by the finite state machine (FSM) which contains registers and combinational circuits.

Bilinear Interpolation

Bilinear interpolation is performed using linear interpolation first in one direction, and then again in another direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location. Fig.4 shows the bilinear interpolation, target pixel $p(x, y)$ is obtained by linear interpolation $f(x, y_1)$ and $f(x, y_2)$ pixels in y-direction. $f(x, y_1)$ and $f(x, y_2)$ pixels are obtained by linear interpolation in x-direction.

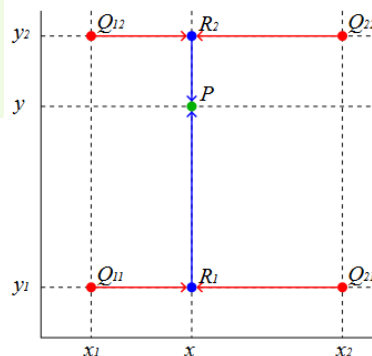


Fig 4. Bilinear Interpolation

$f(x, y_1)$ is obtained by linear interpolation of Q_{11} and Q_{21} .

$f(x, y_2)$ is obtained by linear interpolation of Q_{12} and Q_{22} .

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad (3.1)$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad (3.2)$$

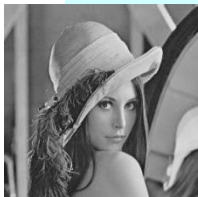
$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \quad (3.3)$$

$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \quad (3.4)$$

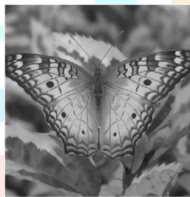
$$f(x, y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} [f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{21})(x - x_1)(y_2 - y) + f(Q_{22})(x - x_1)(y - y_1)] \quad (3.5)$$

$$f(x, y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} [x_2 - x \quad x - x_1] \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix} \quad (3.6)$$

IV. EXPERIMENTAL RESULTS



(a)



(b)



(c)



(d)



(e)

Fig 5. Input images (a) Lena of 256x256 resolution. (b) Butterfly of 256x256 resolution. (c) Sail Boat of 1920x1080 resolution. (d) Trees of 736x1104 resolution. (e) Taj mahal of 256x256 resolution.

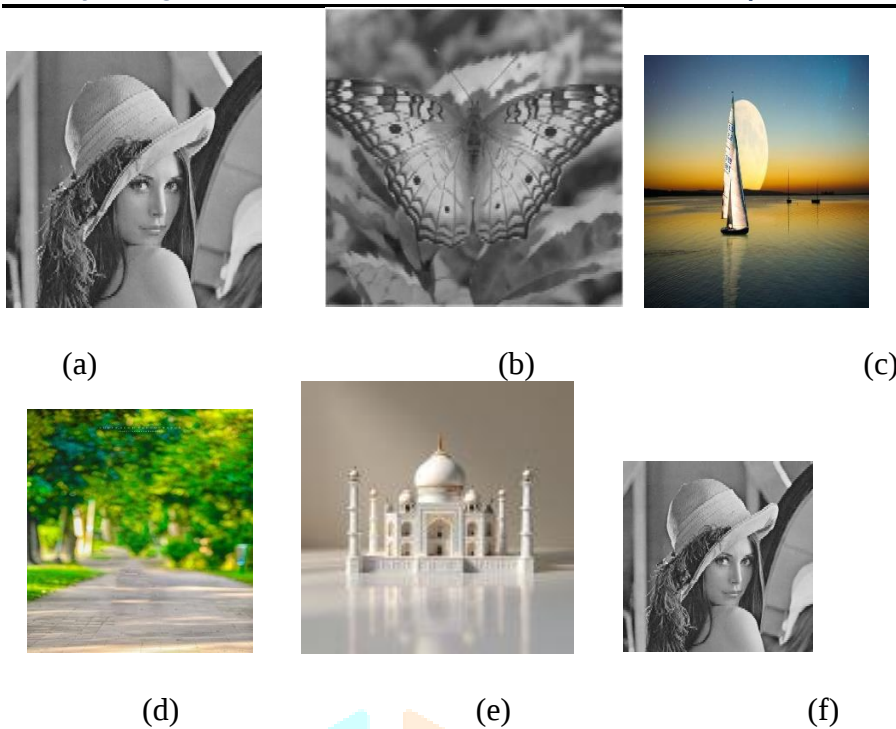


Fig 6. (a) Output image of 512x512 resolution(Up-scaling). (b) Output image of 512x512 resolution(Up-scaling).(c) Output image of 512x512 resolution(Down-scaling).(d) Output image of 500x500 resolution.(Down-scaling).(e) Output image of 128x128 resolution (Down-scaling). (f) Output image of 128x128 resolution (Down-scaling).

Input images of 256x256 resolution are shown in Fig.5(a), (b), (e).Fig.5(c), (d) have the resolutions 1920x1080 and 736x1104 respectively.

All input images are scaled up/down to 512x512, 500x500, 128x128 resolutions using bilinear interpolation. Fig.5(c), and Fig.6(c) shows the image is down scaled from 1920x1080 to 512x512 resolution. Fig.5(d) and Fig.6(d) shows the image is downscaled from 736x1104 to 500x500 resolution.

Table4.1 PSNR for 256x256 to 512x512 (Up-scaling)

Image	PSNR (dB)
Lena	28.325
Butterfly	26.764
Taj mahal	36.880

TABLE 4.2 PSNR FOR 512x512 TO 128x128 (DOWN-SCALING)

Image	PSNR (dB)
Lena	22.9757
Butterfly	32.6756
Taj mahal	26.4872
trees	33.7138
sailboat	32.6756

TABLE 4.3 SYNTHESIS RESULT OF PREVIOUS WORK AND PROPOSED WORK

Method	Delay(ns)
Adaptive (existing)	2.287
Bilinear (proposed)	1.902

V. CONCLUSION

This paper proposes a new VLSI architecture designed for high-speed image scaling in contemporary multimedia systems. With the utilization of hardware parallelism, pipelined data processing, and bilinear interpolation, the design provides high-quality output images with enhanced speed and efficiency. The architecture outperforms conventional methods by several orders of magnitude in terms of throughput, resource usage, and power dissipation, making it very suitable for portable and battery-powered devices. By using linear interpolation technique delay is reduced to 16.83% as shown in table 3. Designed with Verilog HDL and executed with MATLAB and Xilinx Vivado tools, the design turns out to be pragmatic as well as extensible, providing an expandable solution for existing as well as future image processing requirements.

REFERENCES

- [1] Midde Venkata Siva and Jayakumar E. P., "A Low-Cost High-Performance VLSI Architecture for Image Scaling in Multimedia Applications" IEEE Transactions on Very Large-Integration (VLSI) Systems (2020).
- [2] P. Madhavan, S. Arunachalam, V. Bharath Raj ., "VLSI implementation of low-cost-high-quality image scaling processor" International Research Journal of Modernization in Engineering Technology and Science (irjmets) Volume:06/Issue:06/June-2024.
- [3] A. K. Singh et al., "Low-Power and High-Performance VLSI Architecture for Image Scaling" International Journal of Electronics (2020).
- [4] J. Liu et al., "A Novel VLSI Architecture for Image Scaling Using Approximate Computing" IEEE Transactions on Very Large- Integration (VLSI) Systems (2020).
- [5] S. S. Bhattacharya et al., "An Energy-Efficient VLSI Architecture for Image Scaling" International Journal of Circuit Theory and Applications (2020).
- [6] V. Ramadevi and K. Manjunatha Chari, "FPGA realization of an efficient image scalar with modified area generation technique," Multimedia Tools and Appl., vol. 78, no. 16, pp. 23 707–23 732, Aug 2019.
- [7] Garg B, Chaitanya GVNSK, Sharma GK. "A low-cost energy efficient image scaling processor for multimedia applications," 20th Int. Symp. VLSI Design and Test (VDAT), pp. 1-6, 2016.
- [8] A. Sudhakaran and M. Manuel, "High-quality image scaling using vmodel," in Int. Conf. on Circuits and Syst. in Digit. Enterprise Technol., Dec 2018, pp. 1–4.
- [9] S.L. Chen, "VLSI implementation of an adaptive edge-enhanced image scalar for real-time multimedia applications," IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 9, pp. 1510–1522, Sep. 2013.
- [10] S. L. Chen, H. Y. Huang and C. H. Luo, "A low-cost high-quality adaptive scalar for real-time multimedia applications," IEEE Trans. Circuits Syst. Video Technol., vol. 21, no. 11 , pp. 1600–1611, Nov. 2011.