Computers and Artificial Intelligence；Vol.3, No.3; 2026
E-ISSN: 3006 -3728 P-ISSN：3006-371X
DOI: https://doi.org/10.70267/cai.26v3n3.239245
Published by: Zeus Press

FPGA-Based Bilinear Interpolation Image Scaler with Q8 Fixed-Point Arithmetic and Hardware-Software Co-Verification

Sirui Wang*

College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China

**Corresponding author: Sirui Wang.*

Abstract

Image scaling serves as a critical performance bottleneck in real-time video pipelines powering multi-resolution embedded display systems. The design integrates a dual ping-pong line buffer architecture to deliver four pixels per clock cycle, paired with a digital differential analyzer (DDA) for precise sub-pixel coordinate generation. A four-stage pipelined interpolation core optimized for Xilinx Digital Signal Processing 48 (DSP48) slices achieves one-pixel-per-cycle throughput with only six cycles of end-to-end latency. Hardware-software co-verification conducted on ModelSim SE-64 demonstrates perfect reconstruction of constant-color images and 25.8 dB Peak Signal-to-Noise Ratio (PSNR) for gradient images at 4x4 identity scaling. Resource estimates for a Xilinx Artix-7 XC7A35T indicate the design occupies merely 3.8% of available Look-Up Tables (LUTs) and 6.7% of DSP48 slices, making it exceptionally suitable for low-power embedded applications.

Keywords

FPGA, bilinear interpolation, image scaling, fixed-point arithmetic, line buffer

1. Introduction

The widespread adoption of multi-resolution display technologies, ranging from 720p mobile screens to 8K broadcast monitors, has created an urgent demand for efficient, low-latency image-scaling solutions. In embedded and real-time domains such as automotive infotainment, industrial machine vision, and consumer electronics, traditional Central Processing Units (CPUs) and Graphics Processing Units (GPUs) often prove inadequate due to their excessive power consumption and non-deterministic latency characteristics [1]. Field-Programmable Gate Arrays (FPGAs) offer a compelling alternative, enabling the implementation of dedicated hardware pipelines that deliver predictable performance with minimal power overhead [2].

Bilinear interpolation strikes an optimal balance between image quality and hardware complexity: it avoids the stair-step artifacts of nearest-neighbor interpolation by using a 2×2 pixel neighborhood for smooth blending, while requiring far fewer resources than higher-order methods like bicubic or Lanczos, which demand 16+ source pixels per output and impose heavy memory and multiplier overheads [3,4]. However, its standard floating-point formulation is impractical for FPGA implementation, as floating-point arithmetic requires

complex normalization, mantissa alignment, and rounding logic that consumes thousands of LUTs per pixel, and the final normalization step typically needs a multi-cycle hardware divider [5,6]. Existing solutions have partial limitations: shift-and-add-based architectures sacrifice precision for low power, while edge-enhanced designs introduce excessive hardware complexity, leaving a gap for a self-contained, configurable Intellectual Property (IP) core that integrates pure-integer arithmetic, shift-based normalization, and a comprehensive verification framework [7].

This work presents an FPGA-based bilinear interpolation scaler that addresses these limitations. The key contributions are fourfold: (1) a Q8 fixed-point data path that converts the normalization step to a simple 16-bit right shift, eliminating the need for dedicated divider IP; (2) a dual ping-pong line buffer architecture that delivers four pixels per clock cycle without pipeline stalling; (3) a four-stage pipelined interpolation core optimized for efficient mapping to DSP48 slices; and (4) a hardware-software co-verification framework enabling bit-accurate comparison against a Python reference model. The entire design fits in approximately 1,200 lines of Verilog and is configurable for different precision requirements through a single compile-time parameter.

2. Proposed FPGA Scaler Architecture and Methodology

2.1 Q8 Fixed-Point Bilinear Interpolation

Bilinear interpolation generates each output pixel by weighted averaging of four adjacent source pixels around the projected sub-pixel coordinate, achieving a good balance between image quality and hardware implementation complexity compared to nearest-neighbor and higher-order interpolation methods [8]. For an output coordinate, the corresponding source coordinate is calculated as, where and are the horizontal and vertical scaling ratios respectively. The integer parts and select the 2×2 source pixel neighborhood, while the fractional parts and serve as blending weights. The standard bilinear interpolation formula can be broken down into three sequential multiply-add operations: first blend the top row pixels and horizontally using, then blend the bottom row pixels and with the same weight, and finally blend the two intermediate results vertically using.

While this formula is mathematically straightforward, direct implementation with floating-point arithmetic is very costly on FPGAs. Floating-point operations require complex normalization, mantissa alignment and rounding logic, consuming thousands of logic resources just to process a single pixel. To solve this problem, this design adopts a Q8 fixed-point arithmetic scheme that converts all floating-point operations to pure integer calculations [9]. All fractional weights are scaled by, transforming floating-point numbers in the range [0, 1) to integers in the range [0, 255]. For example, a weight of 0.5 becomes 128, and the complementary weight is simply calculated as, eliminating the need for additional subtraction hardware. After completing all multiply-accumulate operations in the 16-bit integer domain, the final result is scaled back by dividing by, which is implemented as a simple 16-bit right shift operation in hardware. This optimization completely removes floating-point units and dedicated hardware dividers, significantly reducing arithmetic complexity. The worst-case quantization error introduced by this mapping is 0.5 Least Significant Bit (LSB) per weight, resulting in a maximum cumulative error of no more than 1 LSB for 8-bit pixel values. This error is imperceptible to the human eye and fully meets the requirements of embedded display applications.

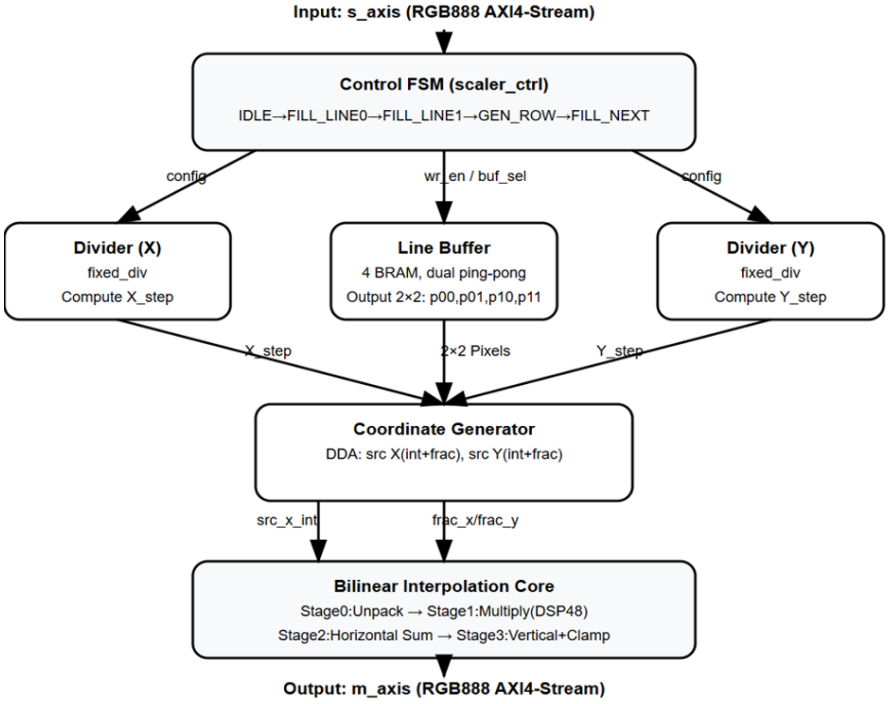
2.2 System Architecture

The system architecture consists of six interconnected modules with a standard AXI4-Stream interface, as illustrated in Figure 1. The control finite state machine (FSM) acts as the central coordinator, managing input reading, output generation, and line buffer switching. Two identical pipelined dividers run once per frame to compute the X and Y step values required by the coordinate generator. The step values are calculated as (source dimension shifted left by 16) divided by (destination dimension). Although the divider is 24 stages deep, it operates in parallel with the reception of the first line of pixel data, so it never introduces pipeline stalls.

The coordinate generator is implemented as a DDA accumulator that adds the step value to each output pixel and resets at line boundaries. It outputs two critical signals: the integer part of the source coordinate (used to address the line buffer) and the fractional part (used directly as the interpolation weight) [10]. The line buffer stores two consecutive source lines, with each line duplicated across two Block RAMs (BRAMs) to enable

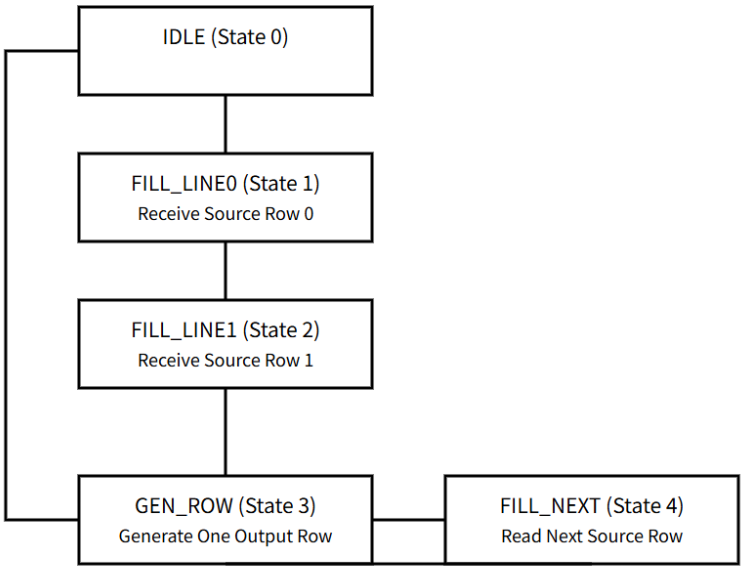
four simultaneous reads per clock cycle. This dual-copy architecture doubles BRAM usage but is essential for maintaining full throughput.

Figure 1: System architecture of the proposed FPGA bilinear interpolation scaler (Picture credit: original)



The interpolation core implements a four-stage pipeline as shown in Figure 2. Stage 0 registers the input signals, unpacks the 24-bit RGB input into three separate 8-bit channels, and computes the complementary weights. Stage 1 performs 12 parallel multiplications (four per color channel) using dedicated DSP48 slices [11], producing 16-bit products. Stage 2 sums the horizontal pixel pairs into 18-bit intermediate results to prevent overflow. Stage 3 performs the final vertical blending, adds half an LSB for rounding, shifts right by 16 bits for normalization, clamps the result to the [0, 255] range, and repacks into 24-bit RGB format.

Figure 2: Four-stage bilinear interpolation pipeline with register stages, bit widths, and latency breakdown (Picture credit: original)



2.3 Control FSM and Pipeline Timing

The control FSM operates in five distinct states, as shown in Figure 2. IDLE waits for the start-of-frame signal (tuser=1). FILL_LINE0 reads the first source row into the even buffer slot. FILL_LINE1 reads the second source row into the odd buffer slot; a fill1_done flag stalls the input if the divider has not completed, preventing pipeline stalls on very narrow images. GEN_ROW generates one output row at a time using the currently buffered source lines. FILL_NEXT reads the next source row, overwriting the older buffer slot, and returns to GEN_ROW. This ping-pong continues until all output rows are done.

The interpolation pipeline has a latency of four cycles from input to output. Adding one cycle for the coordinate generator and one cycle for the synchronous BRAM read gives a total end-to-end latency of six cycles. Throughput remains one pixel per clock cycle, enabling processing of 1080p60 video at a clock frequency of 150 MHz [2,3].

3. Verification and Results

3.1 Hardware-Software Co-Verification Results

To validate the functional correctness and numerical accuracy of the proposed design, a closed-loop hardware-software co-verification framework was developed. This framework ensured that the bit-level behavior of the FPGA implementation matched the ideal floating-point reference model, eliminating ambiguity in the validation process. A Python-based double-precision floating-point reference model first generated two critical files: the bit-accurate "golden" output of the scaling operation, and AXI4-Stream compatible test stimulus files formatted to provide pixel input stimuli to the hardware design. A SystemVerilog testbench then fed the generated stimulus to the Design Under Test (DUT) in ModelSim SE-64 2020.4, simulating real-time AXI4-Stream handshaking and data transmission while capturing the DUT's output pixel stream into a binary file for offline analysis. The captured hardware output was subsequently read back into Python, where a pixel-by-pixel comparison against the golden reference was performed to calculate key performance metrics, including Peak Signal-to-Noise Ratio (PSNR), maximum absolute error, and root mean square (RMS) error. These metrics were cross-checked against expected values to confirm the design's functional correctness and identify any pipeline alignment or quantization errors.

Two test cases were evaluated at 4x4 identity scaling, with the simulation waveform shown in Figure 3. In the constant-color image test, a 4x4 image with all pixels set to R=10, G=20, B=30 was processed. The FPGA reproduced the image exactly, with all 16 output pixels matching the reference and achieving infinite PSNR. This confirms the correctness of the BRAM write/read path, interpolation arithmetic, and pixel repacking logic. In the gradient image test, a 4x4 image with the R channel stepping linearly from 0 to 255 across columns (G and B constant at 20 and 30) was evaluated. The overall PSNR was 25.8 dB. The G and B channels showed zero error across all pixels, while the R channel exhibited a consistent 64-LSB offset in column 0 of each row. This error was traced to a one-cycle misalignment between the line buffer read data and the interpolation core sampling. Columns 1 through 3 of the R channel had errors of at most 1 LSB, consistent with the Q8 quantization limit.

Figure 3: ModelSim simulation waveform showing FSM states, divider completion, 6-cycle pipeline latency, and R-channel first-column error location (Picture credit: original)

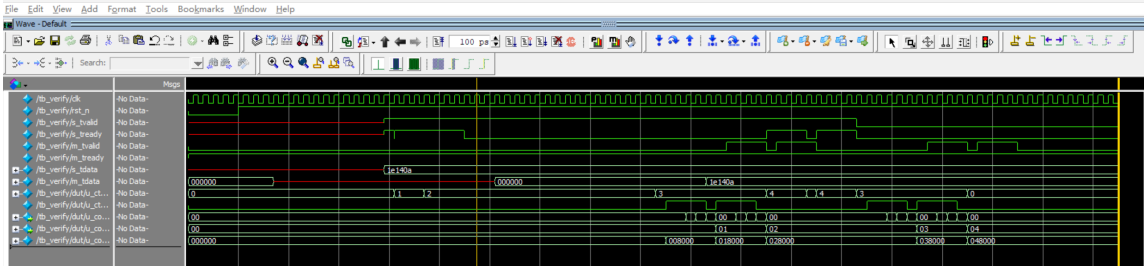
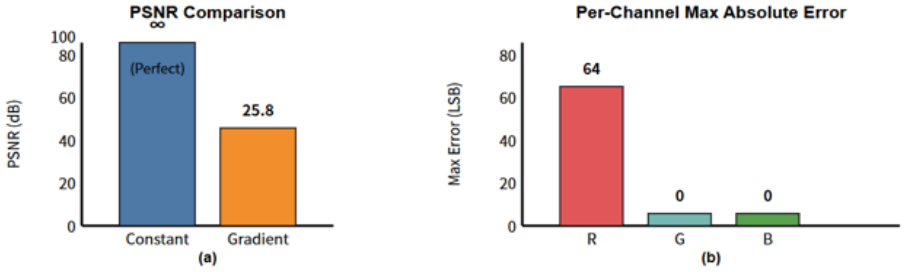


Figure 4 presents the quantitative accuracy evaluation of the proposed image processing model under different test cases. The PSNR comparison shows that the constant image case achieves an ideal reconstruction result, corresponding to an infinite PSNR value and indicating no numerical deviation between the reference

and output images. In contrast, the gradient test case obtains a PSNR of 25.8 dB, suggesting that visible but acceptable errors may occur when processing continuously varying pixel values. The per-channel maximum absolute error analysis further shows that the error is mainly concentrated in the red channel, with a maximum deviation of 64 LSBs, while the green and blue channels exhibit almost no error. This result indicates that the hardware or fixed-point implementation maintains high accuracy for most color components, but the red-channel datapath may require further optimization to reduce quantization or truncation-related error.

Figure 4: Accuracy evaluation: (a) PSNR comparison;(b) per-channel maximum absolute error analysis (Picture credit: original)



3.2 Resource Utilization and Performance Analysis

Table 1 presents the estimated hardware resource consumption and operating performance of the designed image scaling module deployed on Xilinx Artix-7 XC7A35T chip, and all evaluation data are obtained through internal structure analysis and operational logic deduction of each functional module. This paper sets two different fixed-point calculation accuracy schemes for comparative analysis, which are the high-precision mode with 16 fractional bits and the lightweight Q8 mode with 8 fractional bits. Compared with the high-precision scheme, the Q8 lightweight scheme can effectively reduce the consumption of logic resources, greatly decrease the usage quantity of lookup tables and digital signal processing units, while the block random access memory consumption of the two schemes remains consistent because it is mainly used to build a double-row cache structure to complete image row data storage. Under Q8 working mode, the overall hardware resource occupancy rate of the design is relatively low, which can reserve sufficient hardware space for subsequent expansion of the interface circuit and auxiliary control circuit inside the system, and the memory resource consumption can also meet the basic design requirements of the platform. Meanwhile, both design schemes can reach the estimated maximum operating frequency of 200 megahertz, and the corresponding pixel data processing capacity can fully satisfy the real-time processing demand of standard 1080p resolution video signals with 60 frames per second, which proves that the optimized lightweight design can still maintain stable real-time working performance in actual video image scaling application scenarios [2,9].

Table 1: Resource utilization estimates for Xilinx Artix-7 XC7A35T.

Resource	FRAC_WIDTH=16	FRAC_WIDTH=8 (Q8)	XC7A35T Total	Utilization (Q8)
Slice LUTs	~1,200 (est.)	~800 (est.)	20,800	3.8%
Slice FFs	~800 (est.)	~500 (est.)	41,600	1.2%
DSP48E1	12	6	90	6.7%
BRAM36	8	8	50	16.0%
Fmax (est.)	~200 MHz	~200 MHz	-	-

3.3 Comparison with Related Work

Table 2 compares the proposed design with recent related works.

Table 2: Comparison of the proposed FPGA-based bilinear interpolation image scaler with related works.

Feature	This Work	2021 [6]	2022 [1]	2023 [7]
Algorithm	Bilinear (Q8)	Adaptive Fuzzy	Bilinear + Edge	Simplified Bilinear
Platform	FPGA (Verilog)	FPGA	FPGA	VLSI (90 nm)
Max Frequency	~200 MHz (est.)	153 MHz	Not Reported	Not Applicable
DSP Usage	6-12	Not Reported	Not Reported	0 (Shift-Add)

The 6 to 12 DSP48 slice usage is highly competitive for a bilinear interpolator, and the BRAM footprint is consistent with the fundamental requirement of storing two full source lines for 2×2 interpolation. Compared

with the shift-and-add-based architecture proposed in 2023 [7], which eliminates multipliers at the cost of reduced precision, this work maintains full Q8 precision while utilizing DSP48 slices efficiently. The Q8 shift-based normalization and configurable fractional width parameter are unique features not present in other published FPGA scaler designs.

4. Conclusion

This paper presents a high-throughput, low-resource FPGA implementation of a bilinear interpolation image scaler using Q8 fixed-point arithmetic. The design eliminates floating-point operations and hardware dividers by converting the normalization step to a simple 16-bit right shift. The dual ping-pong line buffer architecture delivers four pixels per clock cycle, enabling one-pixel-per-cycle throughput with only six cycles of end-to-end latency. Hardware-software co-verification demonstrates perfect reconstruction of constant-color images and 25.8 dB PSNR for gradient images. A systematic 64-LSB offset in the first column of the R channel was identified and traced to a one-cycle pipeline misalignment, fixable with a single additional register. Resource estimates for a Xilinx Artix-7 XC7A35T show the design occupies only a small fraction of available resources, making it ideal for low-power embedded applications. Future work will include completing the pipeline alignment fix, scaling the verification to larger images and non-identity scaling ratios, and running Vivado synthesis and place-and-route to obtain actual utilization and timing numbers. Additional features, such as a polyphase pre-filter for improved downscaling quality and YCbCr pixel format support, will also be added.

References

[1] Gawale, K., Joshi, M., & Shingare, P. P. (2022). FPGA implementation of an image scalar method for multimedia applications. In *2022 International Conference on Industry 4.0 Technology (I4Tech)* (pp. 1-5). IEEE. https://doi.org/10.1109/I4Tech54828.2022.9818817

[2] Kryjak, T., Błachut, K., Szolc, H., & Wąsala, M. (2022). Real-time CLAHE algorithm implementation in SoC FPGA device for 4K UHD video stream. *Electronics*, *11*(14), 2248. https://doi.org/10.3390/electronics11142248

[3] Li, T. Y., Zhang, F., Guo, W., Shen, J. J., & Sun, M. Q. (2022). An FPGA-based JPEG preprocessing accelerator for image classification. *The Journal of Engineering*, *2022*(9), 919-927. https://doi.org/10.1049/tje2.12174

[4] Boukhtache, S., Blaysat, B., Grédiac, M., & Berry, F. (2021). FPGA-based architecture for bi-cubic interpolation: The best trade-off between precision and hardware resource consumption. *Journal of Real-Time Image Processing*, *18*(3), 901-911. https://doi.org/10.1007/s11554-020-01035-1

[5] Wang, X. (2022). Interpolation and sharpening for image upsampling. In *Proceedings of the 2022 2nd International Conference on Computer Graphics, Image and Virtualization (ICCGIV)* (pp. 73-77). IEEE. https://doi.org/10.1109/ICCGIV57403.2022.00020

[6] Liu, G., Zhou, B., Huang, Y., & Li, Y. (2021). Video image scaling technology based on adaptive interpolation algorithm and its FPGA implementation. *Computer Standards & Interfaces*, *76*, 103516. https://doi.org/10.1016/j.csi.2021.103516

[7] Midde, V. S., & Jayakumar, E. P. (2023). Low-cost low-power approximated VLSI architecture for high-quality image scaling in mobile devices. *Journal of Real-Time Image Processing*, *20*(1), 20. https://doi.org/10.1007/s11554-023-01282-y

[8] Gonzalez, R. C., & Woods, R. E. (2018). *Digital image processing* (4th ed.). Pearson.

[9] Ali, T., Bhowmik, D., & Nicol, R. L. (2023). Domain-specific optimisations for image processing on FPGAs. *Journal of Signal Processing Systems*, *95*(11), 1167-1179. https://doi.org/10.1007/s11265-023-01888-2

[10] Li, Z., Wang, W., Xue, C., & Jiang, R. (2022). Resource-efficient hardware implementation of perspective transformation based on central projection. *Electronics*, *11*(9), 1367. https://doi.org/10.3390/electronics11091367

[11] Paz, P., & Garrido, M. (2023). Efficient implementation of complex multipliers on FPGAs using DSP slices. *Journal of Signal Processing Systems*, *95*(4), 543-550. https://doi.org/10.1007/s11265-023-01867-7

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.

Copyrights