

Design and Verification of a Bicubic Image Interpolation Core for Image Upscaling

Reshma Dasari and Dr. P. Pushpalatha

Department of Electronics and Communication Engineering, University College of Engineering Kakinada (UCEK), JNTU-Kakinada, Andhra Pradesh, India

Abstract:- Image interpolation is a fundamental operation in digital image processing used to resize images while preserving visual quality. Among the common interpolation techniques, bicubic interpolation offers a favourable trade-off between computational complexity and output smoothness compared to nearest-neighbour and bilinear methods. This study presents the design, hardware description, and functional verification of a bicubic image interpolation core implemented in Verilog HDL and targeted to an Xilinx Artix-7 FPGA. The proposed core computes cubic interpolation weights for a 4×4 neighbourhood of source pixels and accumulates the weighted contributions to generate each destination pixel, with a control finite-state machine sequencing the operation across the full destination image. The design was verified both at the coordinate level and at the full-image level, including natural-image and synthetic edge-pattern test cases, and matched the expected output in all cases. Post-implementation, timing-constrained synthesis results show a lightweight resource footprint of 39 look-up tables (LUTs), 36 flip-flops (FFs), no block RAM usage, a total power of 0.093 W, and a positive worst-negative-slack (WNS) of 5.438 ns, confirming timing closure. The architecture draws conceptual motivation from recent literature on optimized radix-2^m cubic arithmetic units, adapting cubic-weight computation principles into a complete, application-level interpolation pipeline rather than an isolated arithmetic primitive.

Keywords: Bicubic interpolation, FPGA, Verilog HDL, image upscaling, VLSI, low-power design, RTL synthesis.

1. Introduction

Digital image scaling is widely used in consumer electronics, medical imaging, remote sensing, and multimedia streaming applications, where source images must be resized to match display resolutions or downstream processing requirements. Software-based interpolation, while flexible, is often too slow for real-time or power-constrained embedded platforms. Therefore, hardware acceleration of interpolation algorithms on FPGAs and ASICs remains an active area of research, particularly for portable and battery-operated devices, where area and power efficiency are critical design constraints.

Among interpolation techniques, nearest-neighbour interpolation is the simplest but produces blocky artifacts; bilinear interpolation improves smoothness using a 2×2 pixel neighbourhood; and bicubic interpolation extends this further using a 4×4 neighbourhood and cubic polynomial weighting, yielding sharper and visually smoother results at the cost of additional arithmetic complexity [2]. This additional complexity motivates the hardware-level optimization of the underlying cubic arithmetic.

This work presents a Verilog HDL implementation of a bicubic image interpolation core, covering weight generation, neighbourhood pixel fetch, weighted accumulation, output clamping, and a control sequencer that iterates the operation across an entire source-to-destination image mapping. The design is verified through directed simulation and synthesized for an Xilinx Artix-7 target. The remainder of this paper is organised as follows: Section 2 states the objectives of the work; Section 3 describes the proposed architecture and the implementation and verification methodology; Section 4 presents the verification and synthesis results; and Section 5 discusses the findings, design trade-offs, and future work. “Cubic operations are among the most used arithmetic operations in many applications that demand higher order simultaneous operand computation.”

The efficient hardware realization of cubic arithmetic has been extensively studied in the context of both cryptographic operations and image processing. da Costa and da Rosa proposed a family of radix-2^m cubic units (RCU-4, RCU-8, and RCU-16) that reduce partial-product generation for the cube operation A^3 by processing multiple bits simultaneously, reporting significant area and energy savings over prior Vedic-mathematics-based and automatically synthesized cubing approaches on both ASIC and FPGA targets [1]. Their RCU-16 configuration, which processes four bits per encoder step, was shown to be the most area- and energy-efficient variant across the configurations studied, outperforming several prior works that compute cubic partial products bit-by-bit [1].

Earlier efforts toward efficient cubic and multiplicative hardware include Vedic-mathematics-based cubing circuits and carry-save array approaches, which reduce latency relative to conventional multiplier chains but do not exploit multi-bit radix encoding to the same extent as the RCU-2^m family [1]. The underlying interpolation theory used throughout this study traces back to Keys, who formalised cubic convolution as a piecewise-polynomial resampling kernel, offering accuracy between linear interpolation and cubic splines [2].

On the hardware architecture side, several FPGA implementations of bicubic interpolation have been reported. Nunez-Yanez proposed one of the earliest dedicated bicubic hardware architectures for real-time digital image scaling, which was structured around weight-generation and accumulation blocks [3]. Zhang et al. presented an FPGA realization of the bicubic enlargement algorithm exploiting hardware parallelism to improve computation speed and resource usage [4]. More recently, Hou et al. surveyed and compared several approximations to full bicubic interpolation, combining linear and cubic stages, to reduce the multiplier and LUT consumption on Intel Cyclone V targets, trading interpolation quality against resource cost [5]. Ben Atitallah et al. proposed a reformulated bicubic algorithm that eliminates redundant multiplications and studied the precision-versus-resource trade-off across Xilinx and Intel FPGA targets [6].

The present work differs from these efforts in scope and motivation: rather than optimizing the interpolation algorithm itself for resource reduction, as in [5] and [6], or proposing a new bicubic hardware architecture, as in [3] and [4], this study implements a complete, directly coded bicubic interpolation datapath, including neighborhood pixel access, weight computation, accumulation, and image-level control, while drawing on the general principle from [1] that multi-bit, structured arithmetic decomposition can keep cubic-weight computation lightweight even when embedded within a larger pipeline. The mathematical foundation of the proposed interpolation approach is based on cubic convolution, which uses neighbouring pixels and interpolation weights to estimate the intensity of a new pixel [2].

2. Objectives

The primary objective of this work is to design and implement, in Verilog HDL, a complete bicubic image interpolation core covering cubic weight generation, 4×4 neighbourhood pixel fetch, weighted accumulation, output clamping, and a control sequencer that maps every destination pixel of an upscaled image to the source image.

The second objective is to verify the core functionally at both the coordinate level and the full-image level using natural-image and synthetic edge-pattern test cases, and to synthesize and implement it on an Xilinx Artix-7 FPGA under a clock constraint so that resource utilisation, power, and timing closure can be reported. A further objective is to draw on the principle of structured, multi-bit cubic arithmetic from the RCU-2^m literature [1] so that cubic-weight computation remains lightweight within a complete, application-level interpolation pipeline rather than an isolated arithmetic primitive.

3. Methods

The proposed core is organised into four functional stages, coordinated by a control finite-state machine that iterates over the destination pixel coordinates (i, j) :

- **Neighbourhood fetch:** For each destination coordinate, the corresponding 4×4 block of source pixels is identified and read from the source image memory.

- **Cubic weight computation (cubic_weights.v):** Fractional interpolation offsets are used to compute the four cubic weighting coefficients required for each dimension, following the standard cubic convolution kernel of Keys [2].
- **Weighted accumulation (bicubic_interp.v):** Each of the 16 source pixels in the neighbourhood is multiplied by its corresponding two-dimensional weight (the product of the row and column weights) and accumulated to form the interpolated pixel value.
- **Output clamping and sequencing (image_upscaler.v):** The accumulated result is range-limited to a valid pixel value, written to the destination image, and the control FSM advances to the next destination coordinate until the full $DST_W \times DST_H$ image is produced.

This structure separates the arithmetic-heavy weight and accumulation stages from the control and memory-access logic, allowing the arithmetic core to be optimized independently, consistent with the motivation in [1] for isolating and streamlining cubic-related computation within a larger system.

The core was described in Verilog HDL across three primary modules -- cubic_weights.v, bicubic_interp.v, and image_upscaler.v -- along with the accompanying testbenches. Functional simulation was first performed using the Xilinx Vivado XSim simulator to validate module-level and integration-level behavior prior to FPGA synthesis.

Verification was carried out in two stages. First, a directed coordinate-level test case was used: for input coordinates $tx = ty = 128$, the design produced an output pixel value of 85, which matched the expected value exactly, confirming correct arithmetic behavior of the weight computation and accumulation stages in isolation.

Second, full image-level verification was performed by loading a 16×16 source image into simulation memory and driving the core through the complete range of destination coordinates to reconstruct a 64×64 ($4 \times$) upscaled output, matching the pixel-neighbourhood and weight values observed in the simulation against a reference bicubic computation. Two source images were used for this stage: (i) a natural grayscale photographic thumbnail to assess general visual smoothness and structure preservation, and (ii) a synthetic test pattern containing a sharp diagonal edge to assess the core's behavior at high-contrast discontinuities.

Subsequently, the design was synthesized and implemented in Xilinx Vivado, targeting the Artix-7 (xc7a100tcs324-1) device. A timing constraint (create_clock) was defined on the design's clock port to ensure that post-implementation power and timing results reflect activity-driven, clock-constrained analysis rather than default vectorless estimation.

4. Results

Fig. 1 shows the natural-image test case. The upscaled output preserves the coarse structure of the source image while smoothing block-edge transitions, consistent with expected bicubic behavior. Fig. 2 shows the diagonal-edge synthetic test case; here, the upscaled result exhibits localised overshoot and ringing artifacts along the diagonal boundary, which is a well-known characteristic of cubic convolution kernels at sharp intensity discontinuities and is consistent with, rather than indicative of, a defect in the implemented arithmetic.

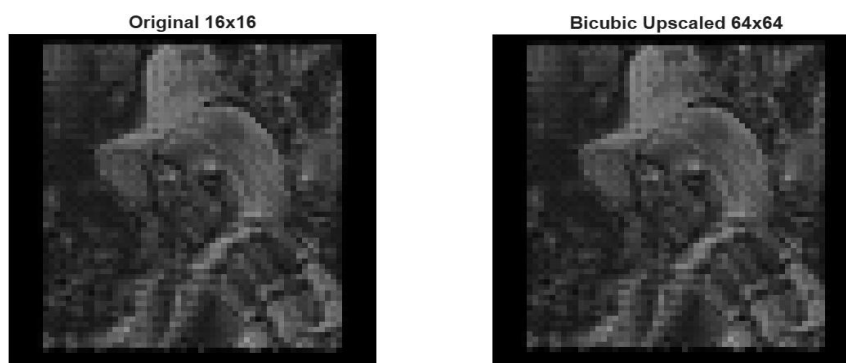


Fig. 1. Natural grayscale image: original 16×16 source (left) and bicubic-upscaled 64×64 output (right).

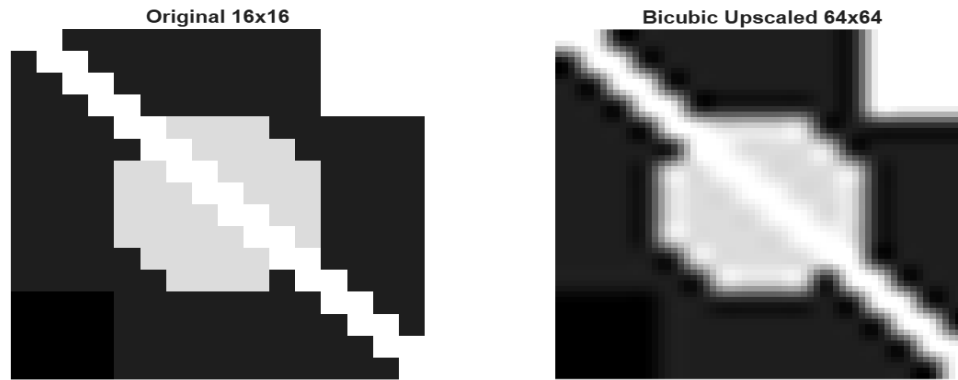


Fig. 2. Synthetic diagonal-edge pattern: original 16×16 source (left) and bicubic-upscaled 64×64 output (right), showing expected edge overshoot.

Under the create_clock constraint, the design achieved full timing closure with a worst-negative-slack of 5.438 ns and zero total negative slack, and was fully routed with no failed routes.

Table I summarises the post-implementation resource utilisation and verification status of the design.

Table I. Post-implementation, timing-constrained results for the bicubic interpolation core on Artix-7 FPGA

Parameter	Value	Notes
Target device	Artix-7 (xc7a100tcs324-1)	Xilinx Vivado 2026.1
LUTs	39	Post-implementation, timing-constrained
Flip-Flops	36	Post-implementation, timing-constrained
Block RAM	0	No BRAM required
Functional verification	Pass (tx=ty=128 $\rightarrow$ 85)	Exact hardware match
Total power	0.093 W	create_clock-constrained, post-route
WNS	5.438 ns	Positive slack; timing met
TNS	0.000 ns	No timing violations
Failed routes	0	Fully routed

The design occupies a lightweight footprint of 39 LUTs and 36 FFs, with no block RAM required, and consumes 0.093 W under a clock-constrained implementation run, indicating that the core is suitable for integration into resource-constrained, low-power embedded imaging pipelines. Notably, this resource and power figure reflects a complete interpolation pipeline (neighbourhood fetch, weight computation, accumulation, clamping, and image-level sequencing) rather than an isolated cubic arithmetic primitive such as the RCU-16 unit reported in [1].

5. Discussion

A direct numerical comparison with [1] is not presented because the two designs differ in functional scope, target device family, and constraint methodology; such a comparison would require isolating an equivalent cubic-weight sub-block from the present design and evaluating it under the same conditions.

The functional correctness of the design was confirmed at both the coordinate and full-image levels. The natural-image test (Fig. 1) demonstrates visually coherent, smoothly interpolated output from a real photographic source, whereas the synthetic edge test (Fig. 2) demonstrates correct handling of high-contrast content and reproduces the overshoot behavior expected of cubic convolution kernels. The absence of routing failures during implementation further indicates that the design is physically realizable on the target device without resource conflicts.

Balancing pipeline depth and latency against FPGA resource constraints involves carefully designing the processing stages to optimise throughput without exceeding available logic elements and registers. Deeper

pipelines can increase clock frequency and throughput but consume more resources and introduce latency, so a trade-off must be found that meets timing and area goals.

Handling edge pixels and boundary conditions in the interpolation requires special treatment since the 4×4 neighbourhood may extend beyond the image borders. Common strategies include replicating border pixels, mirroring the image edges, or padding with zeros to ensure valid input for the cubic convolution kernel without introducing artifacts.

Trade-offs between fixed-point and floating-point arithmetic in FPGA implementation center on resource usage, precision, and performance. Fixed-point arithmetic is more resource- and power efficient and typically preferred for embedded FPGA designs, but it requires careful scaling and overflow management to maintain accuracy. Floating-point offers greater dynamic range and ease of development but at significantly higher resource cost and complexity, often making it unsuitable for low-power or resource-constrained applications.

This study presents the design, implementation, and verification of a bicubic image interpolation core in Verilog HDL targeted to an Xilinx Artix-7 FPGA. The design was functionally verified against a directed test case and synthesized with a lightweight resource footprint. Future work includes extending verification to a broader set of test images and coordinate ranges, applying activity-driven power analysis using simulation switching data, and exploring radix-based optimization of the internal cubic weight computation, following the direction of [1], to further reduce area and power within the complete interpolation pipeline.

Acknowledgment

The authors thank Dr. P. Pushpalatha, Department of Electronics and Communication Engineering, UCEK, JNTU-Kakinada, for her guidance and supervision throughout this project.

References

- [1] E. A. C. da Costa and M. M. A. da Rosa, "RCU-2^m: A VLSI Radix-2^m Cubic Unit," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 33, no. 3, pp. 733–745, Mar. 2025.
- [2] R. G. Keys, "Cubic convolution interpolation for digital image processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, no. 6, pp. 1153–1160, Dec. 1981.
- [3] J. L. Nunez-Yanez, "Real-time FPGA-based architecture for bicubic interpolation: an application for digital image scaling," in Proc. Int. Conf. Reconfigurable Computing and FPGAs (ReConFig), 2005.
- [4] Y. Zhang, Y. Li, J. Zhen, J. Li, and R. Xie, "The hardware realization of the bicubic interpolation enlargement algorithm based on FPGA," in Proc. 3rd Int. Symp. on Information Processing, 2010, pp. 277–281.
- [5] W. Hou, D. Sun, X. Zhang, and Z. Feng, "Alternatives to bicubic interpolation considering FPGA hardware resource consumption," IEEE Access, vol. 8, pp. 199111–199123, 2020.
- [6] A. Ben Atitallah, D. Kachouri, and M. Masmoudi, "FPGA-based architecture for bi-cubic interpolation: the best trade-off between precision and hardware resource consumption," Journal of Real-Time Image Processing, vol. 18, pp. 1103–1119, 2021.