High Clarity, Low Power: Achieving a 43× Speed Boost in Image Defogging Using FPGA Acceleration

Van Khoa Pham
Ho Chi Minh City University of Technology and Education, Vietnam
khoapv@hcmute.edu.vn (corresponding author)

Trung Tin Le
Ho Chi Minh City University of Technology and Education, Vietnam
2390707@student.hcmute.edu.vn

Received: 5 May 2025 | Revised: 20 June 2025, 2 July 2025, and 5 July 2025 | Accepted: 8 July 2025

 DOI: https://doi.org/10.48084/etasr.11931

ABSTRACT

This study presents a Field-Programmable Gate Array (FPGA) accelerator designed for real-time image defogging at the edge, achieving high throughput and low power consumption. The design adapts the Dark Channel Prior (DCP) algorithm for hardware implementation using High-Level Synthesis (HLS) and incorporates advanced optimizations such as pipelining, loop unrolling, and dataflow control to enhance processing on resource-constrained devices. Implemented on a PYNQ-Z2 board running at just 100 MHz, the system achieves a remarkable 12.06 Frames per Second (FPS), surpassing a 2 GHz ARM processor by over 43× in speed. Power measurements show a low power consumption of 0.79 W, translating to a 153× improvement in energy efficiency (FPS/W) compared to an ARM-based software implementation. The proposed accelerator introduces a 4.7% pixel-level error, primarily affecting brightness consistency; nonetheless, it significantly outperforms processor-based approaches in both latency and power consumption. By demonstrating how FPGAs can sustain high-clarity image enhancement at the network edge, this work lays the groundwork for deployment in autonomous vehicles, remote surveillance systems, and environmental monitoring platforms where robust, low-power vision processing is critical.

Keywords-Dark Channel Prior (DCP); High-Level Synthesis (HLS); hardware-accelerated system; real-time defogging; edge devices

I. INTRODUCTION

Images captured by edge devices often suffer from degradation in foggy, hazy, or low-visibility conditions. Defogging is therefore an essential preprocessing step to restore clarity and enhance the performance of subsequent vision tasks, such as object detection and recognition. Achieving real-time defogging on resource-limited edge hardware is challenging, however, due to high computational demands, tight memory-bandwidth budgets, and strict power envelopes. Classical algorithms like Dark Channel Prior (DCP) [1] and Retinex-based methods [2] rely on atmospheric-scattering models, guided filtering, and histogram equalization. These operations are computationally intensive for embedded processors and low-power system-on-chips [3-5]. Limited on-chip memory necessitates repeated off-chip access for frame and intermediate data storage. As demonstrated by authors in [5], external memory access is significantly slower and consumes more energy than local computation. Techniques such as block processing, data reuse, and aggressive quantization are therefore vital for real-time throughput [5]. Deep learning-based approaches, such as DehazeNet [3] and AOD-Net [4], exacerbate the challenge, as their reliance on millions of Multiply-Accumulate (MAC) operations rapidly depletes the battery life of mobile platforms like drones and robots. Authors in [6] confirm that MACs and off-chip memory access dominate power consumption in vision pipelines. However, practical systems—such as autonomous navigation, smart surveillance—require real-time defogging, as any delay may compromise safety or usability [7]. Hardware-efficient strategies such as fixed-point arithmetic, network pruning, and Field-Programmable Gate Array (FPGA) acceleration have therefore been proposed to reduce latency while preserving image quality [8].

Researchers are now turning to purpose-built accelerators: FPGA implementations [8, 9], Application-Specific Integrated Circuit (ASIC) designs for ultra-low-power dehazing [10], and edge-AI chips tuned for enhancement workloads [11]. This work presents an FPGA-based defogging engine that implements a refactored version of DCP using Vitis High-Level Synthesis (HLS), aiming for higher throughput, lower energy per frame, and minimal quality loss. The resulting

system bolsters autonomous-vehicle safety, enhances the reliability of traffic surveillance in adverse weather, and sharpens remote sensing in fog or smog, paving the way for high-quality, low-power defogging at the edge.

II. ALGORITHM SELECTION AND BACKGROUND

This study surveys three leading defogging algorithms including DCP, Retinex, and Contrast Limited Adaptive Histogram Equalization (CLAHE), and compares their performance, complexity, FPGA suitability, and real-time potential. Using Vivado and Vitis HLS, this study maps its algorithm to FPGA, simulates accuracy-latency-resource trade-offs, and validates the design on real hazy images, measuring quality, speed, and energy. This work is based on the atmospheric scattering model, which defines the relationship between light attenuation and airborne particles to guide the image restoration process.

Visual perception, whether by the human eye or a camera sensor, relies on the light reflected from objects and light sources. However, molecules in the air scatter this light, reducing visibility, with the degree of visibility reduction depending on the type and concentration of molecules in the air. Authors in [1] developed a model that describes this process as:

$$I(x) = t(x)J(x) + \big(1 - t(x)\big)A \quad (1)$$

where $I(x)$ represents the observed image, specifically the hazy image being analyzed, $J(x)$ denotes the clear image without haze, also known as the scene radiance, and A refers to the brightness contributed by the atmosphere to the image. The variable $t(x)$, ranging from 0 to 1, represents the transmission map. It indicates the proportion of light that is not scattered and can reach the camera**,** defined as follows:

$$t(x) = e^{-\beta d(x)} \quad (2)$$

Here, $t(x)$ represents the transmission coefficient at a specific point x, describing the proportion of light that is not scattered and successfully reaches the camera, β denotes the atmospheric scattering coefficient, indicating the extent to which the scene radiance is exponentially attenuated as the scene depth increases, and $d(x)$ is the scene depth map. Based on (1), to recover J from I, the values of A and t need to be determined beforehand as input parameters.

$$J(x) = \frac{I(x) - A}{t(x)} + A \quad (3)$$

The design of defogging algorithms is guided by the atmospheric scattering model, which is mathematically described by the inverse Koschmieder law as shown in (3) [12]. This model explains image degradation by describing how light is scattered by particles of varying composition and density in fog, haze, and smoke. A significant challenge in defogging, a crucial preprocessing step for computer vision tasks like object recognition and tracking, is the absence of depth information in a single image.

The main approaches to single-image defogging can be categorized into three groups: image enhancement, prior-based methods, and deep learning techniques. While early methods focused on contrast enhancement, more recent techniques leverage the atmospheric scattering model. Among these, the DCP proposed by authors in [13, 14] has proven to be particularly effective. The definition of the Dark Channel is expressed as follows:

$$I^{dark}(x) = \min_{y \in \Omega(x)}(\min_c(I^c(y))) \quad (4)$$

The DCP estimates haze transmission and atmospheric light from local RGB minima, then inverts the scattering model to recover clear images. Because it explicitly models atmospheric scattering, DCP restores visibility more reliably and avoids the color distortions or residual fog seen with Retinex (illumination boosting) and CLAHE (local contrast stretching). Its core min-filter and per-pixel operations also map cleanly to FPGA pipelines, making DCP both the most effective and hardware-friendly method among those compared. In the context of [15], the iterations are applied to the Red (R), Green (G), and Blue (B) color channels, creating I^c, corresponding to one of the color channels from the input image I. $\Omega(x)$ denotes a small patch centered at point x.

In many outdoor scenes, except for the sky region, the brightness of I^{dark} is usually very low and often close to zero.

$$I^{dark}(x) \approx 0 \quad (5)$$

Based on this assumption, the dark channel is arranged according to the brightness values of each pixel:

$$I^{sorted} = \text{argmax}_x\big(I^{dark}(x)\big) \quad (6)$$

As above, argmax_x provides the indices of the values in the dark channel, sorted in descending order. These indices are stored in I^{sorted}. Next, the global atmospheric light A^c is calculated as follows:

$$A^c = \sum_{i=0}^{N-1} I^c(I^{sorted}(i))/N \quad (7)$$

where N represents 0.01% to 0.1% of the pixels with the highest intensity values, determined from (6). Once the haze coefficient ω is obtained, the transmission map for the input scene can be estimated as follows:

$$t(x) = 1 - \omega \times \min_{y \in \Omega(x)}(\min_c \frac{I^c(y)}{A^c}) \quad (8)$$

III. THE PROPOSED SYSTEM

Figure 1 illustrates the complete pipeline, from raw input to restored output, which is built upon algorithms refined for FPGA deployment. This pipeline focuses on extracting and reconstructing the Dark Channel to enhance image quality while optimizing hardware resource utilization. The process begins by identifying the minimum pixel intensity across the RGB channels to create the Dark Channel. Computation is accelerated using an optimized 2-D filter to form the Dark Channel matrix, an approach that avoids nested loops. Next, a diffusion matrix is derived by comparing 'img_minmat' with 'img_darkChannel' and applying a tuned diffusion coefficient ($\Omega = 0.6$) to preserve bright features such as mist or faint illumination. A brightness-adjustment formula restores the scene, removing dark artifacts while keeping natural highlights, and a final look-up table balances luminance and sharpens

details. The overall goal is to shorten processing time, reduce FPGA resource usage, and deliver clear, evenly lit images.

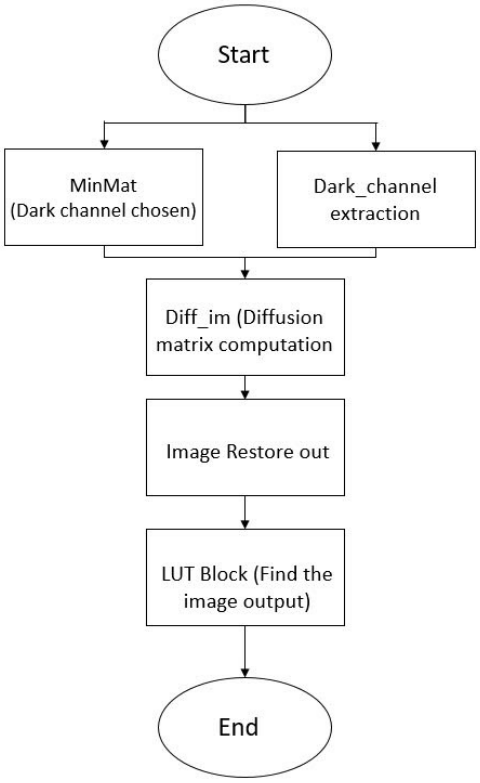

Fig. 1. Implementation diagram of the proposed defogging algorithm.

Figure 2 shows a complete FPGA-based defogging system that integrates hardware, software, and tuning for real-time use. Images are received by the processing system and buffered in DDR3L memory. They are then streamed via an AXI4 interface to the programmable logic, where defogging and post-processing occur, before being sent back for output. Vitis HLS synthesizes the C/C++ code into optimized hardware descriptions (RTL) using Xilinx-specific data types, such as 'ap_uint<24>'. A C simulation verifies functionality before synthesis, enabling code refinement and directive insertion to boost performance and minimize resource usage.

As shown in Figure 3, Vitis HLS compiles optimized C/C++ code into HDL via high-level synthesis and validates its behavior through Co-Simulation, enabling early error detection. It evaluates performance metrics such as latency, throughput, and resource usage (LUTs, DSPs, and BRAMs) to guide hardware optimization. Once verified, the HDL is packaged as a ready-to-use IP core with constraints, models, and drivers for seamless Vivado integration. This study targets the PYNQ-Z2 board, aligning the flow with Zynq-7020's architecture to ensure efficient bitstream execution. The design process—from algorithm validation to IP generation—reduces development time and complexity, improving overall system performance. The accelerator consists of four hardware blocks reflecting the software pipeline [16]: MinMat computes the minimum RGB value per pixel; Dark Channel selects the darkest pixels using original and MinMat data; Image Diffusion applies anisotropic filtering for dehazing [17]; and Image Restoration reconstructs the final color image. Table I outlines the HLS directives used in each block.

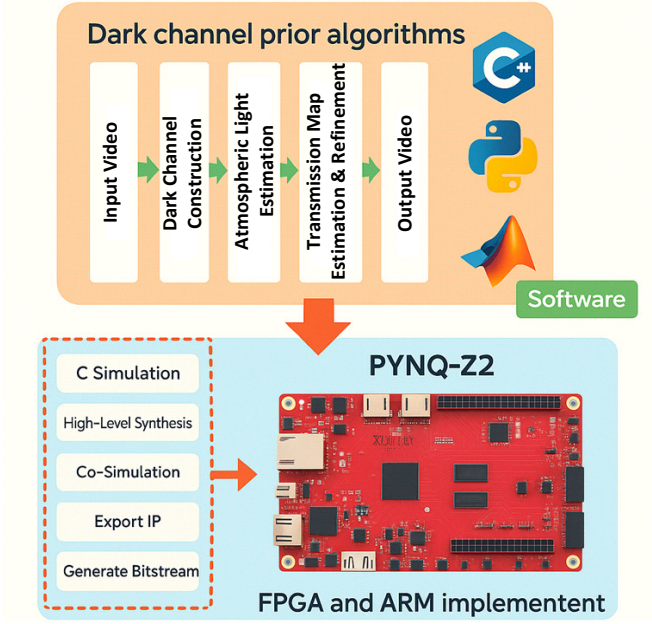

Fig. 2. Software development flow diagram.

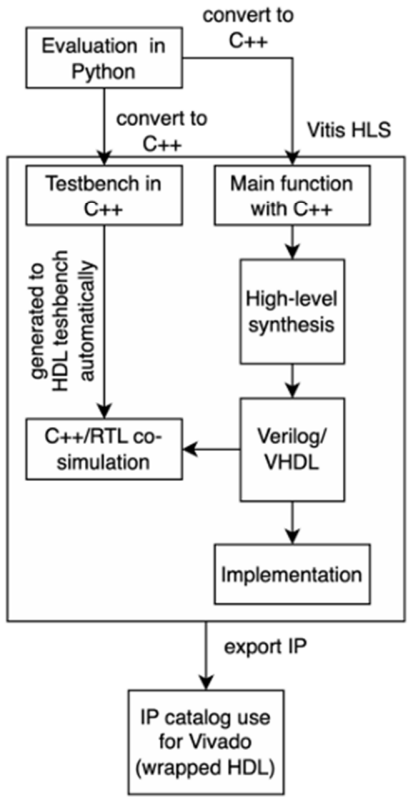

Fig. 3. System development process diagram.

TABLE I. TECHNIQUES USED IN THIS STUDY

Block	Description
MinMat	Pipeline, dataflow, loop tripcount, stream variables, NPPC8
Dark Channel	Pipeline, dataflow, stream variables, NPPC8
Image Diffusion	Pipeline, dataflow, NPPC8
Image Restoration	Pipeline, dataflow, streaming, NPPC8

This work accelerates pixel-scanning loops by combining pipelining, loop unrolling, and the Vitis HLS dataflow model to maximize FPGA resource utilization. Pipelining splits processing into stages—pixel fetch, filtering, and write-back—that operate concurrently, allowing a new region to enter each clock cycle. The '#pragma HLS dataflow' directive enables stage-level parallelism by triggering execution as soon as input data are ready, reducing stalls and balancing LUT, DSP, and BRAM usage. Image kernels scan the frame using two nested loops: the outer loop iterates over 8-pixel words using a three-bit right shift for efficiency (259,200 iterations for a 1920×1080 image), whereas the inner loop is unrolled for parallel 8-pixel processing aligned with the XF_NPPC8 data path. This structure supports both single- and three-channel formats. By integrating '#pragma HLS UNROLL', 'PIPELINE', and 'dataflow', the design exploits spatial and temporal parallelism—duplicating operations to reduce iterations, overlapping them to increase throughput, and enabling stage-level concurrency—resulting in high performance within tight resource constraints.

IV. RESULTS AND DISCUSSION

As shown in Figure 4, the image processing pipeline is realized as five discrete cores. These cores were generated with Vitis HLS, which converted the C-based algorithms into optimized RTL. The first core, minMat, computes the minimum of the three RGB channels for every pixel and writes that single channel result to memory. Its output feeds darkChannel, which simply multiplies each pixel value by 0.9 to form the dark channel image. Both of these cores communicate over AXI4, using AXI4 master ports for image data and AXI4-Lite slave ports for control and scalar parameters. Next, 'diffim' takes the two previous results, finds the pixel-wise minimum, and scales it by 0.6 to create 'diffim'. Isolating this operation in its own core allows the design to fetch both source images simultaneously without overrunning on-chip memory. The most demanding stage is 'restoreOut'. It first splits an incoming RGB image into its three channels, and then combines each channel with 'diffim' through a loop that contains a division—an operator not natively supported by the Vitis Vision library—before merging the processed channels back together into the 'restoreout' image. One copy of 'restoreout' is output; the other feeds find_lut_u0, which generates the lookup table values. Finally, LUT applies this lookup table to the second copy of 'restoreout'. Because it performs only a look up operation, this core has no internal dataflow section.

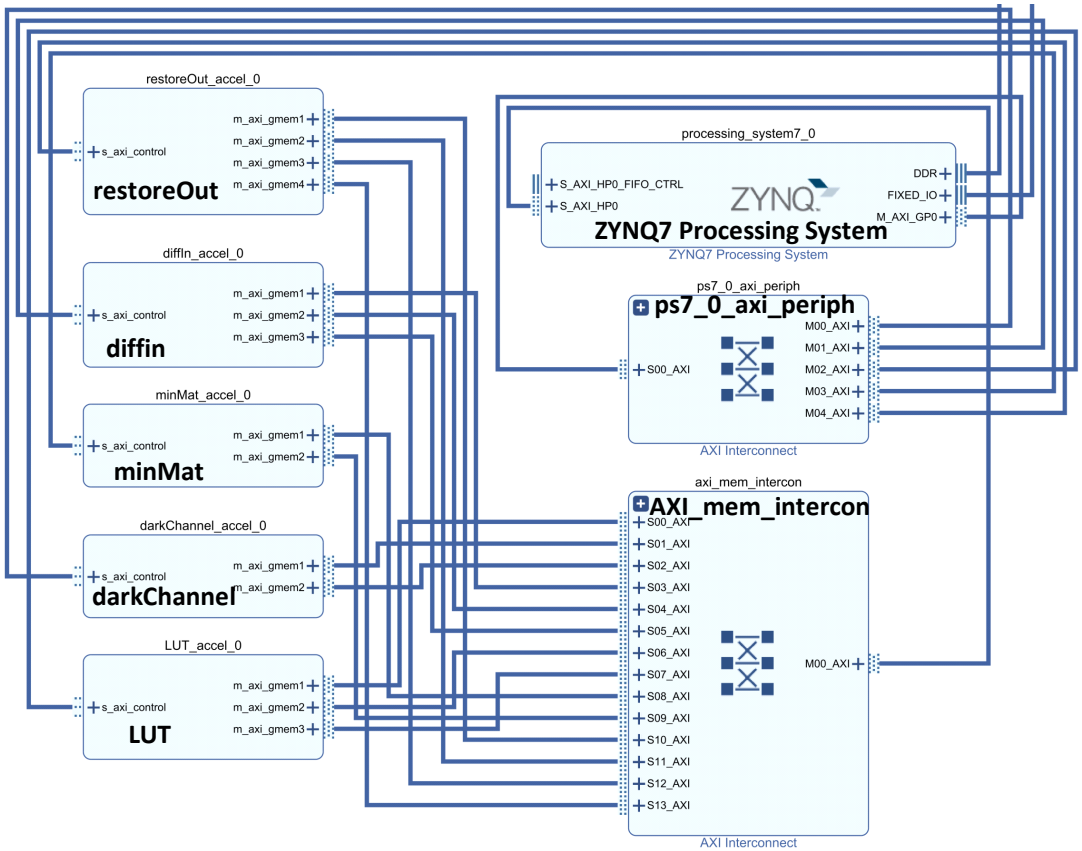

Fig. 4. System block diagram.

Table II highlights how arithmetic precision and datapath width drive FPGA resource usage across four implementation profiles. Profile 1, mirroring the earlier floating-point baseline [18], processes one pixel per clock through a 32-bit bus and therefore consumes the most logic—over half of all available LUTs, nearly a third of flip-flops, a quarter of DSP slices, and a tenth of the on-chip BRAM—leaving only moderate room for expansion. Profile 2 retains the same single-pixel throughput but shifts every computation to fixed-point integers; because fixed-point operations map more efficiently onto the fabric, LUT and flip-flop counts shrink by roughly a third, and DSP demand decreases significantly to just a few percent of the device, easing routing and timing closure while freeing significant headroom. To raise frame rate, the new designs in this study broaden the datapath: Profile 3 widens it eight-fold (processing eight pixels per cycle) yet keeps floating-point arithmetic, which nearly doubles the LUT tally and multiplies DSP and BRAM usage, pushing LUT utilization into the high 80% range and leaving virtually no slack for debugging logic or future features. Profile 4 preserves that eight-pixel-wide pipeline but converts to fixed-point; this recovers a small slice of resources—about a thousand LUTs and several DSP blocks—yet overall utilization still exceeds four-fifths of the LUTs and nearly half of the flip-flops, signaling that further functionality or aggressive timing optimizations would likely require a larger FPGA or major architectural changes.

TABLE II. SYSTEM RESOURCE UTILIZATION

Metric	Profile 1 [18]	Profile 2	Profile 3	Profile 4 (this study)
NPPC	1	1	8	8
Bus width	32-bit	32-bit	256-bit	256-bit
Datatype	Floating	Integer	Floating	Integer
LUT (# / %)	27568 / 52	20249 / 38	45982 / 86	44908 / 85
FF (# / %)	32762 / 31	22772 / 21	52048 / 49	50072 / 47
BRAM (# / %)	16 / 11	16 / 11	62 / 44	62 / 44
DSP (# / %)	55 / 25	9 / 4	47 / 22	44 / 20

Profile 4 is the configuration with an eight-pixel wide data path and uses fixed-point arithmetic. Aggressive parallelization is the main driver of the 85% LUT usage, as processing eight pixels per cycle leads to an eightfold replication of logic, significantly increasing combinatorial resource consumption. In Profile 4, the combined logic for the diffusion and restoration stages dominated LUT usage, as these stages implement complex pixel-wise operations replicated across eight parallel pipelines. Moreover, Profile 4 is slightly more efficient than Profile 3, freeing up about 1,000 LUTs and a few DSP blocks. This is because it uses simpler fixed-point calculations instead of the more complex floating-point operations found in Profile 3. To address this in future iterations or more complex designs, one could either adopt an FPGA with greater logic capacity or refine the algorithm's level of parallelism. For instance, Profile 4 might be better suited for a device with more available LUTs, or its parallel processing could be scaled down to four pixels per cycle to ensure resource usage remains within safer limits.

Figure 5 compares haze removal across four hardware profiles on 1920×1080 images. All configurations successfully eliminate fog, but only Profile 2, which replaces floating-point with fixed-point arithmetic, preserves sharpness and brightness, delivering high-contrast, visually preferred results. Profile 1 removes haze but dims the image due to uncorrected luminance loss. Profiles 3 and 4 improve speed via pixel-level parallelism but inherit the brightness deficit of Profile 1. FPGA outputs remain slightly darker, highlighting a need for future tuning. While the FPGA uses 8-bit grayscale and 32-bit RGB (with padding), software baselines use 32-bit floating point per channel—yielding higher precision but at greater cost. This tradeoff is deemed acceptable in [19], given the significant gains in efficiency.

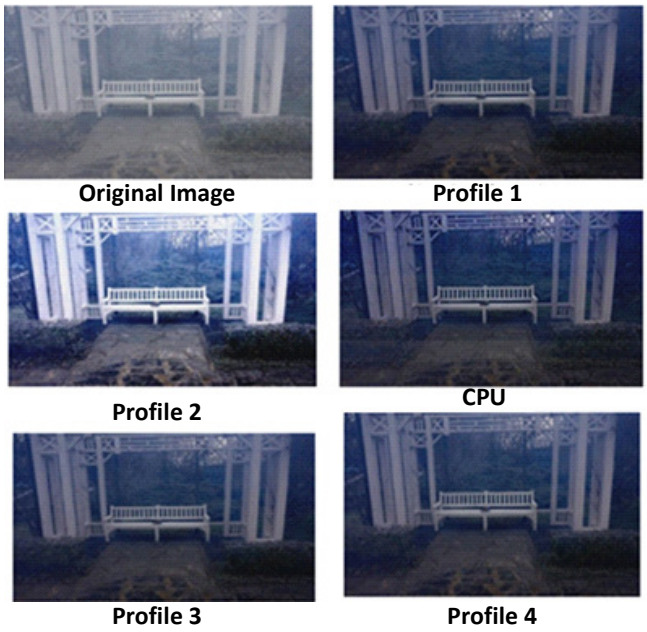

Fig. 5. Defogged images across different configurations.

Table III presents a comparative analysis of system performance and power consumption under four configurations. The single-pixel approaches in Profiles 1 and 2 yield the lowest throughputs, registering 3.66 FPS and 2.47 FPS, respectively. Notably, replacing floating-point arithmetic with integer operations in Profile 2 unexpectedly reduced performance relative to the floating-point baseline in Profile 1. In contrast, the multi-pixel parallelism introduced in Profiles 3 and 4 substantially elevates throughput to 11.44 FPS and 12.06 FPS—almost four times faster than the single-pixel configurations. Converting floating-point operations to integer arithmetic from Profile 3 to Profile 4 delivers a further, albeit modest, performance improvement. However, the implementation of multiple-pixel parallelism also increases power consumption, with these configurations consuming nearly twice the power of the single-pixel configurations.

In addition to the four profile experiments, three CPU-only baselines highlight the limitations of conventional processors. A single thread on an Intel Core i7-9750H (4.3 GHz) achieves 3.62 FPS at 20 W (0.18 FPS/W), closely matching Profile 1

and confirming the single-pixel floating-point version is compute-bound. Using all eight threads increases throughput to 6.57 FPS—only a 1.8× gain instead of 8×—due to memory and cache bottlenecks, though energy efficiency improves slightly to 0.27 FPS/W. In contrast, a 2 GHz ARM CPU consumes just 2.82 W but delivers only 0.28 FPS (0.10 FPS/W), illustrating its limited suitability for floating-point, memory-bound tasks without hardware acceleration. Meanwhile, Profile 4 on FPGA achieves 15.27 FPS/W—153× higher than ARM—demonstrating the FPGA's superior energy efficiency and performance, making it ideal for real-time, power-constrained applications.

TABLE III. COMPARISON OF PERFORMANCE AND POWER CONSUMPTION ACROSS HARDWARE CONFIGURATIONS

Hardware platforms	Performance (FPS)	Power (W)	FPS/W
CPU i7 9750H (1 thread, 4.3 GHz)	3.62	20	0.18
CPU i7 9750H (8 threads, 4.3 GHz)	6.57	24	0.27
CPU ARM 2 GHz	0.28	2.82	0.1
FPGA XCKU060 300 MHz [10]	48.2	2.64	18.3
FPGA PYNQ-Z1 200 MHz [20]	3.42	1.69	2.02
FPGA PYNQ-Z2 100 MHz Profile 1	3.66	0.4	9.15
FPGA PYNQ-Z2 100 MHz Profile 2	2.47	0.34	7.26
FPGA PYNQ-Z2 100 MHz Profile 3	11.44	0.81	14.1
FPGA PYNQ-Z2 100 MHz Profile 4	12.06	0.79	15.27

PSNR and SSIM metrics assess image quality in integer-based accelerators, showing how performance-focused designs impact fidelity. As shown in Figure 6, Profiles 1, 3, and 4 achieved 28.4–29 dB PSNR and 0.69–0.72 SSIM, outperforming the ARM CPU baseline (25.2 dB, 0.68). Profile 2 lagged significantly (12.2 dB, 0.61), likely due to misconfiguration or resource constraints. The accelerator's average pixel error was 4.7%, mainly from 8-bit fixed-point quantization and two algorithmic tweaks: a tuned diffusion coefficient ($\Omega = 0.6$) and a custom brightness restoration step. These introduce minor brightness deviations but maintain scene realism. To improve consistency and accuracy, we propose aligning the 8-bit data format across FPGA and ARM platforms and re-tuning Ω and LUT parameters to reduce quantization errors and enhance cross-platform fidelity.

This study benchmarked the system against two representative FPGA designs from the literature. The first is the 200 MHz PYNQ-Z1 engine reported in [20], a convolution-only accelerator that delivers 3.42 FPS at 1.69 W (≈2.0 FPS/W). The second is the high-end XCKU060 implementation of HDSuper [10]. After normalizing its 100 MPixel/s throughput for a 1920×1080 resolution (approximately 2.07 MPixels/frame), the equivalent performance is 48.2 FPS while consuming 2.64 W, yielding an efficiency of 18.3 FPS/W. Our own design, evaluated on a low-cost PYNQ-Z2 board, attains up to 12.06 FPS at only 0.79 W, translating to 15.27 FPS/W. This is 7.6× the energy efficiency of the widely cited baseline in [20] and within 17% of the much larger, higher-frequency XCKU060 solution—all while using 70% less absolute power. The results indicate that our reconfigurable DSP chain and pruning strategy yield significant advantages on resource-limited FPGAs, facilitating real-time, low-power defogging at the edge.

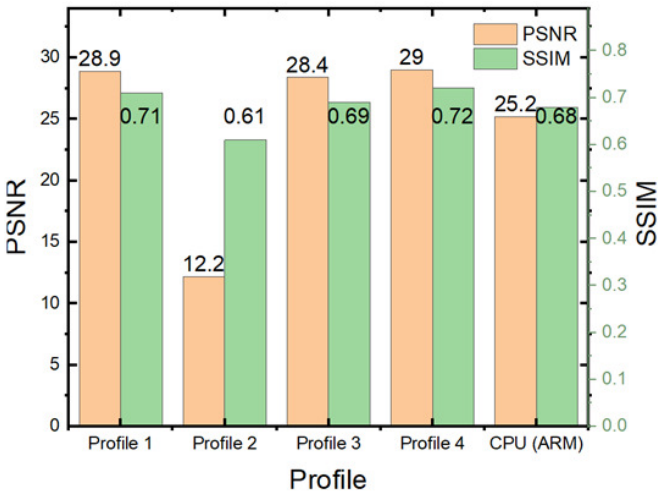

Fig. 6. PSNR and SSIM comparison of image processing quality across FPGA configurations (Profiles 1–4) and ARM CPU.

V. CONCLUSION

This work presents the first sub-200 MHz, resource-balanced Field-Programmable Gate Array (FPGA) pipeline capable of executing the complete Dark Channel Prior (DCP) algorithm on a low-cost PYNQ-Z2 platform using a reconfigurable DSP fabric. Running at just 100 MHz, the design processes 1080p frames at 12.06 Frames per Second (FPS) while consuming only 0.79 W, achieving 15.27 FPS/W—an energy efficiency 153 times higher than a 2 GHz ARM CPU and over seven times greater than recent 200 MHz convolution-only FPGA accelerators, yet without relying on the >200 MHz devices common in top-tier solutions. Despite its modest clock, the accelerator preserves image quality (≈29 dB PSNR, 0.70 SSIM); the remaining 4.7% pixel error stems chiefly from 8-bit quantization and two targeted algorithmic refinements.

The study makes three pivotal contributions: it delivers the first full-frame DCP implementation on a low-cost 100 MHz FPGA, outperforming previous approaches limited to convolution stages or reliant on higher-frequency devices; it provides a comprehensive CPU-versus-FPGA energy assessment that quantifies the substantial power savings attainable on edge-class hardware; and it introduces a brightness-consistency evaluation that directly relates algorithmic simplifications to perceptual image quality under stringent power constraints. Ongoing work will aim to raise FPS/W further through mixed-precision paths, deeper parallelism, and adaptive task partitioning, reinforcing FPGAs as a leading option for low-power, real-time vision at the edge.

ACKNOWLEDGMENT

This work belongs to the project grant No: T2025-87 funded by Ho Chi Minh City University of Technology and Education, Vietnam

REFERENCES

[1] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, pp. 1956–1963, https://doi.org/10.1109/CVPR.2009.5206515.

[2] D. J. Jobson, Z. Rahman, and G. A. Woodell, "A multiscale retinex for bridging the gap between color images and the human observation of scenes," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 965–976, July 1997, https://doi.org/10.1109/83.597272.

[3] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao, "DehazeNet: An End-to-End System for Single Image Haze Removal," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5187–5198, Nov. 2016, https://doi.org/10.1109/TIP.2016.2598681.

[4] B. Li, X. Peng, Z. Wang, J. Xu, and D. Feng, "AOD-Net: All-in-One Dehazing Network," in *2017 IEEE International Conference on Computer Vision*, Venice, Italy, 2017, pp. 4780–4788, https://doi.org/10.1109/ICCV.2017.511.

[5] N. P. Jouppi *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, Toronto, Canada, 2017, pp. 1–12, https://doi.org/10.1145/3079856.3080246.

[6] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, CA, USA, 2014, pp. 10–14, https://doi.org/10.1109/ISSCC.2014.6757323.

[7] W. Ren *et al.*, "Gated Fusion Network for Single Image Dehazing," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 3253–3261, https://doi.org/10.1109/CVPR.2018.00343.

[8] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, USA, 2015, pp. 161–170, https://doi.org/10.1145/2684746.2689060.

[9] T. Saidani, R. Ghodhbani, A. Alhomoud, A. Alshammari, H. Zayani, and M. B. Ammar, "Hardware Acceleration for Object Detection using YOLOv5 Deep Learning Algorithm on Xilinx Zynq FPGA Platform," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 13066–13071, Feb. 2024, https://doi.org/10.48084/etasr.6761.

[10] J. Zhang, D. Fan, and L. Chang, "HIERA: High-Quality and High-Throughput Dehazing Hardware Accelerator with Reconfigurable Computing Unit," in *2024 IEEE Computer Society Annual Symposium on VLSI*, Knoxville, TN, USA, 2024, pp. 75–80, https://doi.org/10.1109/ISVLSI61997.2024.00025.

[11] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-Time Apple Detection System Using Embedded Systems With Hardware Accelerators: An Edge AI Application," *IEEE Access*, vol. 8, pp. 9102–9114, 2020, https://doi.org/10.1109/ACCESS.2020.2964608.

[12] A. Mukhtar, L. Xia, and T. B. Tang, "Vehicle Detection Techniques for Collision Avoidance Systems: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2318–2338, Oct. 2015, https://doi.org/10.1109/TITS.2015.2409109.

[13] A. K. Tripathi and S. Mukhopadhyay, "Single image fog removal using anisotropic diffusion," *IET Image Processing*, vol. 6, no. 7, pp. 966–975, Oct. 2012, https://doi.org/10.1049/iet-ipr.2011.0472.

[14] G. Meng, Y. Wang, J. Duan, S. Xiang, and C. Pan, "Efficient Image Dehazing with Boundary Constraint and Contextual Regularization," in *2013 IEEE International Conference on Computer Vision*, Sydney, Australia, 2013, pp. 617–624, https://doi.org/10.1109/ICCV.2013.82.

[15] C. Xiao and J. Gan, "Fast image dehazing using guided joint bilateral filter," *The Visual Computer*, vol. 28, no. 6, pp. 713–721, June 2012, https://doi.org/10.1007/s00371-012-0679-y.

[16] "7 Series FPGAs Configurable Logic Block User Guide (UG474)." AMD Technical Information Portal. https://docs.amd.com/r/en-US/ug474_7Series_CLB.

[17] "Vitis High-Level Synthesis User Guide (UG1399)." AMD Technical Information Portal. https://docs.amd.com/r/2023.1-English/ug1399-vitis-hls.

[18] N. Pham, "A Novel Fog-Free Computer Vision using Improved Dark Channel Prior with High Level Synthesis in FPGA Design," M.S. thesis, Dept. of Electrical Engineering, National Formosa University, Taiwan, 2023.

[19] L. Forget, G. Harnisch, R. Keryell, and F. de Dinechin, "A single-source C++20 HLS flow for function evaluation on FPGA and beyond.," in *Proceedings of the 12th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*, Tsukuba, Japan, 2022, pp. 51–58, https://doi.org/10.1145/3535044.3535051.

[20] H. Alaeddine, M. Jihene, and M. Khemaja, "An Efficient Deep Network in Network Architecture for Image Classification on FPGA Accelerator," in *2021 International Conference on Cyberworlds*, Caen, France, 2021, pp. 72–77, https://doi.org/10.1109/CW52790.2021.00018.

AUTHOR PROFILES

Dr. Van-Khoa Pham received his B.S. and M.S.E.E. degrees in Computer Technology and Electronics Engineering from the Ho Chi Minh City University of Technology and Education, Vietnam, in 2010 and 2014, respectively. In 2019, he earned his Ph.D. in Electronics Engineering from Kookmin University (KMU), Seoul, South Korea. In 2010, he joined the Integrated Circuit Design Research and Education Center (ICDREC), where he contributed to the development of the VN8-01 MCU—the first commercially designed and fabricated microcontroller in Vietnam. From May 2011 to 2021, he served as a faculty member in the Faculty of Electrical and Electronics Engineering at the Ho Chi Minh City University of Technology and Education (HCMUTE). He is currently a senior lecturer in the Department of Computer and Communication Engineering. He serves as the Head of both the Computer Engineering Technology program and the Electronics and Communications Engineering Technology program under the Faculty of International Education at HCMUTE. His research interests include low-power VLSI design, memory design, the IoT-based applications, and AI-based solutions. He has published numerous research papers in reputable journals and conferences. https://orcid.org/0000-0002-6129-5856

Tin-Le Trung obtained his Bachelor of Engineering degree from the Ho Chi Minh City University of Technology and Education in 2022. He is currently pursuing a Master's degree in Electronics Engineering at the same university. He is working as an NPI and Package Test Development Engineer at OnSemi, where he has developed strong expertise in device modules. His research interests include FPGA acceleration, RTL design, and neural network models.