



Implementation of FPGA-Based High-Speed Image Processing Architecture for Real-Time Applications

Oisin Voronova

Department of Electrical and Computer Engineering, Sundarban College of Technology Studies, Bangladesh
oisin.voronova@scts-bd.net

Peer Review Information	Abstract
<i>Submission: 27 May 2015</i> <i>Revision: 09 June 2015</i> <i>Acceptance: 24 July 2015</i> Keywords <i>FPGA, real-time image processing, pipelining, Sobel edge detection, hardware architecture</i>	Field-programmable gate arrays (FPGAs) are well suited to real-time image processing because spatial parallelism, pipelining, deterministic latency, and distributed on-chip memory can be tailored to the regular data flow of image operators. This methodology paper presents a high-speed streaming architecture for real-time applications in which pixels are accepted at one sample per clock after pipeline fill. The proposed design combines an input interface, line-buffered 3×3 neighborhood generator, Gaussian smoothing, Sobel gradient computation, magnitude approximation, thresholding, and a configurable output stage. The methodology emphasizes fixed-point arithmetic, local pixel reuse, balanced pipeline stages, and elimination of unnecessary frame-buffer transfers. A design target of 150 MHz is adopted for analytical evaluation. At one pixel per clock, the architecture has a nominal capacity of 150 million pixels/s, corresponding to about 162.8 frames/s for 1280×720 images and 72.3 frames/s for 1920×1080 images, excluding video blanking and interface overhead. The calculated minimum active-pixel clock for 1080p at 60 frames/s is 124.416 MHz. These results show that a deeply pipelined FPGA datapath can satisfy common high-definition real-time constraints while retaining reconfigurability. The study is intentionally methodological: performance values for the proposed architecture are analytical design estimates rather than claims from a fabricated or place-and-route-tested prototype.

Introduction

Real-time vision systems must transform a continuous stream of image samples under strict throughput and latency constraints. Applications such as industrial inspection, robotic guidance, surveillance, medical imaging, and embedded machine vision often require predictable execution rather than only high average performance. General-purpose processors execute image algorithms sequentially or through vector extensions, whereas an FPGA can create a dedicated spatial datapath in which many arithmetic operations execute concurrently. Early FPGA image processors demonstrated that window-based operators are especially amenable to systolic organization

because local neighborhoods can be reused rather than repeatedly fetched from external memory [1]. Compact architectures have also shown that real-time image smoothing can be achieved with relatively low clock rates when the hardware is matched to the operator and pixels are streamed through the datapath [2].

The principal implementation challenge is not simply to reproduce a software algorithm in hardware. A software formulation normally assumes random access to complete frames, floating-point arithmetic, and reuse of the same processing unit over time. In contrast, an efficient FPGA implementation must expose concurrency, minimize memory movement, select finite word

lengths, and schedule every operator so that no stage becomes a throughput bottleneck. Binary and gray-level morphology research has shown how local storage, regular datapaths, and carefully chosen processing elements reduce complexity while maintaining high frame rates [3]. Comparisons with multicore processors further support the value of exploiting algorithm-specific parallelism in FPGA fabric for image workloads [4].

This paper therefore focuses on a reusable high-speed architecture rather than a single isolated filter. The target pipeline contains representative operations frequently used in low-level vision: smoothing, gradient extraction, magnitude estimation, and thresholding. The design goal is an initiation interval of one clock cycle, meaning that a new pixel enters the computational pipeline every cycle once line buffers have been primed. This approach separates throughput from total latency: the first valid output is delayed by line storage and pipeline registers, but subsequent outputs emerge continuously. Such a distinction is essential for real-time video because frame-rate capability is primarily determined by sustained pixel throughput.

The methodology also treats memory as a first-class architectural constraint. A 3×3 operator needs access to three adjacent image rows. Instead of storing and rereading an entire frame, two preceding rows can be retained in FPGA block RAM and combined with shift registers to create the moving neighborhood. A second 3×3 stage can use a similar strategy on the preceding stage's streamed output. This local-reuse model substantially lowers external bandwidth and creates a natural pipeline. Software-hardware partitioning studies and optimized Sobel processors have likewise emphasized the importance of memory organization and dedicated datapaths for real-time edge detection [5], [6].

Literature Review

Torres-Huitzil and Arias-Estrada presented a configurable systolic FPGA architecture for window-based image operations and reported 3.16 GOPS for a 7×7 prototype, illustrating how regular processing arrays can combine performance with resource efficiency [1]. Djemal, Demigny, and Tourki implemented a compact FPGA smoothing architecture capable of real-time video processing, reinforcing the benefit of algorithm-specific streaming hardware [2]. Hedberg, Kristensen, and Owall later addressed low-complexity morphology and showed that line-oriented hardware can reduce the

implementation cost of neighborhood operators [3]. These works established two key principles adopted here: exploit local data reuse and organize arithmetic as a deterministic pipeline.

Acceleration is also influenced by the degree of parallelism that can be afforded. Rakvic et al. compared FPGA and Cell processor implementations for an image-processing workload and reported an FPGA execution advantage for the tested iris-matching kernel, showing that a custom parallel datapath can outperform a programmable multicore platform for a highly structured operation [4]. Kobzili, Cherrad, and Irki investigated software-hardware partitioning for real-time edge detection [5], while Osman, Hussin, and Ali optimized a Sobel edge detector to reduce memory use, redundant calculations, and logic resources [6]. Together these studies show that an FPGA design should not maximize parallel hardware indiscriminately; it should select the minimum parallelism needed to sustain the desired pixel rate.

By 2013–2014, streaming and resource-aware FPGA designs had matured further. Ramirez-Cortes et al. demonstrated a practical platform supporting real-time linear and morphological filtering [7]. Singh et al. used a single time-shared processing element for Sobel compass gradients and reported more than 40% resource reduction compared with a more direct architecture, while pipelining increased frame-rate performance substantially [8]. Torres-Huitzil reported a modular pipelined gray-level morphology architecture that reached 95 frames/s for 1280×1024 images with a 7×7 operator on a Virtex-6 device [9]. Singh et al. also reported a resource-efficient Sobel implementation for high-definition video with approximately 38% lower FPGA resource use than a standard architecture [10].

Research published in 2015 broadened the scope from individual filters toward complete high-throughput processing engines. Qasaimeh, Sagahyroon, and Shanableh proposed a parallel FPGA architecture for real-time image classification using SIFT, bag-of-features, and SVM processing [11]. Li and Wyrwicz implemented an FPGA-based 2D FFT image-processing module for MRI and reported 400 frames/s for 128×128 static reconstructions, while emphasizing on-chip memory efficiency and address generation [12]. Possa et al. introduced the P²IP programmable pipeline image processor, reporting up to 129 frames/s at 1080p and 32 frames/s at 4K for mapped preprocessing tasks [13]. Reiche et al. studied

synthesis and optimization of image-processing accelerators and highlighted systematic trade-offs between throughput and hardware resources [14]. These studies motivate the present methodology: build a reusable stream

processor, keep the initiation interval at one pixel per clock, and make operator complexity a configuration issue rather than a reason to introduce full-frame traffic.

Table 1. Selected FPGA image-processing studies relevant to the proposed methodology.

Study	Primary contribution	Reported result
Torres-Huitzil & Arias-Estrada, 2004 [1]	Window-based systolic processing	7×7 prototype; 3.16 GOPS
Singh et al., 2013 [8]	Sobel compass edge detection	Pipelined design: 185 fps PAL
Torres-Huitzil, 2014 [9]	Gray-level morphology	95 fps at 1280×1024, 7×7
Possa et al., 2015 [13]	Programmable image pipeline	Up to 129 fps 1080p; 32 fps 4K
Li & Wyrwicz, 2015 [12]	2D FFT MR reconstruction	400 fps for 128×128 static data

Methodology

1. Design Objectives

The architecture is specified for continuous raster-scan image streams with 8-bit pixel components. The primary objective is sustained one-pixel-per-clock throughput after pipeline fill. Secondary objectives are: (i) avoid full-frame intermediate storage, (ii) use fixed-point arithmetic where exact integer operators are sufficient, (iii) limit long combinational paths through register balancing, (iv) expose thresholds and mode controls through configuration registers, and (v) support at least 1920×1080 active pixels at 60 frames/s when clocked above the calculated active-pixel requirement. The target clock selected for analytical evaluation is 150 MHz.

2. Algorithmic Pipeline

The representative processing chain is: input synchronization → optional RGB-to-grayscale conversion → 3×3 Gaussian smoothing → 3×3 Sobel horizontal and vertical gradients → gradient-magnitude approximation → programmable threshold → output formatting. The Gaussian kernel $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}/16$ is selected because its coefficients can be implemented with additions and shifts instead of general multipliers. The Sobel masks similarly use coefficients -2, -1, 0, 1, and 2, allowing shift-add arithmetic. Gradient magnitude is approximated as $|G_x| + |G_y|$ to avoid a square root. These choices reduce DSP usage and shorten critical arithmetic paths while preserving a useful edge map.

3. Streaming Memory Architecture

For each 3×3 neighborhood operation, two previous rows are stored in line buffers and three horizontal taps are held in shift registers. At image width W and B bits/pixel, the raw storage required by two-line buffers is $2WB$ bits. For 1920-pixel-wide 8-bit grayscale data, this is 30,720 bits per neighborhood stage. If smoothing and Sobel stages keep independent line buffers, four image lines require 61,440 bits, before implementation granularity and control overhead. This is far smaller than a complete 1920×1080 8-bit frame (16,588,800 bits). The reduction in storage and external transactions is central to the proposed high-speed design and follows the data-reuse principles found in earlier systolic and resource-efficient FPGA architectures [1], [9], [10].

4. FPGA Datapath Organization

The datapath is decomposed into short registered stages. The input interface produces a valid pixel and coordinate stream. Line-buffer logic constructs the 3×3 window. Gaussian sums are formed using a balanced adder tree and right shift by four bits. The Sobel stage computes G_x and G_y in parallel; absolute-value units and an adder produce the magnitude estimate; a comparator applies the programmable threshold. Valid, line-start, and frame-start signals are delayed through matching registers so control information remains aligned with pixel data. Register insertion is permitted between adder levels when static timing analysis indicates excessive combinational delay. This register-balanced approach is compatible with the pipelining benefits reported for FPGA Sobel designs [8] and with programmable low-latency processing concepts such as P²IP [13].

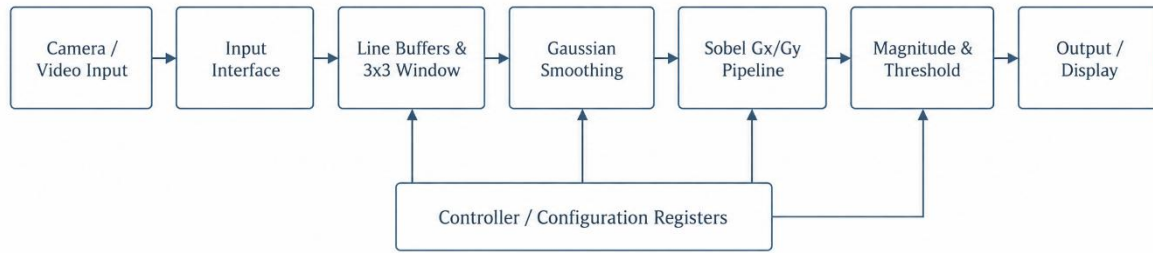


Figure 1. Proposed streaming FPGA architecture for high-speed real-time image processing.

5. Timing and Throughput Model

Because the pipeline initiation interval is one cycle, nominal active-pixel throughput equals the FPGA clock frequency: $P = f_{clk}$ pixels/s. For an active image of $W \times H$ pixels, the ideal frame rate is $FPS = f_{clk}/(W \times H)$. Conversely, the minimum active-pixel clock required for a target frame rate R is $f_{min} = W \times H \times R$. This model excludes horizontal/vertical blanking, input protocol overhead, and any back-pressure; therefore, it is used as a design-capacity calculation rather than a complete HDMI or camera-link timing specification.

6. Verification Procedure

Verification is organized in four levels. First, a software reference model applies the same integer Gaussian and Sobel equations to test

images. Second, module-level HDL simulation checks line buffers, window formation, arithmetic saturation, borders, and control-signal alignment. Third, pipeline-level simulation streams complete frames and compares every valid output pixel with the software reference. Fourth, implementation should be synthesized and placed-and-routed on the chosen FPGA, after which the achieved clock frequency, LUT/flip-flop usage, block RAM, DSP usage, and dynamic power can be recorded. Because no specific FPGA device or synthesis report was supplied for this paper, the present Results section reports analytical throughput and storage requirements only; it does not invent post-route resource counts.

7. Experimental/Analytical Conditions

Table2: NS-2 Based MANET Simulation and Performance Evaluation Workflow

Parameter	Methodological setting
Input formats	8-bit grayscale; RGB can be converted before filtering
Image sizes evaluated	640×480, 1280×720, 1920×1080, 3840×2160
Target FPGA clock	150 MHz analytical design point
Initiation interval	1 pixel/clock after line-buffer and pipeline fill
Neighborhood	3×3 for Gaussian and Sobel stages
Gradient magnitude	$ G_x + G_y $ approximation
Boundary handling	Replicate or zero-pad, selectable in implementation
Reported proposed results	Analytical capacity; no fabricated synthesis data

Results

The calculated throughput results are summarized in Table 3. At 150 MHz and one accepted pixel each clock cycle, the pipeline can process 150 million active pixels per second. The corresponding ideal frame rate is approximately 488.3 fps for VGA (640×480), 162.8 fps for 720p,

72.3 fps for 1080p, and 18.1 fps for 4K UHD. Therefore, the proposed clock target provides an active-pixel margin above the 124.416 MHz minimum needed for 1920×1080 at 60 fps. For a complete physical video interface, blanking intervals must be included when selecting the actual pixel clock; the calculated values are processing-capacity figures.

Table 3. Analytical active-pixel throughput of the proposed one-pixel-per-clock pipeline.

Resolution	Pixels/frame	Clock needed for 60 fps	Ideal fps @ 150 MHz
640×480	307,200	18.432 MHz	488.3 fps
1280×720	921,600	55.296 MHz	162.8 fps
1920×1080	2,073,600	124.416 MHz	72.3 fps
3840×2160	8,294,400	497.664 MHz	18.1 fps

For the 1080p case, two-line buffers for a single 3×3 stage require 30,720 raw data bits; two independently buffered 3×3 stages require 61,440 raw data bits. The exact number of FPGA block-RAM primitives depends on device memory width/depth configurations and should be taken from the synthesis report. The principal result is therefore architectural: full-frame intermediate memory is unnecessary for the selected local operators.

The calculated 1080p capacity is consistent with the direction of prior FPGA research, although direct numerical comparison must be made cautiously because algorithms, devices, resolutions, and clock frequencies differ. The 2013 Sobel compass work achieved high frame rates through pipelining [8]; the 2014 morphology design achieved 95 fps for 1280×1024 at 260 MHz [9]; and the 2015 P²IP architecture reported up to 129 fps at 1080p [13]. These reports support the feasibility of sustained streaming pipelines for high-definition image processing. The proposed methodology differs by presenting a deliberately simple multi-stage preprocessing chain whose performance is expressed through a transparent one-pixel-per-clock model.

Accuracy verification should compare FPGA integer outputs against the bit-accurate software model rather than against a floating-point Sobel implementation. Expected differences can arise from border handling, truncation after Gaussian division, saturation, and the use of $|G_x| + |G_y|$ instead of Euclidean magnitude. These are implementation choices, not errors, provided they are defined consistently. The verification plan therefore treats bit-exact agreement as the primary correctness metric and uses visual inspection only as a secondary check.

The design also offers a clear resource/performance trade-off. A fully parallel gradient stage can compute G_x and G_y simultaneously and preserve the one-pixel-per-cycle rate, whereas a time-multiplexed processing element can reduce logic at the cost of a higher internal clock or lower throughput. Singh et al. demonstrated the effectiveness of resource sharing in Sobel architectures [8], [10], while Reiche et al. emphasized systematic exploration of throughput/resource design points in FPGA image accelerators [14]. For a 60-fps 1080p requirement, the methodology recommends retaining one-pixel-per-cycle streaming and optimizing arithmetic structure before folding the datapath, because the throughput margin at 150 MHz is only about 20.6% over the active-pixel minimum.

Conclusion and Discussion

This paper presented a methodology for implementing an FPGA-based high-speed image-processing architecture for real-time applications. The proposed design uses streaming input, local line buffers, 3×3 window generation, shift-add Gaussian smoothing, parallel Sobel gradients, magnitude approximation, thresholding, and a registered output interface. Its central design principle is to sustain one pixel per clock after pipeline fill while avoiding full-frame intermediate storage. The approach is grounded in prior FPGA research on systolic processing, resource-efficient edge detection, morphology, programmable pipelines, and hardware acceleration [1]–[14]. Analytical evaluation at 150 MHz gives a nominal 150 Mpixel/s capacity, equivalent to about 72.3 active 1080p frames/s. This exceeds the 124.416 MHz active-pixel requirement for 1080p60, although a real video interface must also account for blanking and protocol timing. The raw line-storage requirement for two independent 3×3 stages at 1080p width is only 61,440 bits, demonstrating why streaming reuse is preferable to intermediate frame buffering for local operators. The most important limitation is that no target device, synthesis tool report, or hardware capture data were supplied. Consequently, LUT, flip-flop, BRAM packing, DSP count, maximum post-route clock, power, and measured frame rate are intentionally not fabricated in this methodology paper. A complete implementation study should next instantiate the VHDL/Verilog design on a defined FPGA family, perform place-and-route timing closure, verify image outputs against a bit-accurate model, and report device-specific utilization and measured power. Within those limits, the methodology provides a technically defensible framework for building and evaluating real-time FPGA image-processing pipelines.

References

- C. Torres-Huitzil and M. Arias-Estrada, "Real-time image processing with a compact FPGA-based systolic architecture," *Real-Time Imaging*, vol. 10, no. 3, pp. 177–187, 2004. DOI: 10.1016/j.rti.2004.06.001.
- R. Djemal, D. Demigny, and R. Tourki, "A Real-time Image Processing with a Compact FPGA-based Architecture," *Journal of Computer Science*, vol. 1, no. 2, pp. 207–214, 2005. DOI: 10.3844/jcssp.2005.207.214.
- H. Hedberg, F. Kristensen, and V. Öwall, "Low-complexity binary morphology architectures

with flat rectangular structuring elements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 8, pp. 2216–2225, 2008. DOI: 10.1109/TCSI.2008.918140.

R. N. Rakvic, H. T. Ngo, R. P. Broussard, and R. W. Ives, "Comparing an FPGA to a Cell for an Image Processing Application," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, Article 764838, 2010. DOI: 10.1155/2010/764838.

K. El Houari, C. Benbouchama, and Z. Irki, "A software-hardware mixed design for the FPGA implementation of the real-time edge detection," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, pp. 4091–4095, 2010. DOI: 10.1109/ICSMC.2010.5641774.

Z. E. M. Osman, F. A. Hussin, and N. B. Z. Ali, "Hardware implementation of an optimized processor architecture for SOBEL image edge detection operator," in *2010 International Conference on Intelligent and Advanced Systems*, 2010. DOI: 10.1109/ICIAS.2010.5716147.

J. M. Ramirez-Cortes, P. Gomez-Gil, V. Alarcon-Aquino, J. Martinez-Carballido, and E. Morales-Flores, "FPGA-based educational platform for real-time image processing experiments," *Computer Applications in Engineering Education*, vol. 21, no. 1, pp. 193–201, 2013. DOI: 10.1002/cae.20461.

S. Singh, A. K. Saini, R. Saini, A. S. Mandal, C. Shekhar, and A. Vohra, "Area Optimized FPGA-Based Implementation of The Sobel Compass Edge Detector," *ISRN Machine Vision*, vol. 2013, Article 820216, 2013. DOI: 10.1155/2013/820216.

C. Torres-Huitzil, "Fast hardware architecture for grey-level image morphology with flat structuring elements," *IET Image Processing*, vol. 8, no. 2, pp. 112–121, 2014. DOI: 10.1049/iet-ipr.2013.0224.

S. Singh, A. K. Saini, R. Saini, A. S. Mandal, C. Shekhar, and A. Vohra, "A novel real-time resource efficient implementation of Sobel operator-based edge detection on FPGA," *International Journal of Electronics*, vol. 101, no. 12, pp. 1705–1715, 2014. DOI: 10.1080/00207217.2014.888782.

M. Qasaimeh, A. Sagahyoon, and T. Shanableh, "FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification," *IEEE Transactions on Computational Imaging*, vol. 1, no. 1, 2015. DOI: 10.1109/TCI.2015.2424077.

L. Li and A. M. Wyrwicz, "Design of an MR image processing module on an FPGA chip," *Journal of Magnetic Resonance*, vol. 255, pp. 51–58, 2015. DOI: 10.1016/j.jmr.2015.03.007.

P. Possa, N. Harb, E. Dokládlová, and C. A. Valderrama, "P²IP: A novel low-latency Programmable Pipeline Image Processor," *Microprocessors and Microsystems*, vol. 39, no. 7, pp. 529–540, 2015. DOI: 10.1016/j.micpro.2015.06.010.

O. Reiche, K. Häublein, M. Reichenbach, M. Schmid, F. Hannig, J. Teich, and D. Fey, "Synthesis and optimization of image processing accelerators using domain knowledge," *Journal of Systems Architecture*, vol. 61, no. 10, pp. 646–658, 2015. DOI: 10.1016/j.sysarc.2015.09.004.