FPGA-Based Implementation of Gaussian Filtering for HDR Image Tone-Mapping

Akash Ranjan Sahu (24614), Girish R (24878), Mohammad Farukh Zafar (25047), Sourish Vijayamadhavan (24140), DESE, IISc

ABSTRACT

High Dynamic Range (HDR) images provide superior visual quality with greater contrast compared to Standard Dynamic Range (SDR) images. However, displaying HDR content on SDR devices requires tone-mapping algorithms to preserve the visual quality while mapping the high dynamic range to a lower one. This paper presents an FPGA implementation of a tone-mapping algorithm to reduce computational burden on processors. We focus specifically on implementing the Gaussian filtering component of the algorithm, which is the most computationally intensive step, on an FPGA platform. The implemented system achieves effective tone-mapping between HDR and SDR domains with a TMQI score of 0.79, indicating high-quality tone-mapping results. Our design incorporates UART-based communication with the host processor and implements an efficient line buffer approach for processing high-resolution images despite limited on-chip memory.

I INTRODUCTION AND MOTIVATION

High Dynamic Range (HDR) images have revolutionized digital imaging by significantly increasing the contrast ratio compared to traditional Standard Dynamic Range (SDR) images. This enhanced contrast is achieved through the use of 32-bit floating-point representation per color channel in HDR, as opposed to the standard 8-bit integer representation used in SDR [1]. The expanded dynamic range allows HDR images to capture and display a wider range of luminosity levels, resulting in more realistic and visually appealing images that better represent what the human eye perceives in real-world scenes.

However, this technological advancement introduces challenges when displaying HDR content on devices that only support SDR. Simply truncating the HDR values to fit within the SDR range leads to loss of detail and visual artifacts, particularly in very bright or dark regions of the image. This issue has led to the development of tone-mapping algorithms, which intelligently compress the dynamic range of HDR images while preserving the essential visual characteristics and contrast relationships in the original content [2].

The motivation for implementing tone-mapping on an FPGA stems from the computationally intensive nature of these algorithms. As shown in Fig. 1, the visual difference between properly tone-mapped HDR content and non-tone-mapped content on SDR displays is significant. Most display devices supporting only SDR implement tone-mapping algorithms in software, which can be resource-intensive for the host processor. By offloading this computation to dedicated hardware like FPGAs, we can achieve better performance while freeing up the main processor for other tasks.

Figure 1: Comparison of HDR image displayed on SDR device (left) versus HDR device (right)

This paper focuses specifically on implementing the Gaussian filtering component of the tone-mapping algorithm on an FPGA, as this is typically the most computationally expensive step in the process. Our implementation demonstrates the feasibility and benefits of hardware acceleration for image processing tasks, particularly for applications requiring real-time processing of high-resolution HDR content.

II BACKGROUND STUDY

2.1 Tone-Mapping Algorithms

Tone-mapping algorithms aim to preserve the visual appearance of HDR images when displayed on devices with limited dynamic range. While many algorithms have been proposed in the literature, they generally fall into two categories: global operators and local operators [2].

Global operators apply the same transformation to all pixels in the image based on global statistics such as the average luminance or histogram. These operators are computationally efficient but may result in loss of local contrast. Local operators, on the other hand, adapt the transformation based on the local neighborhood of each pixel, preserving local contrast at the expense of increased computational complexity.

The tone-mapping algorithm implemented in this paper follows a local operator approach based on layer decomposition and image fusion as described by Han and Rahardja [1]. This method decomposes the HDR image into base and detail layers, processes them separately, and then recombines them to produce the final tone-mapped image.

2.2 FPGA-Based Image Processing

Field-Programmable Gate Arrays (FPGAs) offer several advantages for image processing applications, including parallelism, reconfigurability, and dedicated hardware resources for specific operations. Previous work has demonstrated the effectiveness of FPGAs for various image processing tasks, such as filtering, edge detection, and compression [3].

For HDR imaging specifically, FPGAs have been used to implement real-time tone-mapping operators [4]. However,

most implementations focus on simple global operators due to the resource constraints of FPGAs. Our work extends these efforts by implementing a more sophisticated local operator that provides better visual quality.

III DESIGN SPACE EXPLORATION

3.1 Tone-Mapping Algorithm Overview

The tone-mapping algorithm implemented in this work consists of the following steps, as illustrated in Fig. 2:

1. Conversion of the RGB image to HSV (Hue-Saturation-Value) color space
2. Logarithmic compression of the V channel
3. Gaussian filtering of the compressed V channel to obtain the base layer (implemented on FPGA)
4. Subtraction of the base layer from the compressed V channel to obtain the detail layer
5. Thresholding to generate three virtual images from the base layer
6. Multi-resolution fusion of the virtual images
7. Contrast stretching on the detail layer
8. Downscaling of saturation and conversion of HSV back to RGB

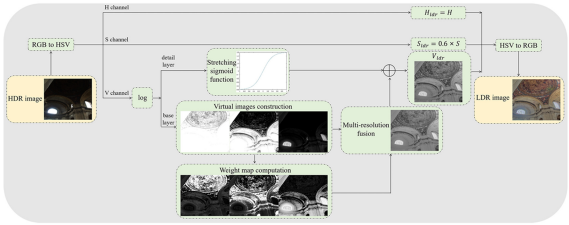

Figure 2: Block diagram of the tone-mapping algorithm

Among these steps, the Gaussian filtering operation is the most computationally expensive, making it an ideal candidate for hardware acceleration using an FPGA.

3.2 FPGA Architecture

The hardware implemented on the FPGA consists of two main modules, as shown in Fig. 3:

1. UART Transceiver Module: Handles communication with the host processor for receiving image data and sending back processed results
2. Convolution Module: Performs 2D Gaussian filtering on the received image data

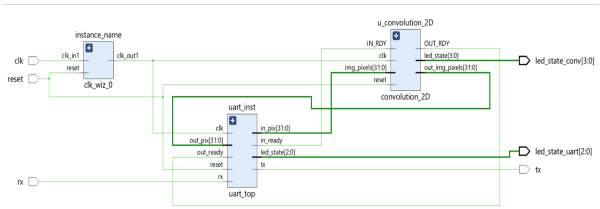

Figure 3: Block diagram of the FPGA architecture

The design uses a finite state machine (FSM) approach for both modules to manage the data flow and processing operations. The convolution module implements a 5×5 Gaussian filter kernel, which requires storing multiple rows of the image in on-chip memory.

3.3 Memory Management Strategy

Given the limited memory resources available on the FPGA, efficiently managing image data is crucial. For a 256×256 image with 8-bit pixel values, the total memory requirement is 64 KB, which exceeds the capacity of block RAM (BRAM) on many low-cost FPGAs.

To address this limitation, we adopted a line buffer approach where only a small portion of the image (5 rows) is stored on the FPGA at any given time. This approach allows processing the image row by row while minimizing memory usage.

The algorithm for processing the image using the line buffer approach is as follows:

Algorithm 1 Line Buffer Processing

```
Initialize line buffer with first 5 rows of image
for each row i from 1 to image height - 4 do
    Process line buffer to generate output row i
    Send output row i to host
    if i < image height - 4 then
        Receive next row from host
        Shift line buffer (discard oldest row)
        Add new row to line buffer
    end if
end for
```

This approach significantly reduces the memory requirements while maintaining the ability to perform the 2D convolution operation correctly.

IV IMPLEMENTATION CHALLENGES

4.1 Convolution Implementation

The implementation of the 2D convolution operation on the FPGA presented several challenges. The convolution finite state machine (FSM), shown in Fig. 4, manages the process of fetching data from the line buffer, performing the convolution operation, and storing the results.

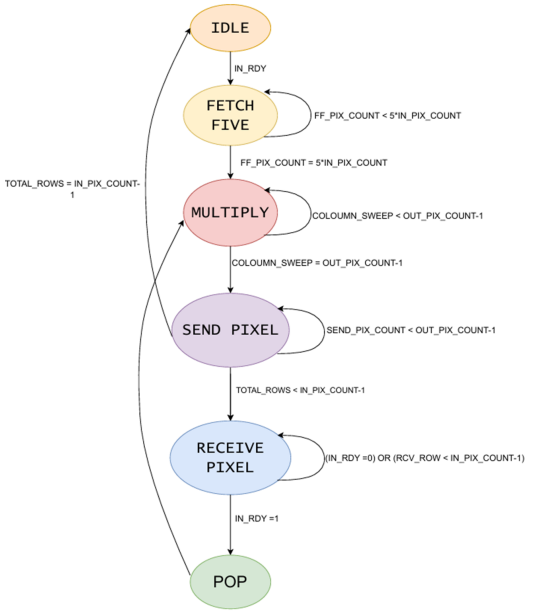

Figure 4: State diagram of the convolution FSM

The convolution process involves the following states:

- IDLE: Waiting for input data to be ready
- FETCH_FIVE: Loading 5×5 pixel neighborhood from line buffer
- MULTIPLY: Performing multiplication and accumulation operations
- SEND_PIXEL: Sending processed pixel to output buffer
- RECEIVE_PIXEL: Receiving new pixel data from UART module
- POP: Shifting line buffer to make room for new row

One of the major challenges was implementing the 5×5 filter kernel efficiently. To reduce resource usage, we implemented the kernel with fixed-point arithmetic rather than floating-point, with appropriate scaling to maintain precision.

4.2 *Communication Interface*

The UART interface, implemented as shown in Fig. 5, manages the bidirectional communication between the FPGA and the host processor.

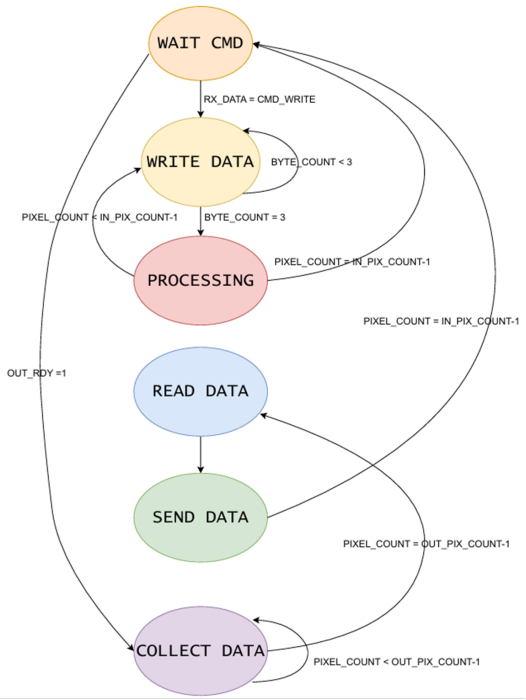

Figure 5: State diagram of the UART transceiver FSM

The UART module operates with the following states:

- WAIT_CMD: Waiting for command from host
- WRITE_DATA: Receiving pixel data from host
- PROCESSING: Signaling the convolution module to process data
- READ_DATA: Reading processed data from convolution module
- SEND_DATA: Sending processed data back to host
- COLLECT_DATA: Collecting multiple bytes for transmission

The UART transceiver was configured to operate at 115,200 baud, which was determined to be the optimal speed for reliable communication with the host processor. Higher baud rates resulted in sampling errors and data corruption, while lower rates would unnecessarily slow down the processing.

4.3 *Timing and Synchronization*

Proper timing and synchronization between the UART module and convolution module were critical for correct operation. Hold violations were encountered during initial implementation, particularly in the data path between the two modules. These violations were resolved by inserting BUFG (global buffer) components at strategic points in the design to ensure proper timing closure.

The total combinational delay of the 32-bit floating-point multiplier and adder in the datapath of the convolution module was observed to be more than 50 ns. Hence, in order to meet

the setup time requirements, the clock frequency of the system was reduced to 16 MHz. To operate the system at higher clock frequencies, pipelining can be used within the datapath, however this can result in increased complexity in FSM design.

The reliability of the UART transceiver module was found to be dependent on the clocks-per-bit parameter, which is a ratio of the clock frequency and the baud rate. Having a higher clocks-per-bit parameter meant that the bit can be sampled over a wider number of clocks, hence improving reliability. When downscaling the system clock frequency to 16 MHz while keeping the baud rate at 1.5 Mbps initially, the clocks-per-bit parameter was too low, resulting in unreliable sampling of the bitstream from the PC. This issue was resolved by reducing the baud rate as well to 115200 bps, thereby ensuring reliable sampling.

V RESULTS

5.1 *Image Quality Assessment*

The effectiveness of the implemented tone-mapping algorithm was evaluated using the Tone-Mapped Quality Index (TMQI), which measures the quality of tone-mapped images by considering both structural fidelity and statistical naturalness. The implemented system achieved a TMQI score of 0.79, indicating high-quality tone-mapping results.

Fig. 6 shows a comparison between the original HDR image, the result of the OpenCV software implementation, and the result of our FPGA implementation. The visual differences between the software and hardware implementations are minimal, demonstrating that the FPGA implementation maintains the image quality of the software version.

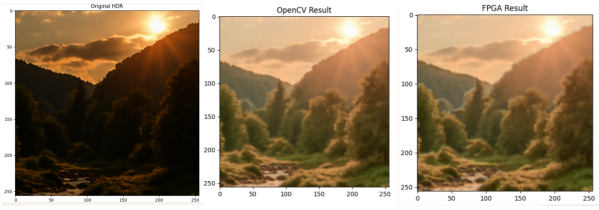

Figure 6: Comparison of original HDR image (left), OpenCV result (center), and FPGA result (right)

Additionally, Fig. 7 shows the difference between the OpenCV and FPGA implementations, highlighting that the discrepancies are minimal and mainly concentrated in high-frequency detail regions.

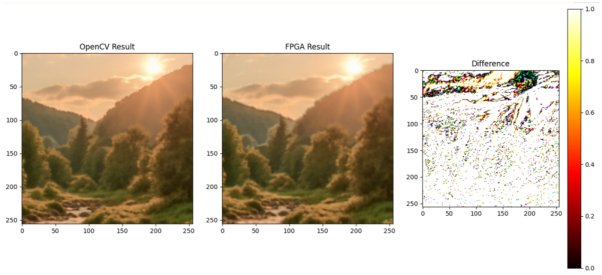

Figure 7: Difference between OpenCV and FPGA results

5.2 *Resource Utilization*

The resource utilization of the implemented design on the Nexys Video FPGA (Xilinx XC7A200T) is summarized in Table 1. The design uses a significant portion of the available Look-Up Tables (LUTs), but is relatively efficient in terms of Flip-Flop (FF) and Digital Signal Processing (DSP) slice usage.

Resource	Utilization	Available	Utilization %
LUT	127726	134600	94.89
FF	66985	269200	24.88
DSP	2	740	0.27
IO	11	285	3.86
MMCM	1	10	10.00

Table 1: Resource utilization of the implemented design

5.3 *Power and Timing Analysis*

The power consumption of the implemented design is 0.402 W, with dynamic power accounting for 65% (0.263 W) and static power accounting for 35% (0.140 W). Within the dynamic power consumption, the Mixed-Mode Clock Manager (MMCM) is the largest contributor at 46% (0.122 W), followed by clocks at 27% (0.070 W).

The timing analysis confirms that all user-specified timing constraints are met, with a worst negative slack (WNS) of 2.956 ns for setup time and 0.013 ns for hold time. The maximum operating frequency was limited to 16 MHz to ensure reliable operation with the UART interface.

VI CONCLUSION

This paper presented an FPGA implementation of the Gaussian filtering component of a tone-mapping algorithm for HDR images. The implemented system successfully achieves high-quality tone-mapping with minimal visual differences compared to a software implementation.

The key contributions of this work include:

- An efficient line buffer approach for processing high-resolution images with limited on-chip memory

- A comprehensive UART-based communication interface for integration with host processors
- Detailed resource utilization and performance analysis of the implemented design

The implemented system demonstrates the feasibility and benefits of hardware acceleration for image processing tasks, particularly for computationally intensive operations like 2D convolution.

6.1 Future Work

Several directions for future work have been identified:

- Implementation of a systolic array architecture for more efficient convolution operations
- Extension of the FPGA implementation to include other components of the tone-mapping algorithm, such as thresholding, sigmoid stretching, and logarithmic compression
- Exploration of higher-speed interfaces, such as PCI Express or Gigabit Ethernet, for faster data transfer between the host and FPGA
- Optimization of the design for lower power consumption, enabling its use in battery-powered devices

These enhancements would further improve the performance and applicability of the FPGA-based tone-mapping system for real-world applications.

REFERENCES

[1] X. Han and S. Rahardja, "High Dynamic Range Image Tone Mapping Based on Layer Decomposition and Image Fusion," 2023 IEEE International Conference on Image Processing (ICIP), Kuala Lumpur, Malaysia, 2023, pp. 221-225.

[2] T. Mertens, J. Kautz and F. V. Reeth, "Exposure Fusion: A Simple and Practical Alternative to High Dynamic Range Photography", Computer Graphics Forum, vol. 28, no. 1, pp. 161-171, Mar. 2009.

[3] D. G. Bailey, "Design for Embedded Image Processing on FPGAs," John Wiley & Sons (Asia) Pte Ltd, 2011.

[4] P. Lapray, B. Heyrman and D. Ginhac, "HDR-ARtiSt: An Adaptive Real-Time Smart Camera for High Dynamic Range Imaging," Journal of Real-Time Image Processing, vol. 12, no. 4, pp. 747-762, 2014.

[5] Xilinx, Inc., "Xilinx 7 Series FPGAs Data Sheet: Overview," DS180 (v2.6), 2018.