FPGA-Based Real-Time Video 2D FFT Accelerator

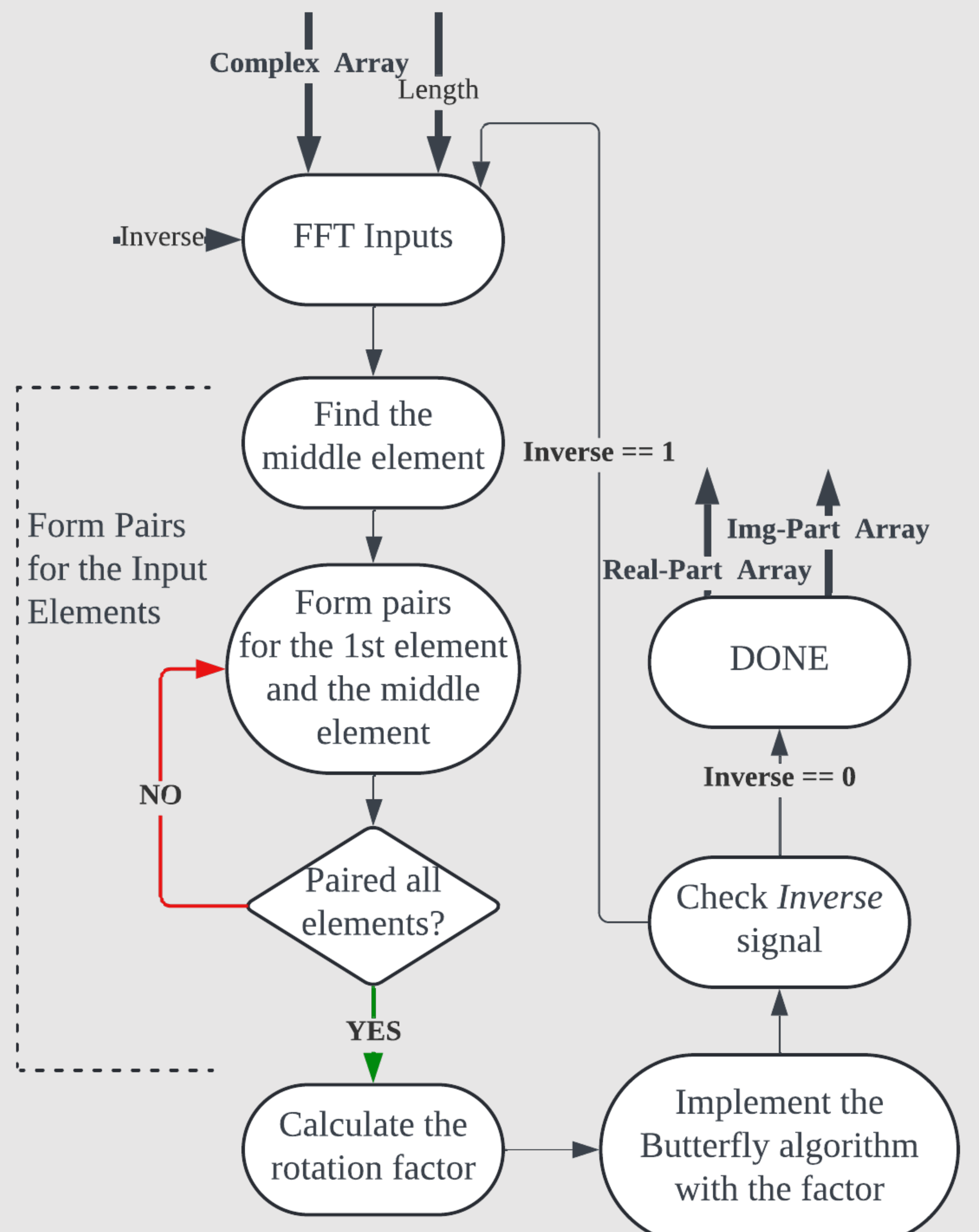
Author(s): Yibin Xu, Ruyi Zhou

Advisor: Hunter Adams

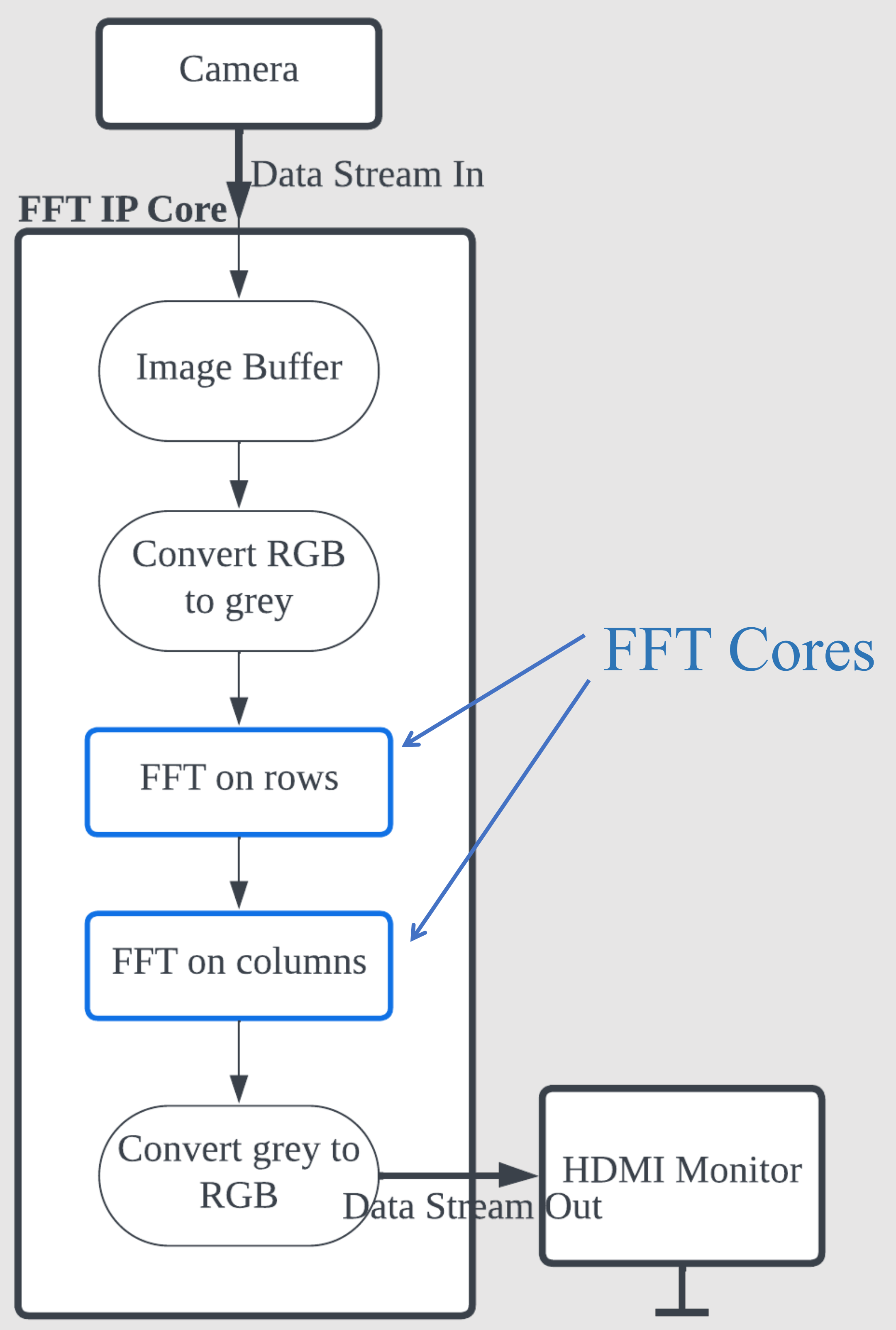
FFT for Image Alignment

- **Demonstrates the feasibility of using 2D FFT to compress a 128 * 64 video image on an FPGA board.**
- Collaborative effort with ASML to compress input images for photolithography machines using Fast Fourier Transform algorithm.
- Utilization of FPGAs for efficient data processing compared to CPUs.
- Initial steps involve FFT algorithm implementation on software platform, followed by expansion to 2D FFT for image processing.
- High-Level Synthesis tools used to generate RTL code.
- FPGA-based accelerator designed to process real-time input images from camera and output compressed images efficiently.

Algorithm Implementation


Complex Array
Length
Inverse
FFT Inputs
Find the middle element
Inverse == 1
Form Pairs for the Input Elements
Form pairs for the 1st element and the middle element
NO
Paired all elements?
YES
Calculate the rotation factor
Implement the Butterfly algorithm with the factor
Check *Inverse* signal
Inverse == 0
DONE
Real-Part Array
Img-Part Array

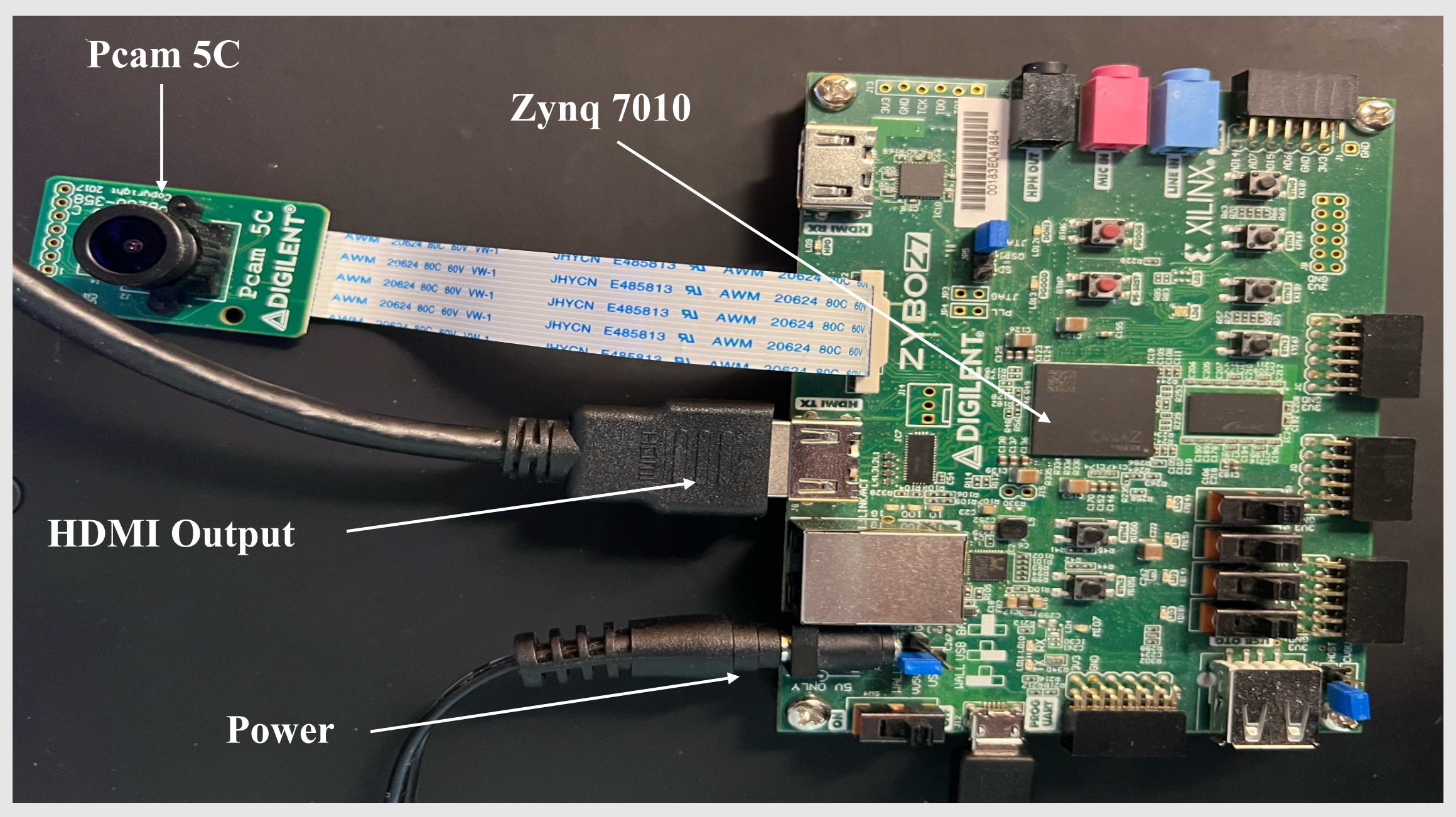
FFT IP Core



- An analog camera is connected to the Zynq 7010 FPGA board.
- Creation of a 2D matrix to store the processed data.
- FFT algorithm applied to rows and columns of the matrix.
- Data encoder converts data into HDMI signals for display.

The FFT hardware core is synthesized from the C++ implementation using the Vitis high-level synthesis tool. Once the hardware core is generated, it undergoes integration into the Zynq Real-Time Image Processing System architecture. This integration process ensures seamless incorporation of the FFT functionality into the broader system framework, enabling efficient real-time image processing capabilities. The implementation of the FFT IP on the Zynq 7010 FPGA board begins with the connection of an analog camera to the board. The camera data is then processed through the FFT IP core, designed specifically for this purpose. Initially, the RGB data undergoes conversion to grayscale to match the FFT core's single-channel requirement. Subsequently, a 2D matrix is generated to store the data, after which the FFT algorithm is applied to both rows and columns of the matrix. Finally, the processed data stream is directed to an RGB to DVI encoder block for conversion into HDMI-compatible signals, enabling display on suitable devices.

Hardware Breakdown



- Zynq 7010
- Pcam 5C
- HDMI Port

Video Output

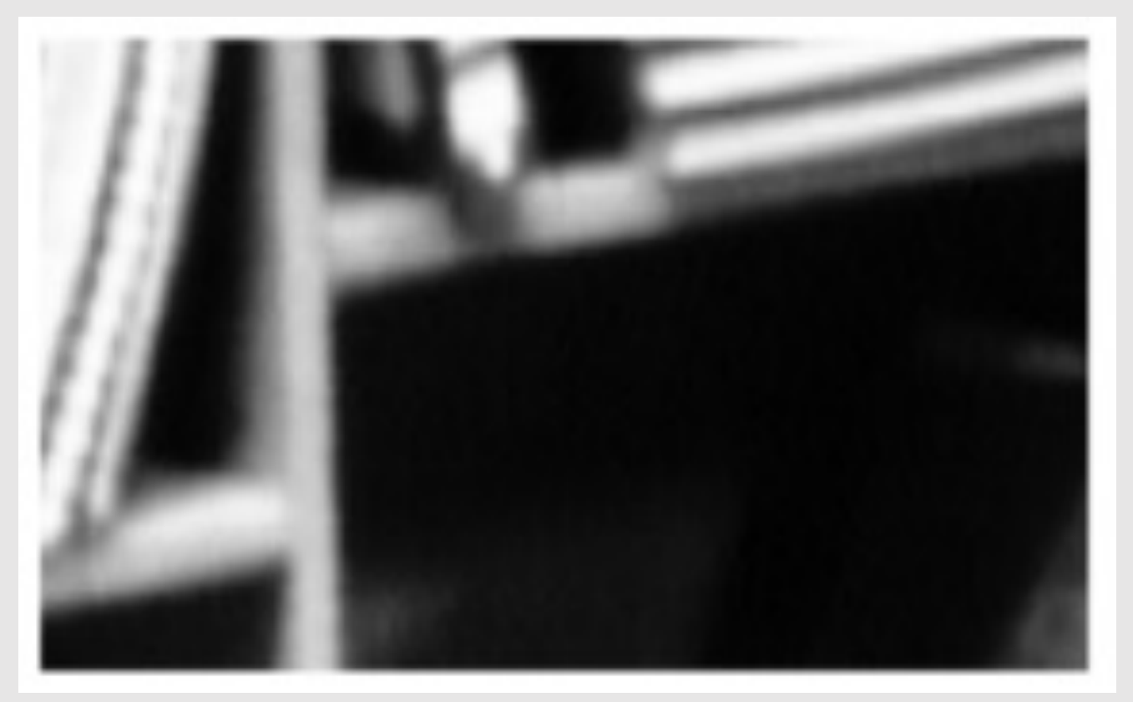


Figure 1. The original figure

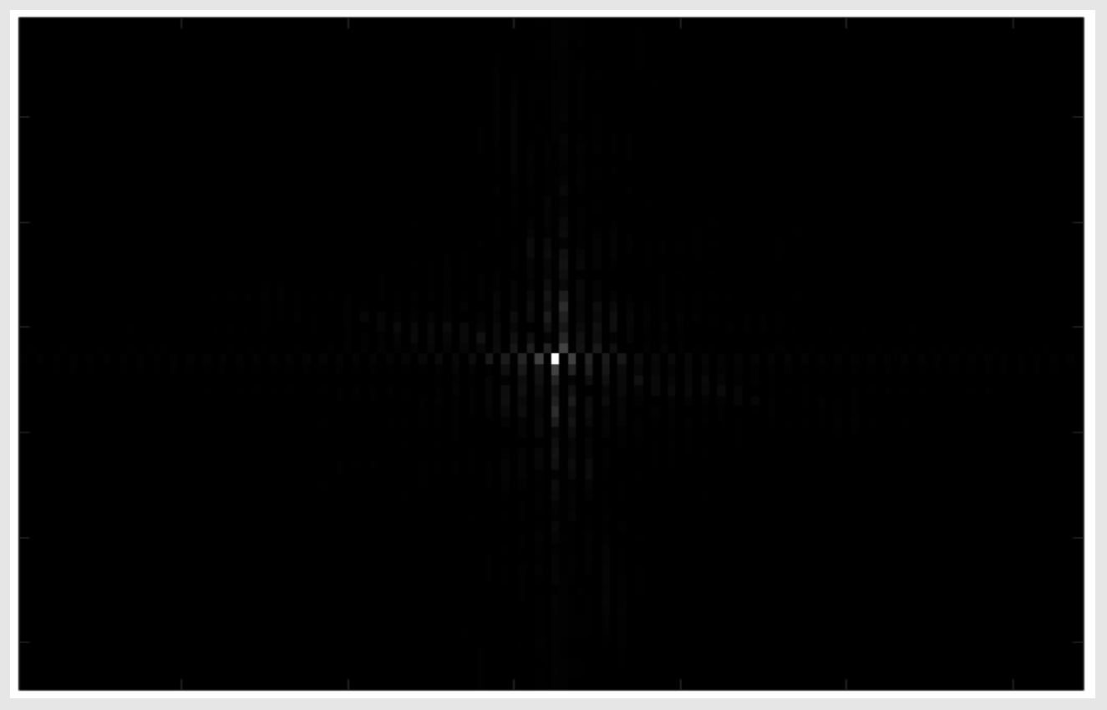


Figure 2. The FFT figure using python

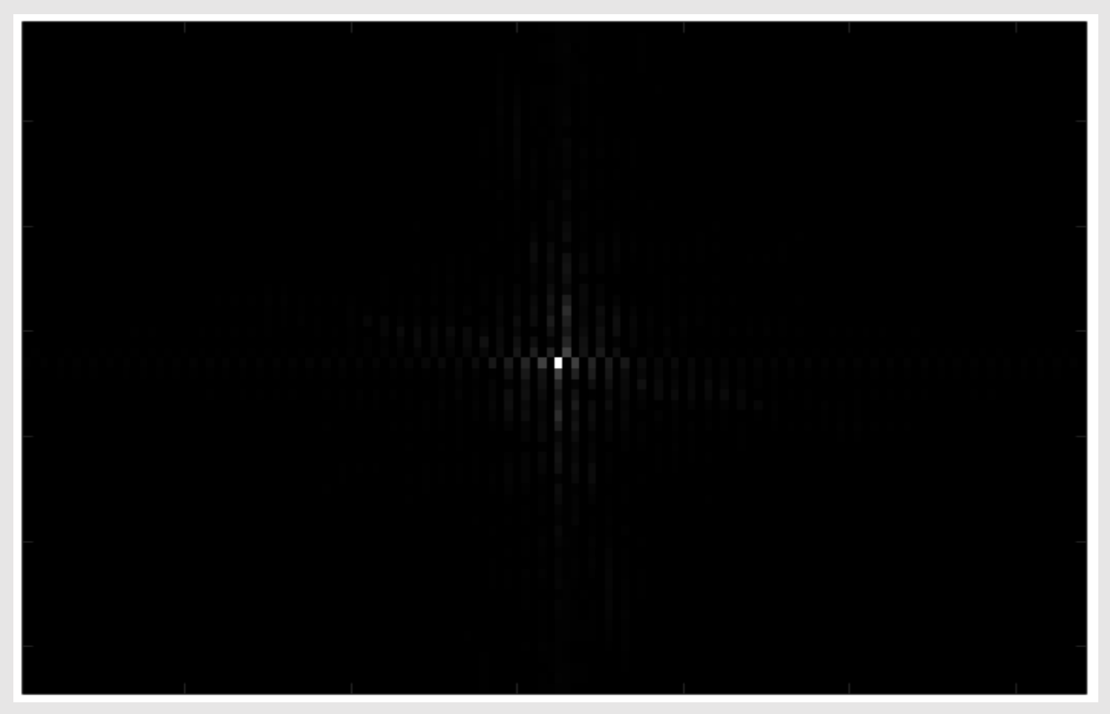


Figure 3. The FFT figure using C++

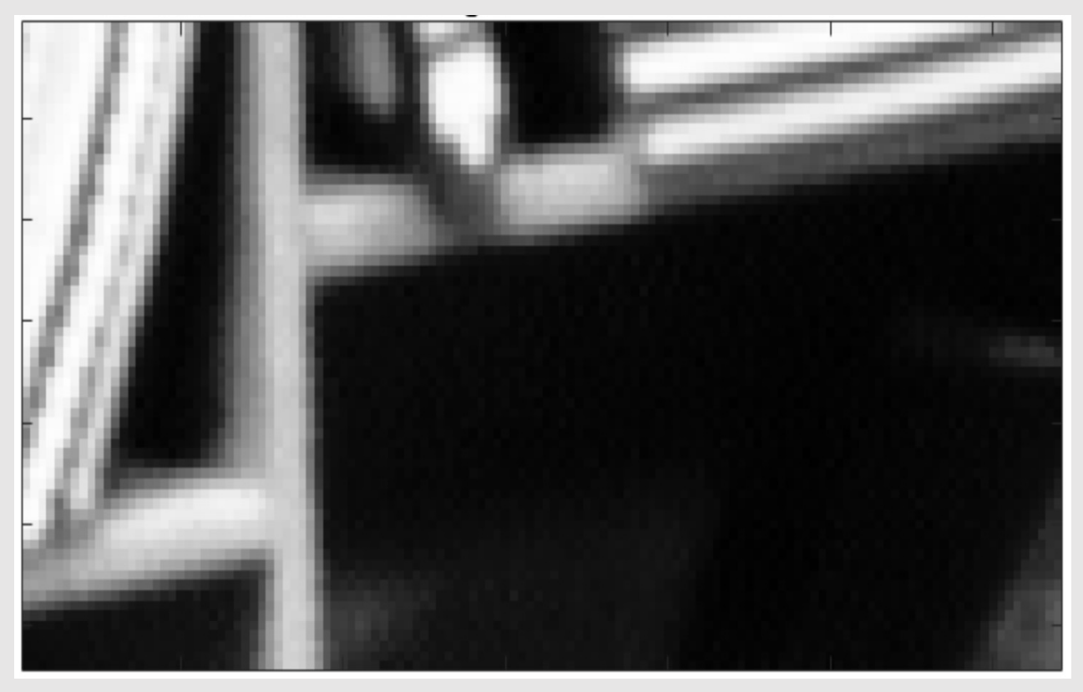
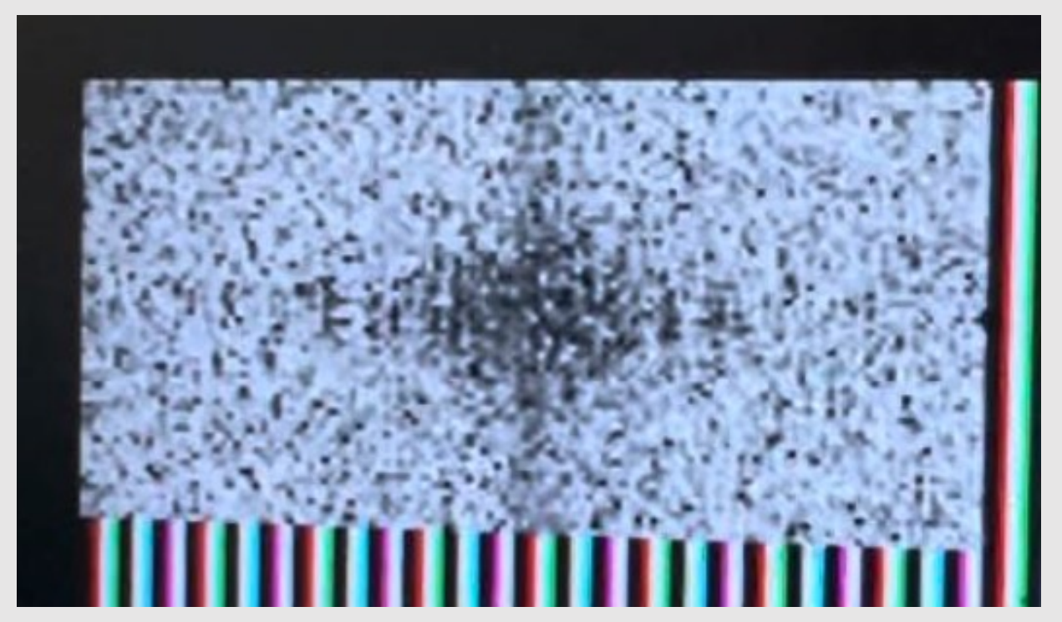


Figure 4. The inverse FFT figure using C++

- **Figure 1:** The test image which simulates the input from the camera.
- **Figure 2:** This figure is utilized to verify the correctness of the FPGA-based FFT figure.
- **Figure 3:**The RTL (Register-Transfer-Level) code is generated by Vitis HLS (High-Level-Synthesis). The C++ model is utilized to feed the HLS tool.
- **Figure 4:** The C++ inversed FFT figure matches the original figure, indicating the functionality of the C++ is correct.

Figure 3 matches Figure 2, indicating that the C++ module works as expected. Figure 4 reflects that the inverse FFT image matches the original Figure 1, which also indicates that the C++ implementation passes the functionality verification.

- The camera input and FPGA-based FFT module output:



**The FFT image output from the FPGA is NOT entirely correct. This discrepancy may stem from issues such as data conversion or overflow. Consequently, further debugging and development are required to address these potential issues.*

Acknowledgement

[1] AMD. "Programming the Device." Vivado Programming and Debugging, version 9.0.8, AMD, [Online]. Available: https://docs.amd.com/r/en-US/ug908-vivado-programming-debugging/Programming-the-Device

[2] Van Hunter Adams, "Understanding the Cooley-Tukey FFT," [Online]. Available: https://vanhunteradams.com/FFT/FFT.html

[3] Massachusetts Institute of Technology, "FFT Tutorial," Nov. 12, 2002. [Online]. Available: https://web.mit.edu/6.111/www/f2017/handouts/FFTtutorial121102.pdf. [Accessed: Dec. 1, 2023].

CornellEngineering
Electrical and Computer Engineering