FPGA-Based Real-Time Video 2D FFT Accelerator

Proposal for a design project for the School of Electrical and Computer Engineering

By

Yibin Xu | yx623
Ruyi Zhou | rz443

Project Advisor: Adams, Hunter

Signature:

Date: Nov 30nd 2023

Abstract

Advanced Semiconductor Materials Lithography Holding N.V. company (ASML) seeks a solution to compress the size of the input images of their photolithography machines. To address this, we propose the implementation of two-dimensional (2D) Fast Fourier Transform (FFT) on real-time input images. However, the computationally intensive nature of the FFT algorithm can increase latency. To counter this, we aim to develop an Field Programmable Gate Array (FPGA) based accelerator, leveraging the parallel computing capabilities of FPGA for efficient execution of 2D FFT on live image data streams [1]. These streams, sourced directly from camera feeds, will be processed on a ZYNQ Z7010 FPGA, enabling rapid and efficient 2D FFT computation suitable for various applications. We are currently in the process of designing and verifying the software-based FFT algorithm. Once the algorithm is successfully implemented in C++, our plan is to utilize the Vitis High-Level-Synthesis (HLS) tool to generate Register-Transfer-Level (RTL) code [2], which will then be synthesized on the ZYNQ FPGA board. We anticipate that this FPGA-based FFT approach will significantly outperform CPU-based FFT computations, effectively aiding ASML in compressing the size of the real-time input images.

Introduction/Overview

In the rapidly advancing field of modern semiconductors, ASML has emerged as a leader among chip fabrication companies. Our collaborative effort with ASML focuses on developing a solution to compress the size of input images for their photolithography machines. To achieve this, we have implemented the Fast Fourier Transform (FFT) algorithm. At this juncture, the high configurability and flexibility of Field-Programmable Gate Arrays (FPGAs) play a crucial role. FPGAs, known for their ability to be custom-programmed for specific applications, offer more efficient data processing and algorithm execution compared to CPUs. This capability grants FPGAs considerable advantages in parallel processing and real-time data processing. Consequently, we are employing FPGAs to design an accelerator specifically for the computation-intensive aspects of the FFT algorithm. We anticipate a significant reduction in execution time, and thus a substantial increase in performance, once the intensive computational part of the FFT algorithm is transferred to the FPGA platform.

To achieve our goal, our initial step involves implementing the Fast Fourier Transform (FFT) algorithm on a software platform. This is to verify the algorithm's feasibility before expanding it into a 2D FFT, which will enable the processing of input images. Following this, the second step is to use High-Level Synthesis (HLS) tools to generate the Register Transfer Level (RTL) code for the ZYNQ FPGA. After conducting post-synthesis simulation of the algorithm, we will program the FPGA with the generated bitstream to construct our first prototype. If the prototype functions as expected, our next plan involves optimizing the design using HLS directives, aiming to further enhance performance. This optimized FFT accelerator is expected to significantly reduce the execution time of the FFT.

The FPGA-based accelerator will be configured to receive real-time input images from a camera. It will utilize the programmed hardware to perform parallel computing for the FFT algorithm, and then output the compressed images. This setup ensures efficient and timely processing, leveraging the strengths of FPGA in handling complex computational tasks in real time.

Issues to be addressed

Our first challenge was setting up the environment for the ZYNQ board and configuring the test bench with Vitis HLS. This was essential to verify our FFT algorithm and its conversion from C++ to RTL code. Additionally, C++ coding for HLS differs from standard C++ coding. We needed to modify the algorithm to ensure that the software code aligns optimally with the board's hardware structure to achieve maximum throughput. Moreover, it was crucial to ensure that the pragmas or directives added did not cause the design to exceed the board's hardware resource limits and to verify that the generated code was synthesizable. After successfully programming the FPGA, another

challenge arose: designing test benches that could cover corner cases and pass both the functionality coverage test and the line coverage test.

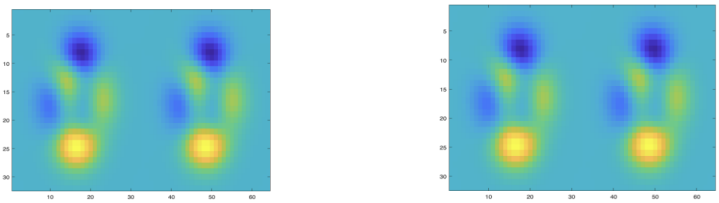
Once the FFT accelerator functions as expected, our next challenge is managing the interface between the real-time input and the FPGA board. Given that the entire system operates in real-time, it's imperative to ensure that the interface protocol is effective and provides the FPGA with accurate input data. This involves careful calibration and testing to maintain seamless data flow and system integrity in a real-time environment.

This design is complex, so we decided to adopt an incremental approach, starting with simple modules and progressively moving to more complex ones. This incremental design strategy enables us to test each unit thoroughly and advance more smoothly through the development process. By building and validating in stages, we can ensure the integrity and functionality of the design at each step, making the overall process more manageable and efficient.

Approach and Expected Results

We aim to implement a 2D FFT function on the Zybo board by utilizing Vivado HLS for translating the function from C++ into Verilog. This will enable the Zybo to process an input image and produce its FFT equivalent as output. There are a few major milestones we expected to achieve:

- FFT in C++:
    - Develop a 1D FFT algorithm in C++ and verify its accuracy against MATLAB's built-in FFT library.
    - Write a 2D FFT algorithm in C++ and validate its correctness with MATLAB's built-in FFT library.
    - Conduct edge case testing.
    - Transition from integer to fixed-point representation for variables.



C++ IFFT Matlab IFFT

Figure 1. The Inverse-FFT from C++ and Matlab

The figures above show the inverse-FFT results from C++ and Matlab, which indicate the functionality verification of the C++ FFT is done and as expected.

- Vivado HLS Application:
    - Create a C++-based LED blink project, convert it to Verilog using HLS, and test it on an FPGA board.

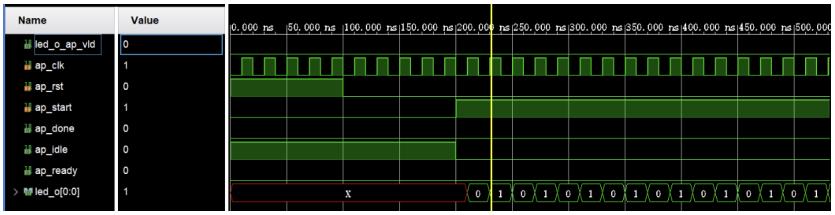


Figure 2. The waveform of the LEB blink program

The waveform above shows the post-synthesis simulation of the LED blink project, which indicates the Vitis HLS can be successfully done for a C++ program.

    - Develop an ARM-controlled C++ program and deploy it on an FPGA board.
    - Import images into the ARM program, process the image data through the program's Logic Unit, and verify the output.
- 2D FFT Implementation on Zybo Board:

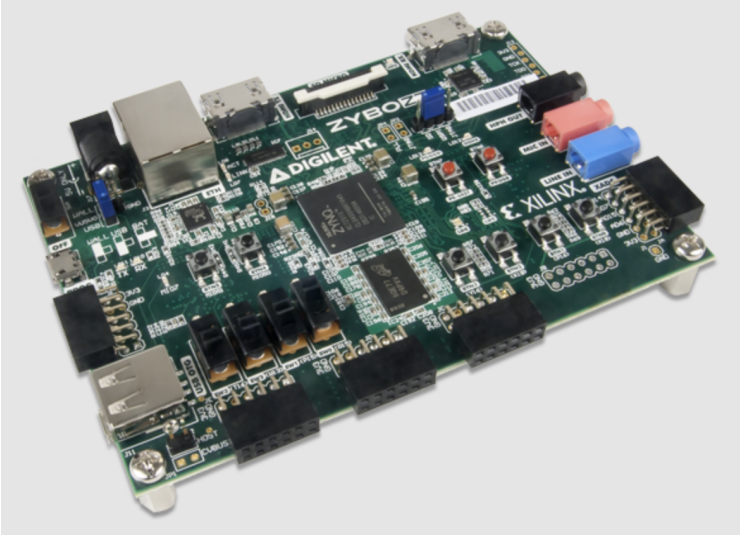


Figure 3. The Zybo Z7010 System-on-Chip board

    - Convert the 2D FFT C++ code into HLS format for compatibility with the Zybo board.
    - Integrate the code with the Zybo board, ensuring correct pin outputs, and test the implementation.
- Real-time Video Processing:
    - Assemble a camera module to test the FFT function's integration.
    - Transform the camera's input into a matrix format and route it to the FFT function's input.
    - Perform tests and debugging to ensure functionality.
- Testing and Debugging:
    - Assess the FFT function's overall performance and establish a performance baseline.
- HLS Acceleration for FFT:
    - Implement an accelerated FFT algorithm using HLS.
    - Compare the accelerated version's performance against the baseline to gauge efficiency improvements.

Outcome Expectation:
Anticipate that the 2D FFT processing will compress image size without significant quality loss and enhance processor efficiency. Target processing 30 images per second to achieve fluid video playback. This restructured outline segregates your plan into clear stages and objectives, making it straightforward and methodical.

Process and Project Flow:
In general, our project design begins with the development of the FFT algorithm using a software language such as C++. We then verify its correctness using MATLAB. Once the functionality is confirmed, we utilize the Vitis HLS tool to generate the RTL code, followed by conducting post-synthesis simulation. After successful functionality verification, we implement the design on the FPGA board.

The essence of the project is that the device will accept images as input and execute the FFT algorithm. This part of the process will be carried out on the FPGA. Subsequently, the FPGA will output the frequency-domain data, enabling the client to apply filters as needed. The device also has the capability to perform the inverse FFT, converting the data back from the frequency domain to the time domain. The figures below illustrate the process flow.

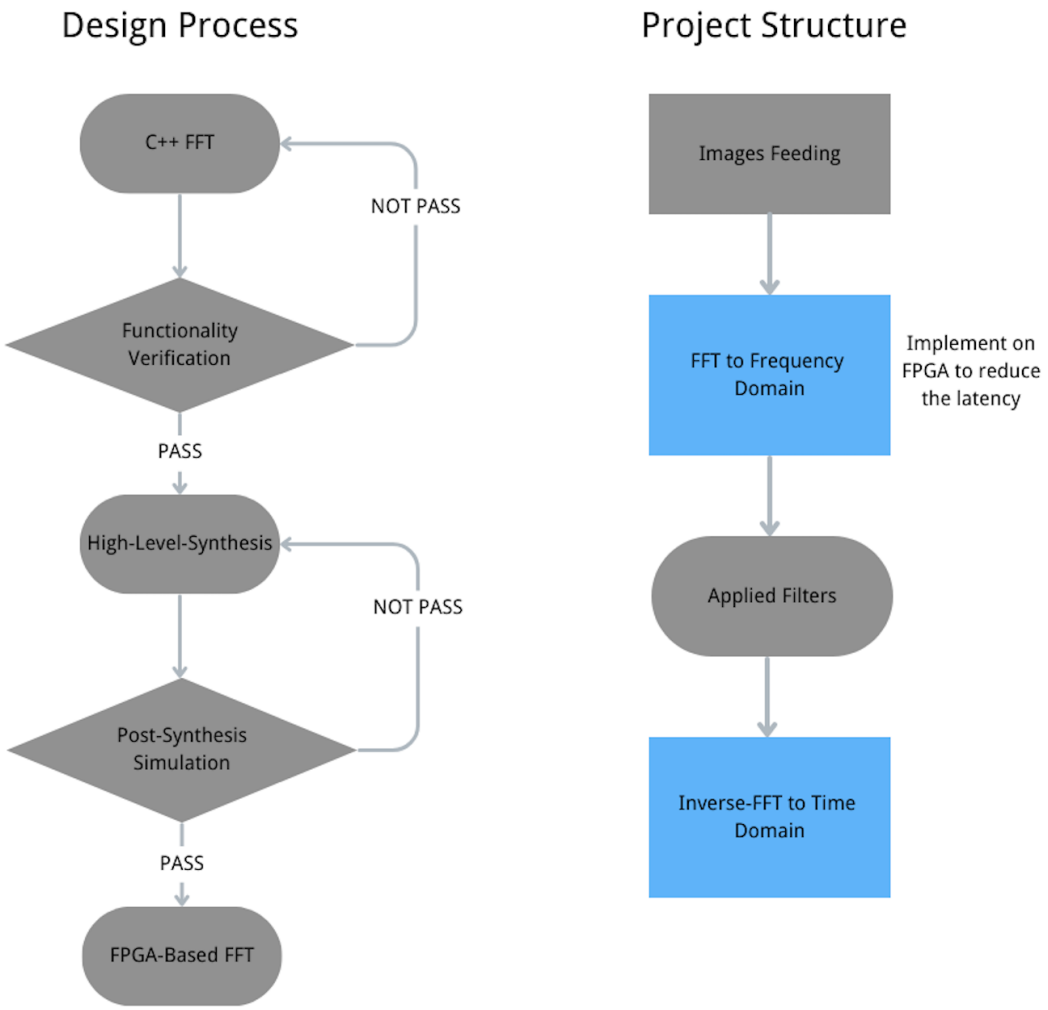


Figure 4. The Design Process Flow of the FPGA-Based FFT Accelerator

Conclusions

In conclusion, this proposal outlines a comprehensive approach to develop an FPGA-based accelerator for ASML, focusing on the compression of real-time input images using a 2D Fast Fourier Transform (FFT) algorithm. Our strategy involves a meticulous process of developing, testing, and optimizing the FFT algorithm, first in a software environment and then transitioning to a hardware implementation on a ZYNQ Z7010 FPGA. We have addressed the initial challenges in setting up the environment and ensuring the algorithm's compatibility with the hardware. Our design process, adopting an incremental approach, ensures thorough testing and optimization at each stage. We anticipate that this FPGA-based solution will not only meet but exceed the performance capabilities of CPU-based systems, providing ASML with an efficient, real-time image compression tool. This project, with its focus on

advanced semiconductor technology and innovative design methodologies, represents a significant step forward in the field of image processing and photolithography, positioning both our team and ASML at the forefront of technological advancements in semiconductor manufacturing.

Timeline

This figure below shows a timeline of this project.

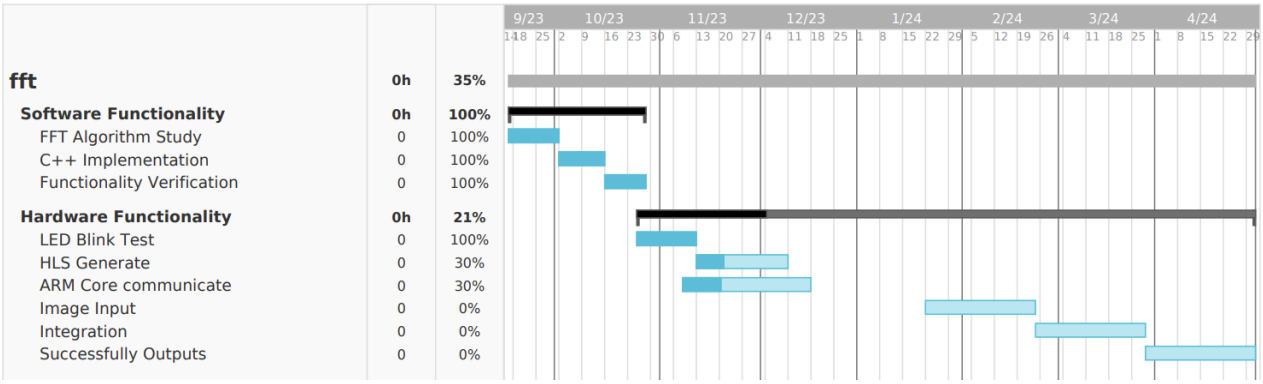


Figure 5. The Project Timeline

As shown above, we have completed the implementation using C++ and the functionality verification. Additionally, the demo project involving an LED blink has been finished to test the FPGA board's functionality. The remaining work primarily concerns the hardware aspect. Currently, we are engaged in establishing ARM core communication and undertaking the pre-HLS step. Managing the ARM core control and interfacing with real-time inputs may pose the greatest challenge in the next phase. Therefore, we anticipate allocating more time to this step. Considering the winter break, we expect to complete the project by April.

References

[1] 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].

[2] R. Kastner, J. Matai, and S. Neuendorffer, "Parallel Programming for FPGAs," arXiv:1805.03648v1 [cs.AR], May 9, 2018.