Level-Cam

Akrit Shrikant Patrick Parsegian

***Abstract*—This report presents the design and implementation of our team's final project in COM SCI 152B. The purpose of this project was to implement a camera with an indicator to tell the user if stabilizing is required. We implemented this device with a Basys3 FPGA, a camera module, an accelerometer module, and a gyroscope module. We aimed to create a product with real-world value, using the skills we learned throughout this course. To this end, we used FGPAs to implement our project, along with modules that we had never used before.**

I. Introduction

With the proliferation of embedded systems and the increasing demand for efficient processing of multimedia data, there is a growing interest in utilizing FPGA technology for real-time image processing tasks. In this project, we leveraged the capabilities of Basys3 FPGAs to develop a camera stabilization system. The system aims to assist users in maintaining stable camera footage by analyzing data from the inertial sensors (accelerometer and gyroscope).

Our fully-functioning camera conjoined with a built-in level provides a feature that handheld digital filming equipment in this budget does not. When steady and leveled camerawork is required in filming, it is crucial for the director or cameraman to be in tune with the camera. While modern cameras can recognize these changes in inertial movement, they are both prohibitively expensive and unnecessarily complicated. Our goal in designing this camera was to create a simple camera with this inbuilt functionality using an accelerometer and gyroscope. We have also cut out any latency issues with wireless communication between cameras and monitors, thus utilizing the VGA output from the Basys3 board.

Using this device would work as follows:

- The User turns on the device and the camera feed displays to the monitor through a VGA connection.
- If the camera is tilted too far in any direction, a light is turned on based on whether the camera is tilted along the X or Y axis to notify the user to stabilize the camera.
- If the camera is too far tilted on both the X and Y axis, both lights are turned on.

II. Related Work/Survey

Our research is fairly broad and related to cameras on FPGAs and the use of inertial sensing modules. This field is well-researched, so we were able to find a number of sources relating to integrating these modules into our FPGA. With regards to cameras, there are myriad sources with regards to installing a camera to an FPGA, and displaying it to a VGA display. Additionally, there are a number of uses for a gyroscope and accelerometer on an FPGA, but not for our uses. In order to take the knowledge we found and apply it to our use case, we required a lot of time in order to become familiar with our FPGA and the inner workings of each module. We first spent time working with our camera, and trying to understand how it works, and then moved on to our sensing modules. With our sensing modules, we required time in order to understand our accelerometer first, specifically to understand how we can use its functionalities in our Level-Cam. We then did the same with our gyroscope, understanding what it can do differently than our accelerometer which brings value to our project.

Apart from research and testing, we also studied the importance of our Level-Cam in today's market. With the increased use of smartphones and the cameras within them, digital cameras have come to be less impactful in day-to-day usage. However, in industries like filming, a digital camera is preferred over a wireless one. This is due to the lack of latency issues digital cameras provide. With this in mind, we valued our product to serve as a low-budget filming camera that offers a stabilization feature along with a real-time feed via VGA.

III. Implementation/Methodology

The implementation of our camera stabilization system involved several key components. After initial delays in deciding the path we wanted to take in our project, we decided to design a system that would provide us the opportunity to work with modules we had no experience with before. Being a multi-module device, on top of the Basys3 boards, the modules we utilized in the design of our system were the OV7670 Camera Module, PMOD ACL2, and PMOD GYRO.

Fig. 1. OV7670 Camera Module

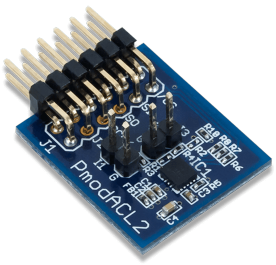

Fig. 2. PMOD ACL2 (Accelerometer)

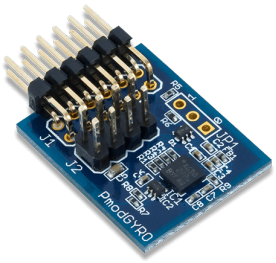

Fig. 3. PMOD GYRO (Gyroscope)

With the combination of these modules, we were able to begin our implementation of the system.

Our first steps involved working with the camera module to receive a live feed via the VGA port on the Basys3 FPGA. With help from the starter guide provided on BruinLearn, we were able to compile the Verilog modules to analyze the given code. After multiple sessions of analyzing and understanding the code, we planned to connect the camera module to the board. This proved to be a challenging task as the first four times the VGA showed no display. Compared with a similar device project online, we were able to find the correct pins to insert the camera into [1]. The unique aspect of working with the OV7670 camera module is that it takes up more than one PMOD port on the Basys3 FPGA. This was a new type of connection that we had no experience with prior to this project. After refreshing our hardware with a new order of camera modules and using electrical tape to stabilize the wires, we were finally able to get a live feed from the camera. Apart from light noise from the surrounding lab environment and the less than ideal camera definition quality, our camera module was fully functional and the VGA connection provided a monochrome display.

Fig. 4. VGA Display from Camera Feed

Once our camera portion was working, we decided to move on and add the accelerometer and gyroscope to the Verilog project. This, however, proved to be a challenge that served as a barrier to the remaining modules of our project. In order to implement the accelerometer and gyroscope, we needed to add the MicroBlaze IP to our board component, however, in Vivado 2018.2, users are unable to pursue both a MicroBlaze board project along with a Verilog project. Since the camera module also took up more than one PMOD port, it was not possible to program the camera using MicroBlaze. Seeing as the MicroBlaze implementation was more important for our learning and experience, we opted to add an additional FPGA and set up the remaining modules there. This way, we would gain experience working in Verilog and C.

To start the inertial modules implementation, we followed the PMOD tutorial provided by Digilent and created the block diagram. After the setup steps were complete, we added the PMODs we planned to use and the sixteen LEDs present on the FPGA itself.

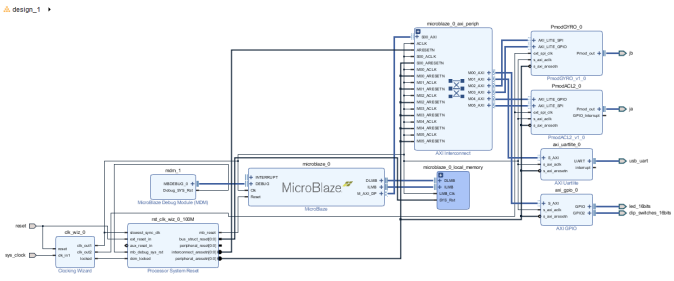

Fig. 5. Inertial Modules Block Diagram

Following this, we uploaded our hardware to SDK and began programming in C. Here, we designed our stabilizing code to sense if the FPGA is tilted over a 0.25 distance from the origin in each direction. We used the accelerometer to measure the 0.25 tilt on the x-axis and the gyroscope to

measure the 0.25 tilt on the y-axis. We then added the code to light up the first and second LED based on the axis of tilt. After testing, we came across a problem where the two LEDs were not lighting up simultaneously when the FPGA was tilted past the threshold in both axes. Since the sixteen LEDs are on a binary system, we created a mask to light up both LEDs rather than lighting each one independently. This solved our problem and ended our implementation of the inertial modules.

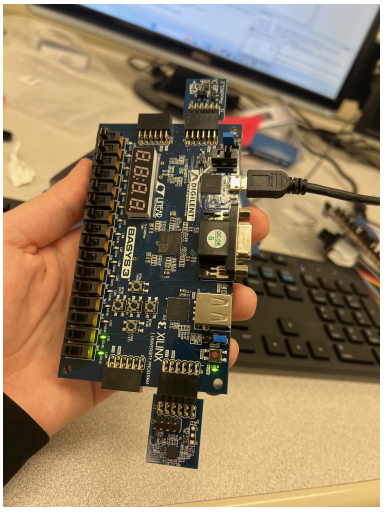

Fig. 6. FPGA Tilted in Both X and Y Axes

An extra feature we wanted to add to our camera was edge detection. To accomplish this, we needed to create a method for taking camera-captured data and transferring it over UART to the PC. Luckily, our camera starter code had an elementary set-up for us to explore more. By developing a Python script and utilizing OpenCV and Pygame, we were able to accomplish this extra feature and get a window for edge detection. Essentially, the camera would take a photo every 10 seconds and analyze the edges within the Pygame window.

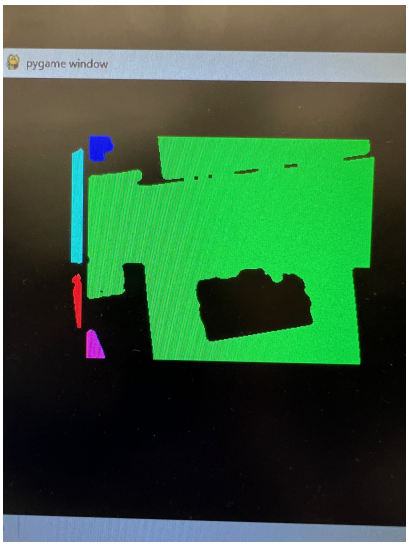

Fig. 7. Edge Detection Pygame Window

IV. Results

The result of our project was a fully-functioning Level-Cam that worked as expected – if the device was tilted, an LED would light up based on what axis needed to be corrected. We saw the output of our camera on our VGA display, along with rudimentary edge detection. This was a result of our tireless work understanding each module individually, and then integrating them to produce a functioning device. Additionally, we implemented edge detection, a foray into advancing our stabilization and recording techniques. This would allow a more advanced Level-Cam to intuitively focus on an object in frame, and determine independently the ”level” to which the camera should be tilted. This would not only make the stabilizer more effective but also provide assistance to amateur camera users on how to film.

In the future, we plan to take this experience working with the FPGAs and their respective modules to advance our knowledge in the integration of sensor data fusion techniques. If we had more time to work on our device, we would have liked to dive deeper into edge detection and be able to do object tracking or some sort of facial recognition. Furthermore, we would have also liked to work with a more high-definition camera module, which would have had color image processing. That way we would be able to add color recognition to our edge detection cases as well. Lastly, rather than using a zip-tie to put together the two FPGAs we would have liked to have the opportunity to design and 3D print a casing for our device, stabilizing the camera module within the printed case itself.

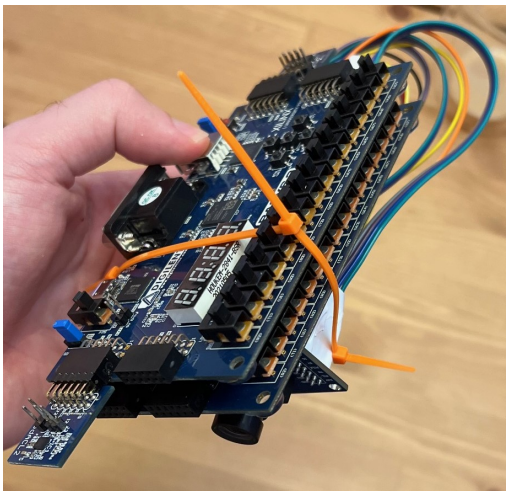

Fig. 8. Combined Basys3 Boards

V. Conclusion

Our Level-Cam project showcases the potential of FPGA technology in developing camera stabilization systems. By leveraging the parallel processing capabilities of FPGAs and integrating sensor data fusion techniques, we have created a robust system capable of providing real-time feedback to users. Future work could involve further optimization of algorithms

and expanding on edge detection to add additional features such as object tracking and more advanced stabilization techniques. Overall, this project contributes to the ongoing research in FPGA-based multimedia processing systems and opens up avenues for various applications in the field of image and video processing.

REFERENCES

[1] https://www.fpga4student.com/2018/08/basys-3-fpga-ov7670-camera.html