Ben-Gurion University of the Negev

Faculty of Engineering Science

School of Electrical and Computer Engineering

Dept. of Electrical and Computer Engineering

Fourth Year Engineering Project

Final Report

SW/HW optimization for NN in a TFLM environment on a RISC processor

Project number: p-2024-046

Students: Noam Perets, Noam Vaknin

Submitting date: 18/08/2024

Table of Contents

Table Of Figures

SW/HW optimization for NN in a TFLM environment on a RISC processor

1. Abstract

Background: With the growing accessibility of AI technologies, developing neural networks on affordable RISC processors and FPGA platforms has become increasingly viable for developers and researchers. The use of FPGA boards like the Ultra96-v2, combined with inexpensive components such as the OV7670 camera, offers a promising approach for low-cost, high-performance embedded systems.

Purpose: The purpose of this project is to create a fundamental infrastructure that enables the seamless integration of the OV7670 camera with the Ultra96-v2 board. This infrastructure is designed to support future neural network projects, leveraging the processing power and flexibility of FPGA technology.

Objectives: Our primary objectives are to develop a versatile and expandable FPGA design that captures and processes image data from the OV7670 camera. This design will be optimized for low power consumption and high performance, ensuring it can be easily adapted for various neural network applications.

The innovation: This project presents a modular FPGA design that simplifies the integration of low-cost cameras, like the OV7670, with RISC-based processors on the Ultra96-v2 board. The innovation lies in creating a flexible, expandable infrastructure that enhances the adaptability of embedded systems for AI.

Proposed method: Using Vivado and Vitis, we configured the Ultra96-v2 board with an OV7670 camera, implementing modules for capturing, controlling, and storing image data in BRAM. The captured images were processed using the PYNQ operating system, which allowed us to filter noise and reconstruct the images.

Expected results: We anticipate that our project will deliver a functional and expandable infrastructure for integrating cameras into FPGA-based neural network systems, though image quality and system complexity may require further refinement.

Keywords: XILINX FPGA, TFLM, ULTRA-96v2 board, OV7670 camera, Vivado, Neural Networks, RISC Processor, IoT.

אופטימיזציית חומרה ותוכנה לרשתות נוירונים בסביבת TFLM על מעבד RISC

2. תקציר

רקע: עם הנגישות הגוברת של טכנולוגיות בינה מלאכותית, פיתוח רשתות עצביות במעבדי RISC ופלטפורמות FPGA בעלות סבירה הפך לאפשרי יותר ויותר עבור חוקרים ומפתחים. השימוש בFPGA כמו ה-Ultra96-v2, בשילוב עם רכיבים זולים כמו מצלמת OV7670, מציע גישה מבטיחה למערכות משובצות מחשב בעלות נמוכה ובעלת ביצועים גבוהים.

מטרה: מטרת פרויקט זה היא ליצור תשתית בסיסית המאפשרת שילוב חלק של מצלמת OV7670 עם לוח Ultra96-v2. תשתית זו נועדה לתמוך בפרויקטים עתידיים של רשתות עצביות, תוך מינוף כוח העיבוד והגמישות של טכנולוגיית FPGA.

יעדים: היעדים העיקריים שלנו הם לפתח חומרה מבוססת FPGA רב תכליתית שמעבדת נתוני תמונה ממצלמה OV7670. פיתוח זה יהיה מותאם לצריכת הספק נמוכה וביצועים גבוהים, מה שמבטיח שניתן להתאים אותו בקלות ליישומי רשתות נוירונים שונים.

החידוש: פרויקט זה מציג חומרה מבוססת FPGA מודולרית המפשטת את השילוב של מצלמות בעלות נמוכה, כמו OV7670, עם מעבדים מבוססי RISC כמו Ultra96-v2. החידוש טמון ביצירת תשתית גמישה וניתנת להרחבה שמשפרת את יכולת ההסתגלות של מערכות משובצות מחשב עבור AI.

שיטה מוצעת: באמצעות Vivado ו-Vitis, הגדרנו את לוח Ultra96-v2 עם מצלמת OV7670, תוך הטמעת מודולים ללכידה, שליטה ואחסון נתוני תמונה בזיכרון. התמונות שצולמו עובדו באמצעות מערכת ההפעלה PYNQ, שמאפשרת לסנן רעשים ולשחזר את התמונות.

התוצאות הצפויות: אנו צופים שהפרויקט שלנו יספק תשתית פונקציונלית וניתנת להרחבה לשילוב מצלמות במערכות רשתות נוירונים מבוססות FPGA, אם כי איכות התמונה ומורכבות המערכת עשויים לדרוש שיפור נוסף.

3. Introduction

As technology continues to evolve, the integration of artificial intelligence (AI) into everyday applications is becoming increasingly accessible. This trend has given rise to the development of sophisticated yet affordable systems that can be built and optimized even by those with limited resources. A key area of interest within this field is the use of Field Programmable Gate Arrays (FPGAs) in conjunction with RISC-based processors to create efficient, low-cost platforms for AI-driven applications. The versatility and power of these components make them ideal for tasks ranging from AI to Internet of Things (IoT) devices, where real-time processing and low power consumption are critical [1].

The motivation for this project stems from the desire to explore the capabilities of combining an FPGA with a RISC processor to efficiently implement neural network models. Specifically, this project aims to integrate a low-cost OV7670 camera with the Xilinx Ultra96-v2 board to create a versatile platform for neural network applications [1]-[3]. The Ultra96-v2, with its RISC-based processing capabilities, provides an excellent foundation for such a project. By leveraging the strengths of both components, we aim to build a system that is not only powerful but also accessible to developers and researchers who may not have access to high-end hardware.

The purpose of this project is to design and implement a modular infrastructure that enables the seamless integration of the OV7670 camera with the Ultra96-v2 board. This infrastructure will serve as a foundation for future projects, allowing users to develop and deploy neural networks on a versatile, low-cost platform. The project seeks to address the growing demand for AI-enabled systems that are both efficient and scalable, particularly in the context of AI and IoT applications [1]-[3].

In the realm of embedded systems, several approaches have been taken to integrate cameras with FPGA and RISC-based platforms. Traditionally, the integration process has been manual, requiring extensive knowledge of both hardware and software design. Some approaches have utilized high-level programming languages like MATLAB or Python to develop algorithms, which are then translated into lower-level languages such as C for hardware implementation. This method, while effective, is time-consuming and often requires significant engineering resources.

Despite the effectiveness of these approaches, they often come with limitations, particularly in terms of cost and complexity. High-level programming languages, while powerful, can be difficult

when translating to hardware, and custom hardware accelerators can be prohibitively expensive for small-scale projects. Furthermore, the learning curve associated with these methods can be steep, making them less accessible to developers who are new to FPGA and RISC-based systems.

In light of these challenges, this project adopts a different approach. By focusing on the essential components required for camera integration and neural network processing, we aim to develop a more streamlined and accessible infrastructure. The decision to use the Ultra96-v2 board, coupled with the OV7670 camera, was made based on the board's balance of power, flexibility, and cost [3]. Additionally, the use of Vivado and Vitis IDEs allows for a more intuitive design process, enabling us to focus on optimizing the system's performance rather than getting bogged down by unnecessary complexity.

This project's approach not only simplifies the integration process but also offers a scalable solution that can be easily adapted and expanded for future applications. By providing a modular, flexible design, we aim to empower developers to leverage the full potential of FPGA and RISC-based systems in their AI-driven projects [2].

4. Spec Sheet

Project's name: SW/HW optimization for NN in a TFLM environment on a RISC processor

Project's goal: To develop a versatile and modular FPGA infrastructure that integrates a low-cost camera with a RISC-based processor for efficient neural network processing.

Product application: The resulting system can be used in AI, IoT devices and other embedded systems that require real-time image processing and low-power consumption.

Stages of research and work:

- Conducted a "Hello World" project [4] using Vivado and Vitis to understand the Ultra96-v2 board.
- Studied the OV7670 camera, Pmod interface, and Ultra96-v2 board characteristics.
- Utilized a modular FPGA design using Vivado, focusing on essential components like Capture, Controller, RAM, and AXI interfaces [5].
- Implemented image data capture and storage in BRAM from the OV7670 camera [5].

- Used Vivado ILA to debug system issues.
- Utilized PYNQ [6] to read data from BRAM and reconstruct images in JPEG format, filtering noise and irrelevant data.

4.1 Required Hardware:

1. **Ultra96-V2 board**
2. **OV7670 camera module (2017/3/15)**
3. **SD card**
4. **Micro USB cable**
5. **GPIO low speed pmod bridge**
6. **USB-to-JTAG/UART pod for Ultra96**
7. **18 female-male jumper cables**

4.2 Required Software:

1. **Vivado 2022.2**
2. **Vitis IDE & HLS**
3. **BalenaEtcher**
4. **PYNQ ISO**

4.3 Steps to Set Up:

1. Download and Prepare the Vivado Project:

- **Download the Vivado project** on your computer.
- Open the project in **Vivado 2022.2**.
- Click on **Generate Bitstream**.
- Once the bitstream generation completes, go to **File -> Export -> Export Hardware**.
- You will get an .xsa file. Open this file using WinRAR or a similar tool.
- Extract the .bit and .hwh files, ensuring they have the same name. These files contain the hardware design with the bitstream.

2. Connecting the OV7670 Camera Module to the Ultra96-V2:

Hardware Connections:

1. **Refer to the provided pinout diagram** [7] to make the correct connections:

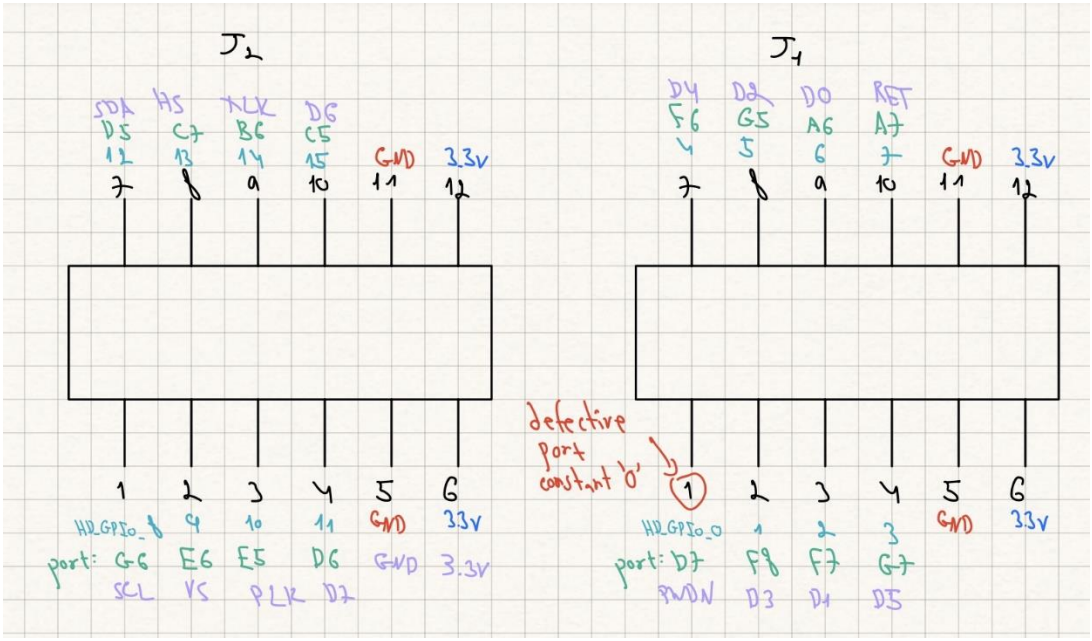

Figure 1: Pinout diagram

2. **Match the purple signals from the camera to the black pins on the bridge**. Here's how to connect each signal:

Camera Signal	**.xdc file port map**	**Ultra96-V2 (J2/J1)**	**Description**
SCL	G6	J2 port 1	SIOC
VS	E6	J2 port 2	Vertical Sync
PCLK	E5	J2 port 3	Pixel Clock
D7	D6	J2 port 4	Data Bit 7
GND	-	J2 port 5	Ground
3.3V	-	J2 port 6	Power
SDA	D5	J2 port 7	SIOD
HS	C7	J2 port 8	Horizontal Sync
XLK	B6	J2 port 9	System Clock

D6	C5	J2 port 10	Data Bit 6
D5	G7	J1 port 4	Data Bit 5
D3	F8	J1 port 2	Data Bit 3
D1	F7	J1 port 3	Data Bit 1
PWDN	D7	J1 port 1 (Defected Port – Constant '0')	Power Down Mode
D4	F6	J1 port 7	Data Bit 4
D2	G5	J1 port 8	Data Bit 2
D0	A6	J1 port 9	Data Bit 0
RET	A7	J1 port 10	Reset

Table 1: Pinout table

Note: The green ports mentioned correspond to the ports in the base.xdc file.

Steps to Connect:

1. **Power off the Ultra96-V2 board** before making any connections.
2. Use the **female-male jumper cables** to connect each pin from the OV7670 camera module to the specified pins on the GPIO bridge.
3. Double-check all connections against the diagram to ensure accuracy.

3. Program the FPGA:

- Go to the **PYNQ website** and navigate to the boards section.
- Download the **Ultra96-V2 SD card image**.
- Use **BalenaEtcher** to write the image onto the SD card.
- Insert the SD card into the Ultra96-V2 board and switch to **SD mode** (not JTAG mode).

4. Setting Up the PYNQ Environment:

- Power on your Ultra96-V2 board and connect it to your computer using Micro USB.
- Open a web browser and go to **192.168.3.1/lab** to access the PYNQ web platform. The /lab extension provides a better view of Jupyter Notebook.
- Drag and drop the .bit and .hwh files into the Jupyter Notebook interface.

5. Running the Python Code:

- Copy the Python code into the Jupyter Notebook. Make sure the filenames in your code match the .bit and .hwh files.
- Run the Python code to initialize the FPGA and capture an image from the OV7670 camera module.
- The code will read data from the memory block, interpret it, and convert it into an RGB444 image.

5. System Solution Approach

The primary objective of this project is to integrate the OV7670 camera with the Ultra96-v2 board, creating a functional infrastructure for capturing and processing image data in real-time using an FPGA. The block diagram below illustrates the system architecture, which is divided into several key components, each playing a crucial role in the overall design [5].

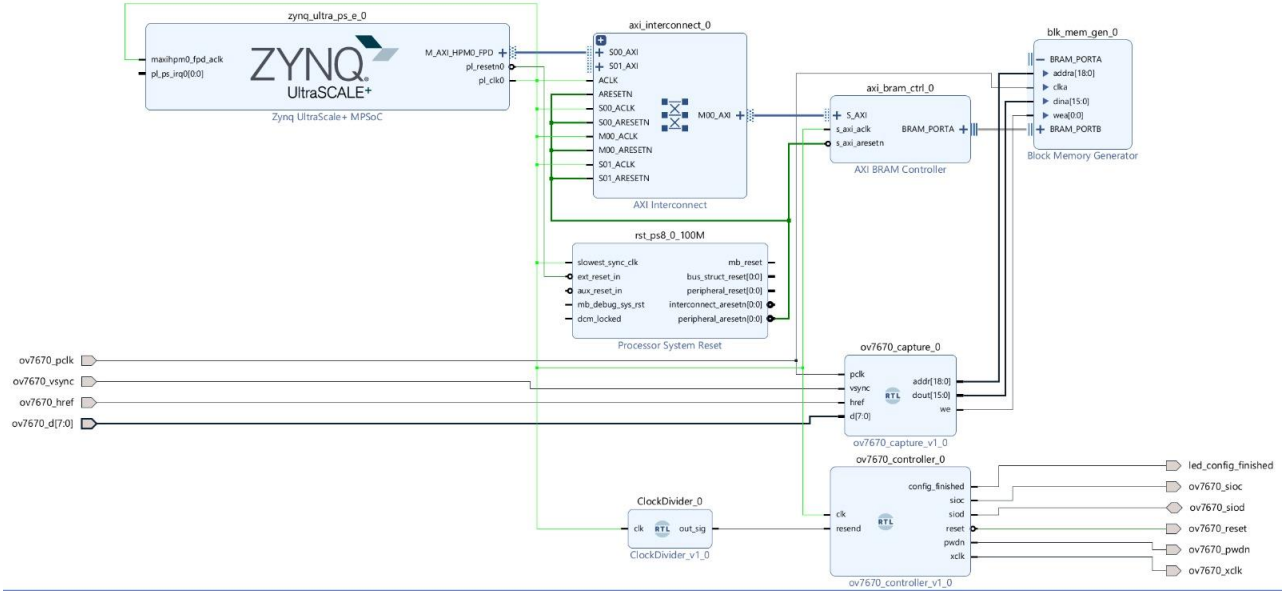

Figure 2: Block Diagram Overview

The block diagram represents the interaction between the Xilinx Ultra96-v2 board and the OV7670 camera, alongside the necessary modules that facilitate data capture, processing, and storage. The

system is designed to be both modular and expandable, allowing for future enhancements and adjustments to the infrastructure as needed [5].

5.1. Detailed Explanation of Each Block

Zynq UltraScale+ MPSoC (zynq_ultra_ps_e_0): This is the central processing unit that manages the communication between the various modules. The Zynq UltraScale+ MPSoC provides a powerful and flexible platform for both software and hardware integration, enabling the processing and control of data captured from the OV7670 camera.

AXI Interconnect (axi_interconnect_0): The AXI Interconnect facilitates communication between the Zynq MPSoC and other peripherals such as the BRAM Controller and Block Memory Generator. It allows for high-speed data transfer and ensures that the system can handle the bandwidth required for real-time image processing.

AXI BRAM Controller (axi_bram_ctrl_0): This module interfaces with the Block Memory Generator to manage the storage of image data captured by the OV7670 camera. The BRAM Controller ensures that data is correctly written to and read from the Block RAM (BRAM) for further processing.

Block Memory Generator (blk_mem_gen_0): The Block Memory Generator provides the necessary storage for image data captured from the camera. It acts as a buffer, temporarily holding the data before it is processed or transmitted to other components of the system.

OV7670 Capture Module (ov7670_capture_0): This module is responsible for capturing the pixel data from the OV7670 camera. It synchronizes with the camera's pixel clock (pclk) and processes control signals like vsync and href to correctly time the storage of image data in BRAM [5].

OV7670 Controller Module (ov7670_controller_0): The controller module manages the configuration and operation of the OV7670 camera. It uses an I2C-like interface to send commands to the camera, setting up parameters such as resolution and color format. The module also handles power control signals like reset and xclk [5].

Clock Divider (ClockDivider_0): This module counts 10 seconds using the system clock and then sends a signal to the OV7670 controller, prompting it to resend the camera configurations. This periodic signal ensures that the camera settings remain stable and are consistently applied during operation.

Processor System Reset (rst_ps8_0_100M): This module handles the system reset signals, ensuring that all components start in a known state. It manages resets for the AXI Interconnect, BRAM Controller, and other peripherals to ensure system stability and proper operation.

6. System Debugging and Testing

Effective debugging is a critical aspect of developing a reliable and functional system. In our project, we employed two primary methods for debugging: Using the Integrated Logic Analyzer (ILA) within the Vivado environment to observe and analyze signal behavior, and utilizing PYNQ [6] to manipulate and read data from the Block RAM (BRAM) to generate images. These methods allowed us to identify, isolate, and resolve issues within our design.

6.1 Test Array and Signal Analysis with ILA

The first method involved the use of Vivado's Integrated Logic Analyzer (ILA), which was integrated into the block diagram of our design. The ILA allowed us to capture and visualize real-time signal data directly from the FPGA, making it possible to monitor critical signals such as the pixel clock (pclk), synchronization signals (vsync, href), and data outputs from the OV7670 camera.

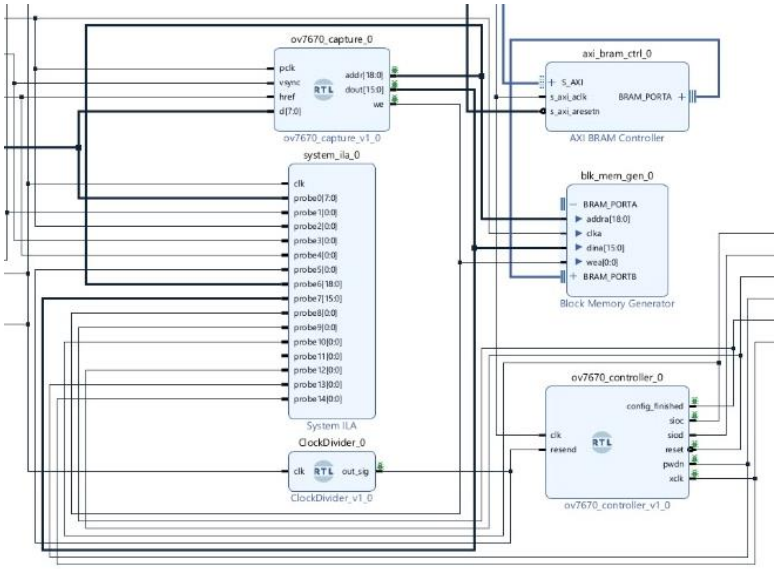

Figure 3: System ILA (Logic Analyzer)

One of the primary issues we encountered during testing was the instability of the pixel clock (pclk). The pclk signal was observed to take unsynchronized and unstable values when compared to the system clock. This instability led to errors in the timing of data capture from the camera, resulting in corrupted or incomplete images.

To diagnose this issue, we configured the ILA to monitor the pclk signal alongside the system clock and other related control signals. The ILA output clearly demonstrated the discrepancy between the system clock and the pclk, confirming that the clock signal from the camera was not properly synchronized with the rest of the system.

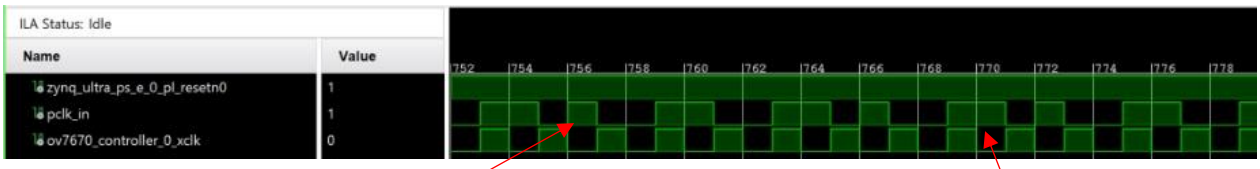

Figure 4: Debugging Using System ILA

Pixel clock is unsynchronized (Different rising edge from system clock)

Pixel clock is unstable (Clock cycle is uneven)

6.2 Data Manipulation and Image Generation with PYNQ

The second method of debugging involved the use of the PYNQ framework. After capturing the image data from the camera and storing it in BRAM, we used PYNQ to read and manipulate this data. This step was essential for verifying that the image data was correctly captured and stored in memory. By generating and analyzing the images, we could detect any discrepancies or errors in the data pipeline.

For instance, after addressing the pclk instability issue identified through ILA, we used PYNQ to read the BRAM contents and reconstruct the images. This allowed us to confirm that the changes made to the clocking and synchronization were effective. The reconstructed images were checked for consistency, noise levels, and overall quality, ensuring that the camera and FPGA were operating as expected.

6.3 Performance Measurement

The overall performance of the system was evaluated using Signal-to-Noise Ratio (SNR) and visual image quality. Initially, due to the instability of the pixel clock (Fig.4) and other problems in the design, the images captured from the OV7670 camera exhibited poor quality, characterized by higher noise levels and a lower average SNR. These images were difficult to analyze, as the instability in clock synchronization led to significant artifacts and distortions (Fig.5).

After addressing the problems, we observed a marked improvement in the captured images. The SNR of the images increased, indicating a reduction in noise and more accurate data capture (Fig.6). Higher SNR values are indicative of better image quality, as they reflect a clearer signal with less interference. We measured the SNR of images before and after the fixes and found an increase in the average SNR, confirming that the modifications had a positive impact.

In addition to SNR, we also assessed the visual quality of the images. Before the system improvements, images displayed notable visual defects, such as horizontal lines, blurred details, and inconsistent brightness. After implementing the corrections, the images showed improved sharpness, consistent color representation, and a reduction in visual artifacts. The overall quality of the images was greatly enhanced, confirming that the system was now capturing and processing data as intended.

Images captured before system optimization, exhibiting low SNR and significant visual artifacts:

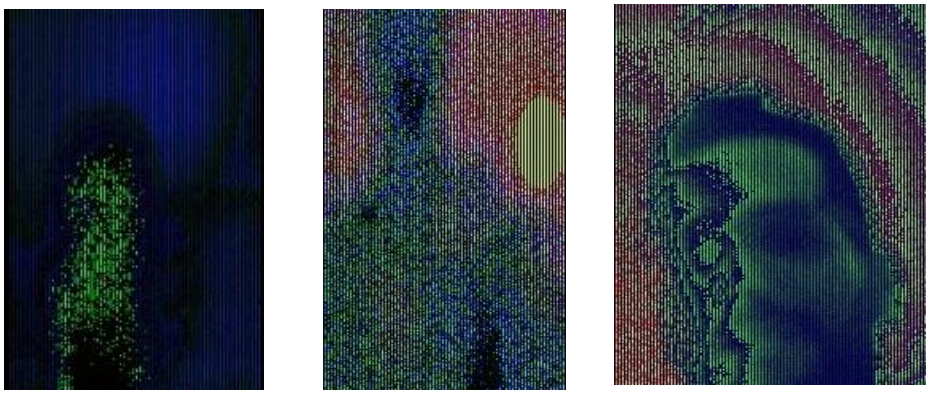

Figure 5: Low Quality Photos With average SNR = 0.83 dB

Images captured after system optimization, demonstrating higher SNR and improved visual quality with reduced noise and artifacts:

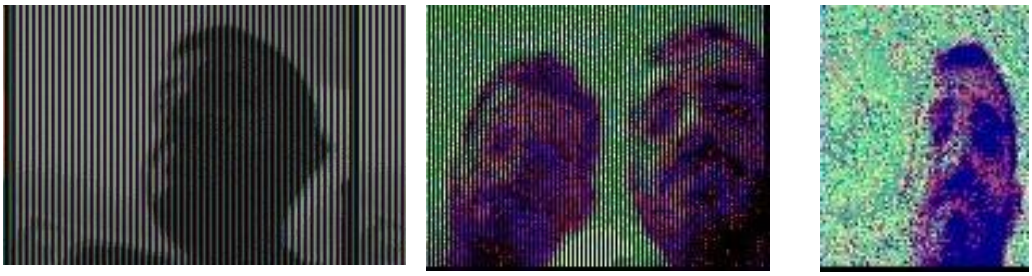

Figure 6: Higher Quality Photos With average SNR = 1.25 dB

These performance measurements, both quantitative (SNR) and qualitative (visual quality), validated the effectiveness of our debugging efforts and the stability of the final system design.

7. Problems and Their Solutions

Throughout the planning, execution, and operation of the project, several challenges arose that required creative problem-solving and adaptation. Some of these problems, along with their corresponding solutions, is outlined below.

1. Incompatible First Board

Initially, the project began with a board that was not suitable for our specific requirements. After spending significant time attempting to configure the system, it became clear that the board lacked the necessary features and compatibility for integrating the OV7670 camera.

After nearly a month of troubleshooting and failed attempts to achieve the desired results, the decision was made to switch to a more suitable board, the Ultra96-v2. This board provided the necessary processing power, compatibility, and flexibility required for the project. Although this change delayed the project timeline, it ultimately allowed for successful integration and operation of the camera system.

2. Overcomplicated Reference Design

The project initially relied on a reference design [8] that was overly complex and included many irrelevant modules, such as PWM and pan-tilt control, which were not required for our specific application. Additionally, the reference was designed for an older version of Vivado (2017), causing compatibility issues with the Vivado 2022 version we were using.
After struggling to adapt the outdated and complicated reference design, we made the decision to revert to the basics [5]. We stripped down the design to only include the essential modules needed for our project, such as the camera interface and memory management. This allowed us to build the project up from a solid foundation, leading to better understanding and more efficient development.

3. Challenges with Vivado ILA Integration

To debug the system effectively, we intended to use the Vivado Integrated Logic Analyzer (ILA) to monitor and analyze signal behavior. However, we encountered significant difficulties while working with the ILA in Vivado 2022.2. Specifically, when attempting to run the debug process within Vivado, the system clock (Zynq clock) was not powered, which in turn prevented the ILA from functioning correctly. The ILA relies on the system clock to capture signals, and without the clock running, we could not view or analyze the signals as intended.
To overcome this challenge, we devised a workaround involving the use of both Vivado and Vitis IDEs. First, we programmed the FPGA using Vitis IDE, which successfully initiated the system clock. Once the clock was running, we returned to Vivado to activate the ILA and capture the necessary signals. Although this process was more time-consuming and required navigating between two different IDEs, it allowed us to effectively debug the system and improve its performance by providing the necessary signal data for analysis.

4. Missing D5 Signal from Camera

During testing, we discovered that the data output signal D5 from the OV7670 camera was not being received, resulting in loss of 38400 pixels per frame (12.5% loss). Upon further investigation, we found that the corresponding port on the Pmod interface was not receiving any voltage, indicating a hardware issue.
With all available Pmod ports already in use, we faced the challenge of finding a workaround without losing functionality. We identified another port that was set to a constant value of 0, which was used for power configuration. By swapping this port with the faulty one, we were able to reroute the D5 signal and ensure proper data capture from the camera without compromising the system's overall functionality.

5. Sharing Pmod Interface with Another Team

Near the end of the project, when development was most intensive, another team working on a similar project encountered a complete failure of their Pmod interface, which was no longer receiving any voltage from their FPGA. This led to a situation where both teams had to share a single Pmod interface.
Sharing the Pmod interface significantly slowed down the development for both teams, as resources were now limited. To mitigate this, we carefully coordinated our work schedules with the other team to ensure that both projects could continue progressing. Although this required additional time and effort, effective communication and collaboration allowed us to overcome this challenge and successfully complete our project.

8. Work Summary

This project successfully developed a modular and efficient infrastructure for integrating the OV7670 camera with the Ultra96-v2 board, enabling the capture, processing, and analysis of image data for neural network applications. The work involved configuring the FPGA using Vivado and Vitis IDEs, developing essential modules for camera control and data storage, and implementing a system to accurately capture images from the camera and store them in Block RAM (BRAM).

Through careful debugging using Vivado ILA and data manipulation via PYNQ, the project achieved a reliable and functional system capable of generating images suitable for AI applications [2]. The system was designed with flexibility in mind, allowing for future enhancements and modifications to meet various project requirements.

8.1 Recommendations for a follow-up project

Camera Upgrade: Consider upgrading to a higher-resolution camera with better optics to improve the overall image quality and increase the SNR further. This could allow for more advanced neural network applications, particularly those requiring detailed image analysis.

Advanced Filtering: Implement more sophisticated noise filtering and image enhancement techniques within the FPGA to further improve the visual quality of the images captured.

Adopt DMA for Data Transfer: Implement Direct Memory Access (DMA) for transferring image data from the camera to memory. DMA would significantly increase the system's efficiency by offloading data transfer tasks from the processor, allowing for faster and more reliable data handling, particularly in real-time applications.

Increase System Scalability: Expand the current design to support multiple cameras or additional sensors. This would allow the system to be used in more complex AI applications, such as autonomous vehicles or advanced surveillance systems, where data from multiple sources need to be processed simultaneously.

8.2 Advantages and Disadvantages

The system achieves a balance between performance and cost by utilizing affordable components such as the OV7670 camera and the Ultra96-v2 board. This makes it accessible to a wide range of users, including developers and researchers with limited budgets. Despite its cost-effectiveness, the system is capable of real-time image processing, a crucial requirement for many AI applications. This capability is enhanced by the FPGA's inherent parallel processing power, allowing for efficient handling of data-intensive tasks while maintaining low power consumption.

On the other hand, one of the primary limitations of the current system is the image quality produced by the OV7670 camera. While sufficient for basic applications, the low resolution and limited dynamic range of the camera restrict its use in scenarios requiring high-fidelity image capture. The system's potential is somewhat constrained by this hardware limitation.

8.3 Achievement of Project Goals

The primary goal of this project was to develop a modular and efficient infrastructure that could integrate a low-cost OV7670 camera with the Ultra96-v2 board to capture and process image data for neural network applications. This goal was achieved. We were able to design and implement a system that successfully captured images using the camera, processed them in real-time, and stored the data in a format suitable for further analysis. The final system demonstrated the ability to generate recognizable images, including those of our faces, which was a key indicator of success. Despite the fact that the image quality was not optimal, with some limitations in resolution and dynamic range, the core functionality of the system was verified. The images captured were clear enough to demonstrate that the camera integration and data processing pipeline were working as intended.

8.4 Planning Versus Execution

The project schedule was carefully planned to accommodate both development and debugging phases. However, several challenges, such as the initial board incompatibility and the complexity of the reference design [8], led to delays. These issues required us to spend additional time on problem-solving and system optimization, extending the project timeline beyond what was originally planned. Despite these setbacks, the project was completed within a reasonable timeframe, with the majority of delays being managed effectively to minimize their impact on the final delivery.

8.5 Budget Estimation

Hardware equipment:

Name	Cost	Purchase/ lab equipment
Xilinx Ultra96v2 Board including USB-to-JTAG /UART pod and external power supply kit	585$	Existing lab equipment
Pmod	64$	Existing lab equipment
32GB SD Card	7.5$	Purchased
MicroUSB Data Cable	5$	Purchased
OV7670 Camera	4.5$	Existing lab equipment
18 Jumper Cabels	3$	Purchased

Table 2: Hardware Equipment Budget

Software and license:

Name	Cost	Purchase/ lab equipment
Vivado Software Platform 2022.2		Previously obtained by the lab

Table 3: Software Equipment

9. References

[1] E. Manor and S. Greenberg, "Using HW/SW Codesign for Deep Neural Network Hardware Accelerator Targeting Low-Resources Embedded Processors," in *IEEE Access*, vol. 10, pp. 22274-22287, 2022, doi: 10.1109/ACCESS.2022.3153119.

[2] E. Gholizadehazari, T. Ayhan and B. Ors, "An FPGA Implementation of a RISC-V Based SoC System for Image Processing Applications," 2021 29th *Signal Processing and Communications Applications Conference (SIU)*, Istanbul, Turkey, 2021, pp. 1-4, doi: 10.1109/SIU53274.2021.9477998.

[3] A. S. Deulkar and N. R. Kolhare, "FPGA implementation of audio and video processing based on Zedboard," *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, Aurangabad, India, 2020, pp. 305-310, doi: 10.1109/ICSIDEMPC49020.2020.9299639.

[4] "Ultra96-V2 Getting Started Guide," *96Boards*. https://www.96boards.org/documentation/consumer/ultra96/ultra96-v2/getting-started/

[5] L. V. Sandi, "ZYBO OV7670 to VGA," *Lauri V. Sandi's HDL Projects*. https://lauri.xn--vsandi-pxa.com/hdl/zynq/zybo-ov7670-to-vga.html

[6] "Ultra96-PYNQ Getting Started Guide," *PYNQ Documentation*. https://ultra96-pynq.readthedocs.io/en/latest/getting_started.html

[7] "Basys 3 FPGA OV7670 Camera," *FPGA4Student*. https://www.fpga4student.com/2018/08/basys-3-fpga-ov7670-camera.html

[8] "ZYBO OV7670 Camera With Pantilt Control," Instructables. https://www.instructables.com/ZYBO-OV7670-Camera-With-Pantilt-Control/

10. Appendix

Tutorials

We utilized the datasheets for the OV7670 camera and the Ultra96-v2 board to guide our work with these components:

Avnet, "Ultra96-V2 Hardware User Guide," v1.3, 2020. https://d7rh5s3nxmpy4.cloudfront.net/CMP7377/files/Ultra96-V2-HW-User-Guide-v1_3.pdf

OmniVision Technologies, Inc., "OV7670 CMOS VGA (640x480) Image Sensor," 2006. https://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf

For access to all the relevant project files, including Vivado, Vitis, Python scripts, and XSA files, please refer to the following link: Project Submission

המלצת ציון לדו"ח מסכם

המלצת ציון לדו"ח מסכם

<u>אם יש צורך, לכל סטודנט/ית בנפרד</u>

מספר הפרויקט: P-20____-_____

שם הפרויקט:

שם המנחה החיצוני:

שם המנחה מהמחלקה:

שם הסטודנט/ית: ת.ז.:

	חלש	בינוני	טוב	ט"מ	מצוין
הצגת גישת הפתרון, והתכנון ההנדסי					
הצגת התוצאות וניתוח השגיאות					
הסקת מסקנות					
גילוי יוזמה וחריצות					
פתרון בעיות, מקוריות ותרומה אישית (מעבר למילוי ההנחיות)					
עמידה בלו"ז ורמת הביצוע המעשי					

אם יש כוונה לפרסם / יפורסם מאמר, שם כתב העת ומועד משוער להגשה:

ציין אם יש כוונה לשקול המלצה כפרויקט מצטיין:

הערות נוספות: