Pipelined Hardware Video Compressor & Decompressor

Design Document

Team Number: 12
Client: John Deere
Faculty Advisor: Professor Zambreno
Team Members/Roles:
Colsen Selk, Algorithms. Kareem Eljaam, FPGA. Caleb Rock, FPGA. Ben Meinders, Algorithms. Logan Mcdermott, FPGA.
Email: sdmay24-12@iastate.edu
Website: https://sdmay24-12.sd.ece.iastate.edu/

Revised: 9/8/2023 V1.0

Executive Summary

The goal of this project is to provide a near-zero latency, fully pipelineable image compression scheme that can be mapped to an FPGA or ASIC. As of now, there is nearly zero latency performing convolutions on incoming data for image processing, but the amount of RAM is limited. We are looking to provide a very lightweight compression and decompression scheme that can be implemented at the input and output of each convolutional line buffer to reduce the amount of RAM needed. The end goal is to provide an RTL implementation on a Zynq Dev Board, and to deliver a demo with HDMI video into an FPGA, compression and decompression being performed on the live input stream. Ideally, a timer or clock will be displayed on the screen that provides real time latency information.

Development Standards & Practices Used

Hardware:

- Maintainability: Ensure modularity to uphold maintainability.
- Efficiency: Should not overuse resources.

Software Practices:

- Reliability: code should work as close to 100% as possible
- Maintainability: code should be well documented, and up to format for modern day software for best possible readability
- Efficiency: code should be the most efficient solution (lowest Big-O possible)

Summary of Requirements

- Research different algorithms that could be used to deal with the problem
- List of ideas on how to solve the problem
- Implement the software based on the most promising idea
- Provide an RTL or HLS implementation on a zynq dev board
- Deliver a live demo with an HDMI video into an FPGA with a clock displayed

Applicable Courses from Iowa State University Curriculum

- COM S 311
- SE 329

- CPRE 381
- CPRE 281
- COM S 309

New Skills/Knowledge acquired that was not taught in courses

Learning algorithms based around video compression and decompression.

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Team

1.1 Team Members

Kareem Eljaam, Caleb Rock, Logan Mcdermott, Ben Meinders, Colsen Selk

1.2 Required Skill Sets for Your Project

Python, VHDL, FPGA Design, knowledge of software algorithms

1.3 Skill Sets covered by the Team

Python: Colsen Selk, Ben Meinders, Kareem Eljaam

VHDL: Kareem Eljaam, Caleb Rock, Logan McDermott

FPGA: Logan McDermott, Kareem Eljaam, Caleb Rock

Algorithms: Colsen Selk, Kareem Eljaam, Ben Meinders

1.4 Project Management Style Adopted by the team

The manager will ensure project requirements are being fulfilled on track with the project timeline. Tasks will be facilitated to members weekly at team meetings and monitored via GitLab.

1.5 Initial Project Management Roles

Colsen Selk, Algorithms. Kareem Eljaam, FPGA, manager. Caleb Rock, FPGA. Ben Meinders, Algorithms. Logan Mcdermott, FPGA. All members are responsible for communicating with each other regarding where the different areas of the project meet.

2 Introduction

2.1 Problem Statement

What problem is your project trying to solve? Use non-technical jargon as much as possible.

To create a realtime video compression and decompression system while keeping minimal loss in order to keep memory to a minimum.

2.2 Requirements & Constraints

List all requirements for your project . This includes functional requirements (specification), resource requirements, qualitative aesthetics requirements, economic/market requirements, environmental requirements, UI requirements, performance requirements, legal requirements, maintainability requirements, testing requirements and any others relevant to your project. When a requirement is also a quantitative constraint, either separate it into a list of constraints, or annotate at the end of requirement as "(**constraint**)". Other requirements can be a single list or can be broken out into multiple lists based on the category.

- Part 1: Software Implementation
    - Benchmark tradeoffs for various lightweight compression algorithms in Python
        - Considerations:
        - Latency VS Compression Amount
        - How much data is lost in each lossy compression algorithm?
- Part 2: Hardware Implementation
    - Minimize cost and complexity of FPGA
    - Maintain near zero latency of compression and decompression
    - The FPGA should be fully-pipelineable
- Part 3: Demo
    - Have a video (prior to compression) being streamed and an additional video being streamed from the FPGA showcasing video after being compressed and decompressed

2.3 Engineering Standards

What Engineering standards are likely to apply to your project? Some standards might be built into your requirements (Use 802.11 ac wifi standard) and many others might fall out of design. For each standard listed, also provide a brief justification.

HDMI for transmitting the video from input and to output.
Compression algorithms such as M-JPEG, MPEG-4, and H. 264 for the video compression systems.
python for basic prototyping of the compression algorithms, in order to find the best compression system for the job.

2.4 Intended Users and Uses

Who benefits from the results of your project? Who cares that it exists? How will they use it? Enumerating as many "use cases" as possible also helps you make sure that your requirements are complete (each use case may give rise to its own set of requirements).

John Deere will use this project inside their equipment to reduce the cost they have to spend on memory. John Deere wants to use the system to make their computer vision systems more efficient. Reducing the memory usage could also have the factor of reducing processing time of their machine learning algorithms.

3 Project Plan

3.1 Task Decomposition

- Part 1: Software Testing Implementation in Python
    - Research some well established algorithms for video compression and decompression
        - Choose 3-5 (or more) Algorithms to thoroughly benchmark the following metrics:
            - Compression Ratio
            - Latency
            - Complexity
    - Find a dataset for testing the compression and decompression
        - Considering that the project demonstration will use HDMI and many development boards have HDMI controllers that convert the data to RGB24, find some input data we can use to test our algorithm that matches the RGB24 protocol.
    - Write the Python implementation that benchmarks the algorithms chosen beforehand
        - Based on the performance of the algorithms we test, we will choose an algorithm that we can implement in hardware
        - Explore parameterizing algorithms so that a knob could be used in the demonstration to control the latency vs compression ratio relationship.
- Part 2: FPGA Implementation + Demonstration
    - Choose a development board to implement the algorithm on for the demonstration
        - Note: HDMI Sink + Source and FPGA large enough to support the algorithm are the requirements for this developmental board
    - Create a plan/schematic for a pipelined FPGA solution for the selected algorithm
    - Implement the compression and decompression Algorithm on the FPGA and benchmark the compression ratio and latency
    - Gather the necessary hardware for the demonstration
        - 2 HDMI Monitors
        - 1 Development Board
        - 1 HDMI Splitter
        - 2 HDMI Cables
    - Final debugging and testing of the entire system working together

3.2 Project Management/Tracking Procedures

We will be using the agile management style for software development. We still intend on having a couple meetings a week and can use these meetings as SPRINTs for planning project goals/outcomes for the next meeting.

Gitlab will track our issues when we work on the python elements of the project. We can also use the issues board to track progress on FPGA pipelining.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Having a list of applicable video compression algorithms.
- Creating a framework to test video compression algorithms in python, and using that to pick our final compression algorithm.
- Having a design laid out of the pipeline
- Having the FPGA finalized
- In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprint).

3.4 Project Timeline/Schedule

3.5 Risks And Risk Management/Mitigation

Python implementation will have low risks (0.2) because python compression libraries exist and are pretty well established.

Having a design of the FPGS will have high risks because putting a compression algorithm to hardware can be a complex, tedious, and abstract (0.5)

- choose an algorithm with less complexity and with plenty of documentation to make the implementation easier

Creating a list of applicable algorithms will be pretty low risk (0.05) because our requirements for picking algorithms are relatively simple: must be not too complex, must have low latency, must be implementable in hardware.

An agile project can associate risks and risk mitigation with each sprint.

3.6 Personnel Effort Requirements

Task	Man-Hours	Explanation
Algorithm Research	4	Spend enough but not too much time finding the most optimal algorithms to run on FPGA.
Python Benchmarking Implementation	10	Creating a Python program that can benchmark the different algorithm prototypes.
Python compression algorithms	20	Implementing various video compression/decompression algorithms in Python that take in a video as an input.
Research and Choose a Dev Board For The Demonstration	1-2	Choose a dev board that has HDMI sink & source, preferably one made for video processing
Create a Plan/Schematic for the FPGA based on the software algorithm benchmark	10	Based on the software implementation of the algorithm, make a plan for the architecture of the FPGA
Implement the Compression part of the FPGA	20	Write the VHDL code to compress an input stream of raw RGB24 values.
Testing/Debugging of Compression Algorithm on FPGA	10	Test some expected values including edge cases, debug accordingly until fully functional
Implement the Decompression part of the FPGA	20	Write the VHDL code to decompress an input stream of compressed RGB24 values.
Testing of Decompression Algorithm on FPGA	10	Test some expected values including edge cases, debug accordingly until fully functional
Gather Hardware Needed for the Demo	1-2	Make sure we have all required hardware for the demo whether that be from Deere or the TLA
Final Debugging of the Entire System	10	Having all the hardware for the demo set up and making

		sure everything works together.

3.7 Other Resource Requirements

Potentially need a camera for live video.

4 Design

4.1 Design Content

The design content of our project includes designing an efficient approach to implement a video compression and decompression algorithm on the Zynq 7000 FPGA. This requires having clear hardware designs/schematics that portray the functionality of the video pipeline on the FPGA.

4.2 Design Complexity

The design of a video compression system involves creating custom hardware and custom software, and requires them to cohesively work together along with given IP circuits that we must also choose. We must design video compression algorithms that work extremely fast, and have it interface with its associated hardware that we've designed. The hardware that we design will be very complex due to the nature of video compression and decompression. Hardware designs also take into account all the limitations and constraints that are ignored in software design.

4.3 Modern Engineering Tools

Python will be used for the software side for video compression and decompression just to test algorithms. We will then have to write C code that does video compression. Some aspects of the video compression will be done in hardware, which at the easiest, requires the code to be designed in C and an API used to convert to hardware, and at the hardest, requires Verilog to be written for the hardware. Draw.io is a tool that will be used for drawing schematics and block level diagrams.

HARDWARE TOOLS HERE (if any were missed)

4.4 Design Context

Area	Description	Examples
Public health, safety, and welfare	This project can potentially indirectly affect the safety or well-being of a John Deere customer in the future since the video compression/decompression can be used for videos running on vehicle displays.	Having lossy images/videos in a tractor/sprayer/etc. can provide a major risk of the vehicle malfunctioning.
Global, cultural, and social	This project could have applications in increasing crop yield for tractors. This project should decrease the cost of hardware inside tractors, allowing farmers to save money when they have to buy new tractors. This would lead to a reduction in the cost of food and crops.	An efficient implementation will satisfy the client and potential customer's aim of having a cheaper and more advanced solution.
Environmental	Having our client as John Deere may help illustrate the kind of environmental impact our project may have. Since tractors aren't fully electric, they do pollute, but the idea of our project is to help tractors run more efficiently, therefore reducing the time of use.	Our project being in the realm of precision farming means that as an engineer, we are trying to innovate the way tractors are traditionally operated, such that the added use of software, and hardware is able to automate tasks, increase yield, and decrease time allotted to various tasks.
Economic	This project will save money for both John Deere and farmers spending money on new tractors. This equipment might interface with technology that increases crop yield, and increases autonomy. This saves farmers money, increases supply on crops, and thus decreases the price of crops.	Our solution being implemented on an FPGA leads to a decrease in cost for John Deere since it'll be faster than a fully software approach and will off-load from the CPU. This results in freeing up more space on the CPU for other tasks.

4.5 Prior Work/Solutions

There exists dedicated hardware video compression chips. These however, are mainly only found on GPU's, which require a lot of power, are expensive, and they are not dedicated to one singular task like our hardware will be. There also exists video compression algorithms and encoders like H265. Video compression techniques are also a thoroughly researched topic.

4.6 Design Decisions

We've kept in mind complexity and cost when designing the layout of the hardware. We've also decided we will use the Zybo Z7-10 as our dev-board because it has both HDMI-in and HDMI-out. We've chosen to use VDMA in and out, Memory buffer as storage, CPU to interface with the hardware and possibly do some software compression with. We will also have our own custom hardware designs. We've chosen to have a CPU so that we can send information to memory, and we

can interact with the hardware if needed, and so future engineers can interface with the CPU to receive the information to do other things with.

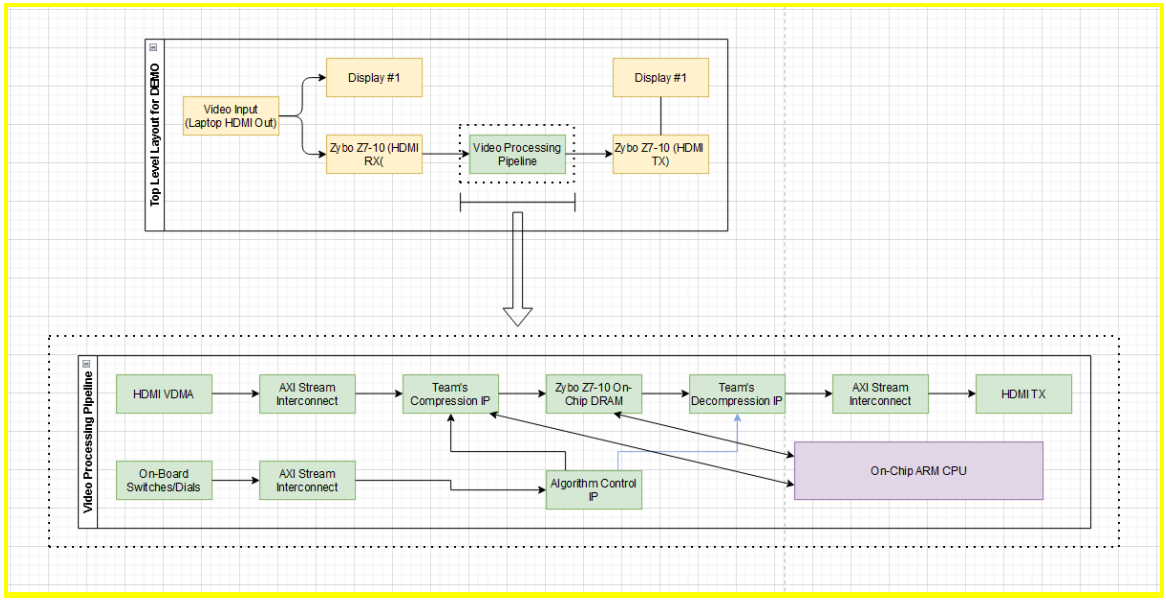
4.7 Proposed Design

We've created a basic hardware diagram for how we want to lay out and arrange the components of our hardware. We've researched compression algorithms and techniques to be considered for implementation in hardware. We've gathered dev-boards to be used for testing and implementation.

4.7.1 Design 0 (Initial Design)

Design Visual and Description

The Zybo Z7-10 converts the data on its HDMI RX line to allow for HDMI VDMA. This is how we will be accessing the HDMI data to run compression on. Once we have the VDMA data, we can compress it and send it to memory provided by the on-board DRAM. Once in memory, here is where we can leverage use of the on-board ARM processor to gather details on latency, compression ratio, memory savings, and loss amount for the algorithm. The Memory will store a compressed frame of the video to be accessed by other hardware and software out of the scope of this project. Then the compressed data will be taken out of memory and decompressed to provide a video. Another goal that we have is to view the tradeoffs between speed vs compression ratio vs video quality by allowing these metrics to be adjusted with some sort of dial or switch control interface. The first design states that we will be using the on-board slide switches but in the future, a separate dial peripheral could be added.



Functionality

Our design is intended to operate on John Deere equipment to minimize on device storage. All of the cameras on a tractor, sprayer, planter, combine harvester will need to have their video compressed with near-zero latency so that they do not require huge amounts of memory. The current design will hypothetically have near-zero latency and output lossless or near-lossless video after decompression thus satisfying the functional requirements. the hardware will temporarily keep the compressed video data in memory for future use.

NOTE: The following sections will be included in your final design document but do not need to be completed for the current assignment. They are included for your reference. If you have ideas for these sections, they can also be discussed with your TA and/or faculty adviser.

4.7.2 Design 1 (Design Iteration)

Include another most matured design iteration details. Describe what led to this iteration and what are the major changes that were needed in Design 0.

Design Visual and Description

Include a visual depiction of this design as well highlighting changes from Design 0. Describe these changes in detail. Justify them with respect to requirements.

4.8 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

4.9 Design Analysis

- *Did your proposed design from 4.7 work? Why or why not?*
- *What are your observations, thoughts, and ideas to modify or iterate further over the design?*

5 Testing

*Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.*

The testing plan should connect the requirements and the design to the adopting test strategy and instruments. In this overarching introduction, given an overview of the testing strategy. Emphasize any unique challenges to testing for your system/design.

5.1 Unit Testing

What units are being tested? How? Tools?

- Software Compression Algorithms
    - We will verify that the output after compressing and decompressing data matches the input. This will be done with a Python program to compute the output, and a diff tool if needed to see how the output and input differ. We should also verify that the algorithm is reducing the amount of storage needed to store data.
- Compression IP
    - The initial software testing will give us a way to validate that our intermediate values in hardware are correct. The compression IP will be tested using Vivado tools, HDL testbenches, and eventually tested physically during system level testing.
- Decompression IP
    - The decompression IP will operate very similarly to the Compression IP. It will also have to pass tests from Vivado and HDL testbenches to verify that it is converting compressed input back into the original output.

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

- Hardware Interface(Zybo board): Using Vivado Synthesis, and Implementation
- Software Interface: Binary/Opcode files (pre-compiled files) (C/C++)

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

The first step in our project is to create an HDMI pass-through demonstration. This will take an HDMI source, send it through the FPGA and output it to an HDMI display. To do this, we will start by seeing if we can get data from HDMI RX on the board in the FPGA. We will verify this using Vivado. Via unit testing we should be able to confirm that Compression and Decompression IPs work correctly so the last step is to make sure we can display the output of the Decompression IP on a display via HDMI. We will test this by verifying an image is displayed to the HDMI display.

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

To test the entire system we plan on running a video stream through an HDMI cable that is plugged into the Zybo board. The video would go through an FPGA pipeline where it is compressed and stored in memory, afterwards there will be a decompressor in the pipeline that will take the compressed data and decompress it. The video will then go out through an HDMI cable into a monitor. Tests for this would include the Compression/Decompression IP unit tests, HDMI integration test, and interface tests for hardware and software.

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure they do not break? Is it driven by requirements? Tools?

A tool that we will be using to ensure that new additions don't negatively impact old ones is the use of git version control, and code reviews. Using these features will ensure that we never break our main source of code (main branch), and that new code will have to be reviewed by at least two people in order to get merged. Additionally, before any new changes are merged to the main branch, it will have to hit certain standards through the pipeline such as code coverage in order to be merged. Assuming that all of the new, and old tests are passing, should be enough to validate that the code is safe to merge into the main branch.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

In order to check whether the design requirements are met, and have been completed to an acceptable level by the client's standards, is to eventually get the software to run in a live demo, where an overlay will be brought over the video stream, to show different values such as data loss, and compression rate. If we run out of time during the development period, and don't have ample time to develop an overlay, we could still create software that outputs these values to a log file, where we can verify that the software is working as intended.

5.7 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

6 Implementation

7 Professionalism

8 Closing Material