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

This project aims to provide a low-latency, fully pipelined image compression scheme that can be mapped to an FPGA. Currently, there is nearly zero latency in 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 goal is to provide an RTL implementation on a Zynq 7000 and deliver an HDMI video into an FPGA, with compression and decompression performed on the live input stream. Ideally, a timer or clock will be displayed on the screen, providing 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 the 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
- CPRE 488

New Skills/Knowledge acquired that was not taught in courses

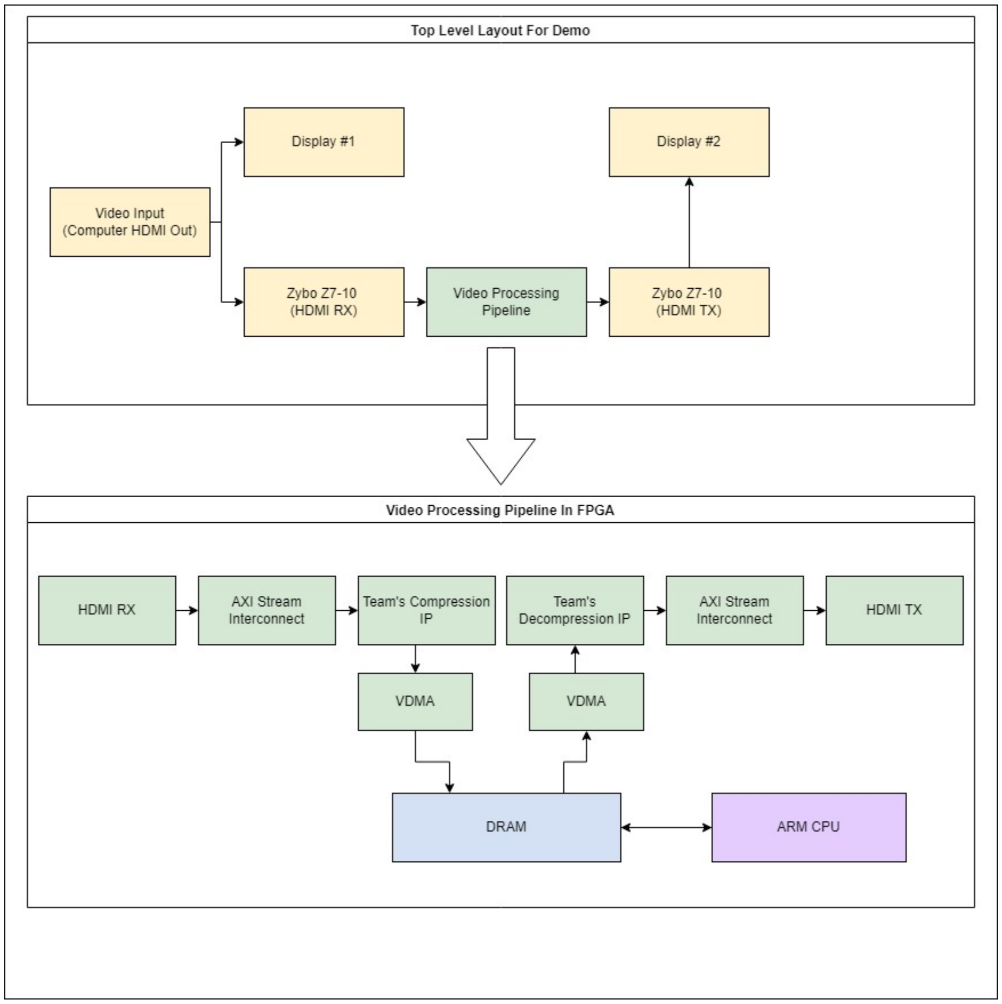
Learning algorithms based on video compression and decompression.

- Dictionary Algorithms (LZW)
- Wavelet

Video compression/encoding pipeline format and their respective IPs/components.

Table of Contents

List of figures/tables/symbols/definitions (This should be 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

Our problem statement is to create a real-time video compression and decompression system on an FPGA with minimal loss to reduce memory utilization.

2.2 Requirements & Constraints

- 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-pipelined.
- 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

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

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

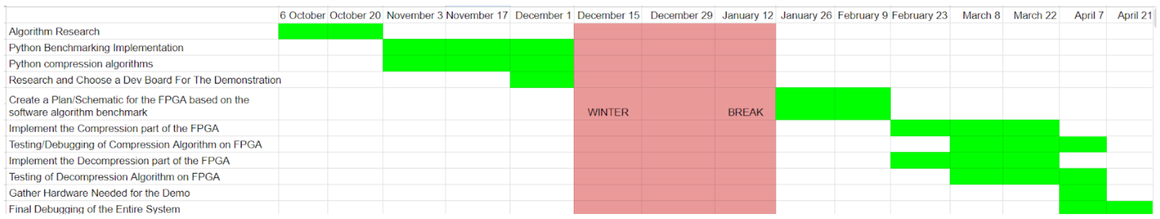
We will be using the agile management style for software development. We still intend on having a couple of meetings a week and can use these meetings as SPRINTs for planning project goals/outcomes for the next meeting.

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 sprints).

3.4 Project Timeline/Schedule



	6 October	October 20	November 3	November 17	December 1	December 15	December 29	January 12	January 26	February 9	February 23	March 8	March 22	April 7	April 21
Algorithm Research	🟩	🟩				🟥	🟥	🟥							
Python Benchmarking Implementation			🟩	🟩	🟩	🟥	🟥	🟥							
Python compression algorithms			🟩	🟩	🟩	🟥	🟥	🟥							
Research and Choose a Dev Board For The Demonstration					🟩	🟥	🟥	🟥							
Create a Plan/Schematic for the FPGA based on the software algorithm benchmark						WINTER	🟥	BREAK	🟩	🟩					
Implement the Compression part of the FPGA						🟥	🟥	🟥			🟩	🟩	🟩		
Testing/Debugging of Compression Algorithm on FPGA						🟥	🟥	🟥				🟩	🟩	🟩	
Implement the Decompression part of the FPGA						🟥	🟥	🟥			🟩	🟩	🟩		
Testing of Decompression Algorithm on FPGA						🟥	🟥	🟥				🟩	🟩	🟩	
Gather Hardware Needed for the Demo						🟥	🟥	🟥						🟩	
Final Debugging of the Entire System						🟥	🟥	🟥						🟩	🟩

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 complex, tedious, and abstract (0.5)

- Choose an algorithm with less complexity and 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: they must not be too complex, must have low latency, and 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	Create a program to benchmark the different algorithm prototypes.
Python compression algorithms	20	Implementing various video compression/decompression algorithms in Python that take in a video as input.
Research and Choose a Dev Board For Demonstration	1-2	Choose a dev board with an HDMI sink and 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, and 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, and debug accordingly until fully functional.
Gather Hardware Needed for Demo	1-2.	Ensure we have all the required hardware for the demo, whether from Deere or the TLA
Final Debugging of the Entire System	10.	Having all the hardware for the demo set up and ensuring everything works together.

3.7 Other Resource Requirements

Zybo Z7-10 Development Board, appropriate Vivado software package for Zynq 7000, 2 monitors, 2 HDMI cables, and a laptop for an HDMI source

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 interface with the associated hardware we have 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.

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 hardware cost of tractors, allowing farmers to save money when buying 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, John Deere, may help illustrate the environmental impact our project may have. Since tractors are not 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 is 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 can 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 the supply of crops, and thus decreases the price of crops.	Our solution is implemented on an FPGA leads to a decrease in cost for John Deere since it will 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. However, these are mainly only found on GPUs, which require much power, are expensive, and 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 have kept in mind complexity and cost when designing the hardware layout. We have also decided to use the Zybo Z7-10 as our dev-board because it has both HDMI-in and HDMI-out. We have chosen to use VDMA in and out, Memory buffer as storage, CPU to interface with the hardware, and possibly do some software compression. We will also have our custom hardware designs. We have chosen to have a CPU so that we can send information to memory, interact with the hardware if needed, and so future engineers can interface with the CPU to receive the information to do other things.

4.7 Proposed Design

We have 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.

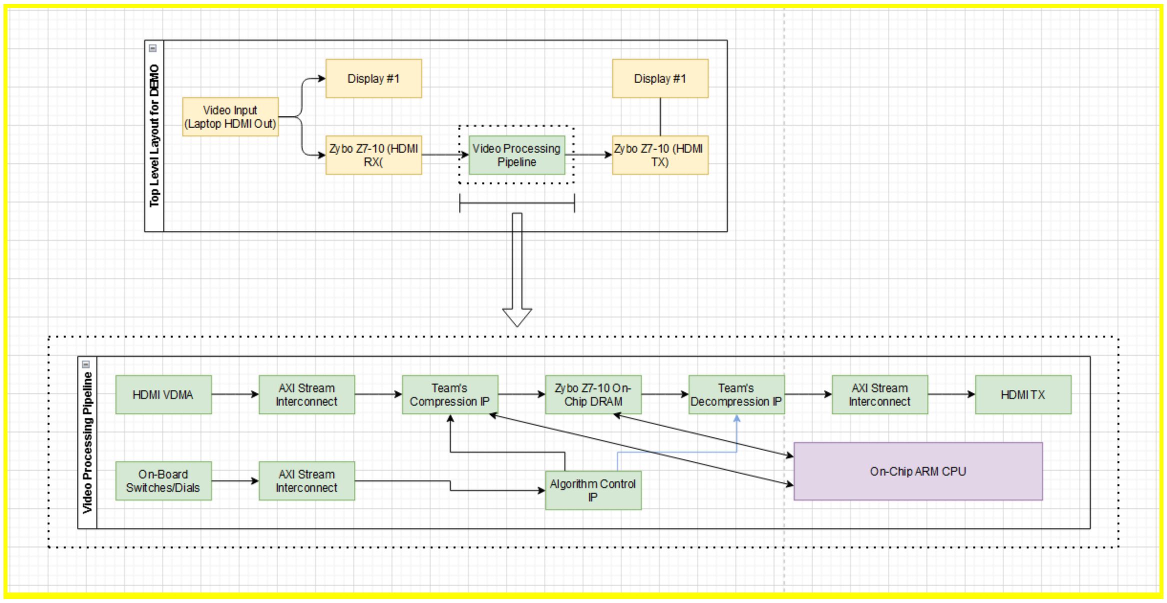


FIgure 1

Functionality

Our design is intended to operate on John Deere equipment to minimize storage utilization for image/video. All of the cameras on a tractor, sprayer, planter, or combine harvester will need to have their video compressed with low latency to reduce the amount of memory required. After decompression, the current design will hypothetically have near-zero latency and output lossless video data, 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)

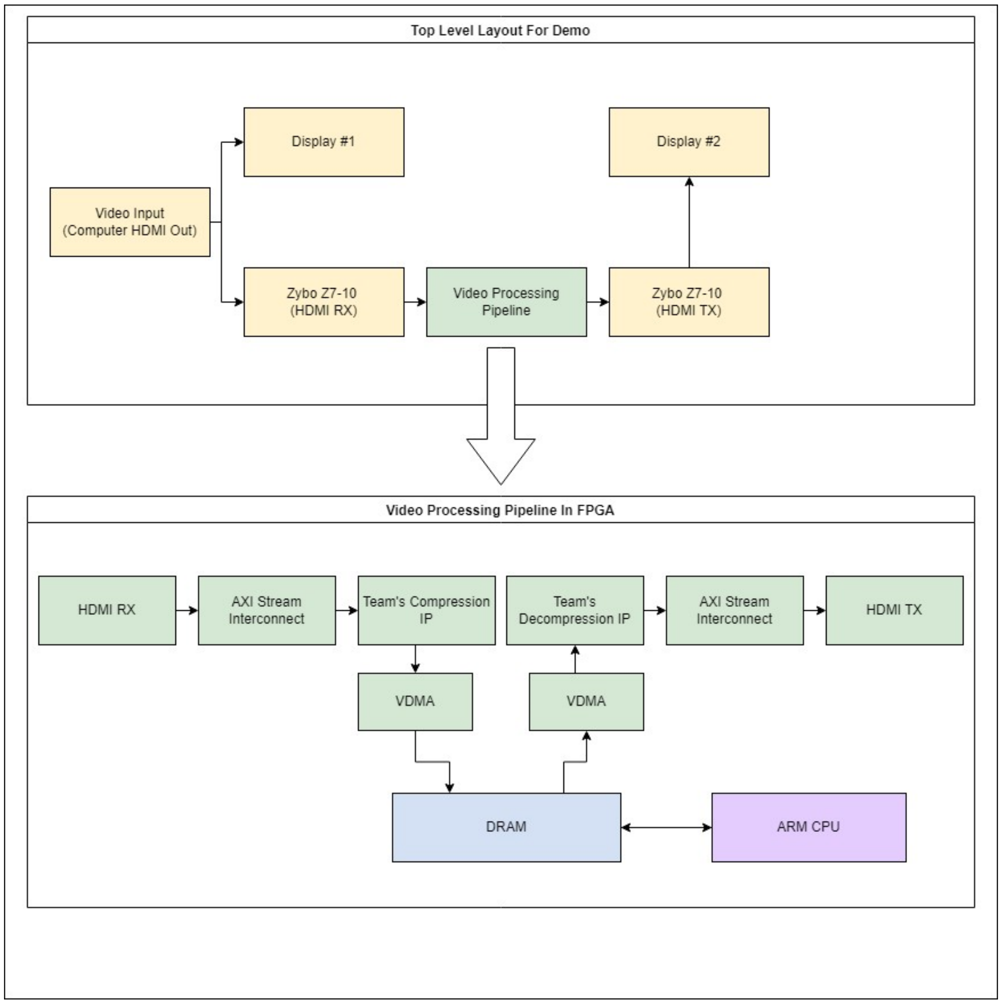


FIgure 2

Design Visual and Description

The design shown above in figure 2 is an updated version of our original design. After discussing our original design with our client, we realized that it needed some improvements. We got a better understanding of where the VDMA and memory (DRAM) come into play, so we adjusted those in the schematic.

4.8 Technology Considerations

We chose the Zybo Z7-10 board for a few reasons. First, we already verified with ETG that there was one available for us to use during development. Another advantage to using this development board is that it already provides HDMI Transmit and HDMI Receive signals to the FPGA. The Zybo Z7 also provides us with a Zynq 7000 board from Xilinx. The lab computers at Iowa State already support Vivado for development of Xilinx boards. Vivado is a powerful tool used for FPGA development, and will provide the team with many useful utilities. The Zynq 7000 not only provides us with the necessary FPGA resources for this project, but also access to a ARM CPU allowing for various different possibilities for the demonstration such as tracking performance metrics.

One disadvantage of using the Zybo Z7-10 is that it is an older development board, and newer development boards are more powerful and provide more for the developers to work with. However, this tradeoff is suitable for the team because the solution we are looking to provide is a lightweight one meaning we do not need the extra size from newer chips. The other board we were considering was the Zedboard. It provides the same FPGA, but does not have both HDMI Receive and HDMI Transmit so we ultimately chose the Zybo-Z7.

4.9 Design Analysis

Upon further analysis of our design, one potential improvement to be made for future iterations is the ability to provide the user to control tradeoffs for compression and decompression algorithms. For example, a knob could be configured to increase compression ratio, but in return provides a worse latency for the encoding and decoding process. Displaying real-time performance metrics during the demo is also a stretch goal that could be implemented in a variety of different ways such using bitmaps and hardware to display overlays.

5 Testing

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

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

5.3 Integration Testing

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

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

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

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

The results of testing on algorithms via Python and C has shown us which algorithms can be pipelined down to minimal clock cycles, and which algorithms are low-latency. Checking out various dev-boards has also narrowed down which dev-board we will use for implementation to the Zybo Z7-10

6 Implementation

Our client mentioned in the initial meeting that we should spend most of our time in the first semester creating a well laid out plan for the second semester, and to make sure we have sufficient details to allow us to work through the development phase, with as few potholes as possible regarding missing information, or lack of constraints. That being said, we have invested a lot of time this semester to make sure that we can follow our gantt chart schedule, and can keep moving the project forward at an acceptable rate. The implementation plan is simply to follow the gantt chart which has enough detail and simplicity to follow.

7 Professionalism

7.1 AREAS OF RESPONSIBILITY

Work Competence - The ACM Code of Ethics directly addresses work competence in statement 2.6 which states, “Perform work only in areas of competence.” This means that one must have a solid understanding of what they are creating so that people can trust their work. There is virtually no difference between the ACM and the NSPE versions here.

Financial Responsibility - The closest that the ACM Code of Ethics gets to this point is in statement 1.1 which states, “Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.” The ACM code only indirectly gets at this point that we are to deliver our works at reasonable costs. There is a pretty big difference in that the ACM code virtually does not mention this point at all.

Communication Honesty - The ACM Code of Ethics also takes this point very seriously. They state in point 1.3, “Be honest and trustworthy.” This means that one is to avoid deception and be honest and clear with what they are doing and what they know. The ACM code goes even deeper into this subject in being clear to avoid deliberate misleading claims.

Health, Safety, Well-Being - The ACM Code of Ethics is littered with statements that further draw out implications of this NSPE concept. Statements 1.1 and 1.2 say, “Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.” and “Avoid harm.”, respectively. There are more statements in the ACM code that explore this concept but these two most closely relate to the NSPE’s explanation.f

Property Ownership - The ACM Code of Ethics states in point 1.5, “Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.” The ACM and NSPE statements are talking about honoring the work people have done and giving credit where credit is due. The ACM’s explanation goes more into the why behind this statement in acknowledging that people should be allowed to gain value from the work they have done.

Sustainability - In statement 3.2 of the ACM Code of Ethics it says to “articulate, encourage acceptance of, and evaluate fulfillment of social responsibilities by members of the organization or group.” This means that one ought to care about society as a whole which touches on sustainability. Statement 1.2 as mentioned above, which states, “Avoid harm., also touches on the idea of sustainability. In the end, though, the ACM code does not explicitly emphasize sustainability like the NSPE does.

Social Responsibility - As stated directly above, statement 3.2 of the ACM Code of Ethics touches right on the idea of social responsibility. Both codes press on the idea that one ought to aim for the benefit of society over all else. The ACM code has even more statements that can be tied to this idea.

7.2 PROJECT SPECIFIC PROFESSIONAL RESPONSIBILITY AREAS

Work Competence (High) - Since this is a group project, we should expect to have to produce high quality work to make each other's separate tasks easier (timeliness), and further enhance the product that we are committed to producing to the best of our ability for our client.

Financial Responsibility (Medium) - We were given a generous budget to help us with any hardware/software that we would need to complete this project. Keeping that in mind, we have put a lot of thought into why we may need a specific component before needing to "purchase" it, and utilizing other resources such as ETG.

Communication Honesty (High) - No matter who is in this situation, honesty is key to a good relationship between companies/employees. We met with our client a couple of times, and gave them our progression status to the best of our abilities. This led us to further success because they were able to see places we may be stuck/excelling in, and able to help guide us to different approaches whether that was to benefit our timeliness, or realign us to design constraints that still needed to be addressed.

Health, Safety, Well-Being (Low) - Granted this project almost completely revolves around software development, there should be no safety concerns pertaining to this project.

Property Ownership (High) - Our project is in a way our project, but at the same time not at all. We all have a strong interest in what we are working on, and it may feel like this project is ours, but in reality, we are a group that is having work for John Deere outsourced to us. Throughout completing this project, we may view one approach to work best, but need to keep in mind our client's constraints as this project is a small piece that needs to fit perfectly in a very large puzzle.

Sustainability (Low) - Our piece of the end product most likely will not affect the environment in any context, but can be viewed as resourceful in a different way. The overall concept of our project is to save memory/resources to help John Deere's live data run more efficiently.

Social Responsibility (Medium) - There is close to a 100% guarantee that anyone operating machinery that utilizes our project will have a clue that it exists. Regardless, that could potentially be a goal for our end result. Most people don't think about algorithms that run websites or video games, they just notice when it runs slow or buffers. We should strive to make our product as best we can so no one notices any problems in the system creating an overall better user experience.

7.3 MOST APPLICABLE PROFESSIONAL RESPONSIBILITY AREA

Property ownership is probably the most important aspect of responsibility to our project. The other six responsibilities could be viewed as a subset of this one. As mentioned above, this isn't really our project, it is John Deere's. We have accepted the obligation to create a product for them, and need to respect them when making decisions. Both work competence, and communication honesty were denoted as high importance responsibilities, in addition to property ownership. Again, these two could be considered a subset since communication honesty pertains primarily to client interaction, and work competence benefits both parties in terms of work quality, and effectiveness.

8 Closing Material

8.1 DISCUSSION

Talking with our client, they wanted a lossless, low-latency, hardware compression and decompression pipeline interfaced with HDMI. So far, we've designed a layout for the hardware IP's that will provide the lowest latency while meeting other requirements like storing a frame(image) in a memory buffer. The compression algorithms we chose (LZW, Cooley–Tukey FFT) are simple enough to be implemented with our small team, near zero-latency, and are decent at compression. LZW can theoretically be pipelined in hardware through the way it stores and accesses data in a dictionary style- designed in hardware properly, LZW can check all dictionary nodes in a table for a match in one clock cycle.

8.2 CONCLUSION

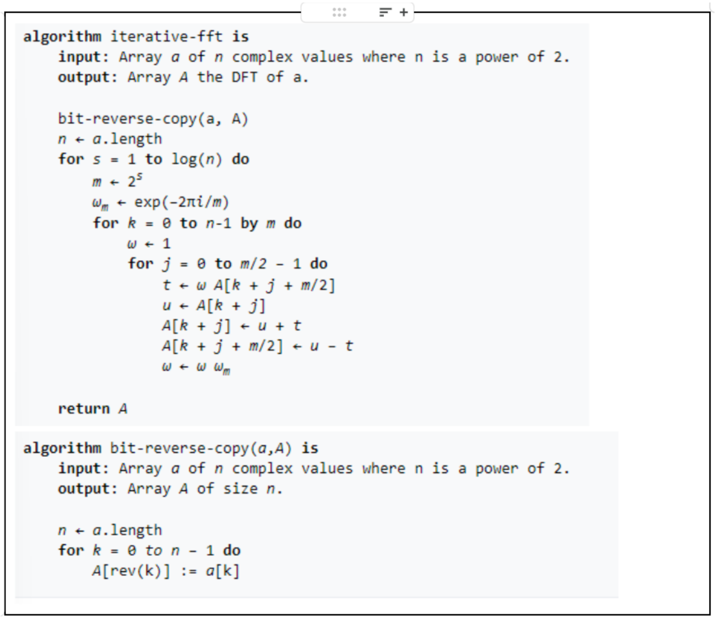
So far we have created the design for the layout of components for the hardware compression pipeline, researched and found compression algorithms that would fit best to be low-latency, simple, and lossless. We have nearly completed a prototype implementation of LZW compression in C for real-world testing. We have also picked out our dev-board and talked with clients to ensure we are keeping to their requirements. We plan on finishing the algorithm implementations, testing them, and then attempting to use Vivado Synthesis to transform the implementation into hardware. If that fails, manual coding in hardware may have to be done. We also plan on getting an HDMI pass-through working on the dev-board. Once both of these have been done, we can take the compression and decompression IP's we made, and add them to the HDMI pass-through. From there we need to do extensive testing, and then implement a way to benchmark memory usage and latency to show off to our clients.

8.3 REFERENCES

Welch, "A Technique for High-Performance Data Compression," in Computer, vol. 17, no. 6, pp. 8-19, June 1984, doi: 10.1109/MC.1984.1659158.

- *LZW compression: https://ieeexplore.ieee.org/document/1659158*

8.4 APPENDICES



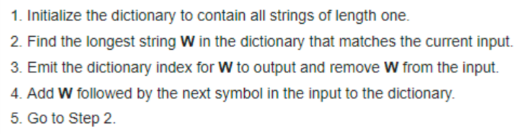
```
algorithm iterative-fft is
    input: Array a of n complex values where n is a power of 2.
    output: Array A the DFT of a.

    bit-reverse-copy(a, A)
    n ← a.length
    for s = 1 to log(n) do
        m ← 2^s
        ω_m ← exp(−2πi/m)
        for k = 0 to n-1 by m do
            ω ← 1
            for j = 0 to m/2 – 1 do
                t ← ω A[k + j + m/2]
                u ← A[k + j]
                A[k + j] ← u + t
                A[k + j + m/2] ← u – t
                ω ← ω ω_m

    return A
```

```
algorithm bit-reverse-copy(a,A) is
    input: Array a of n complex values where n is a power of 2.
    output: Array A of size n.

    n ← a.length
    for k = 0 to n – 1 do
        A[rev(k)] := a[k]
```

Algorithm Name: Lempel–Ziv–Welch (LZW)

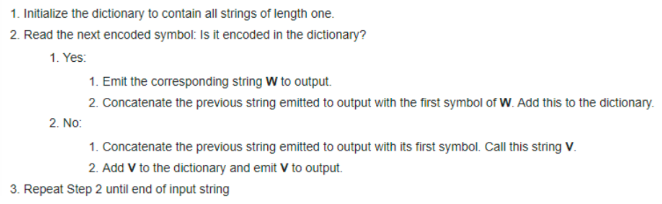
Link/s:https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch

Description:
Lossless.
LZW patent has allegedly expired.
Encoding:



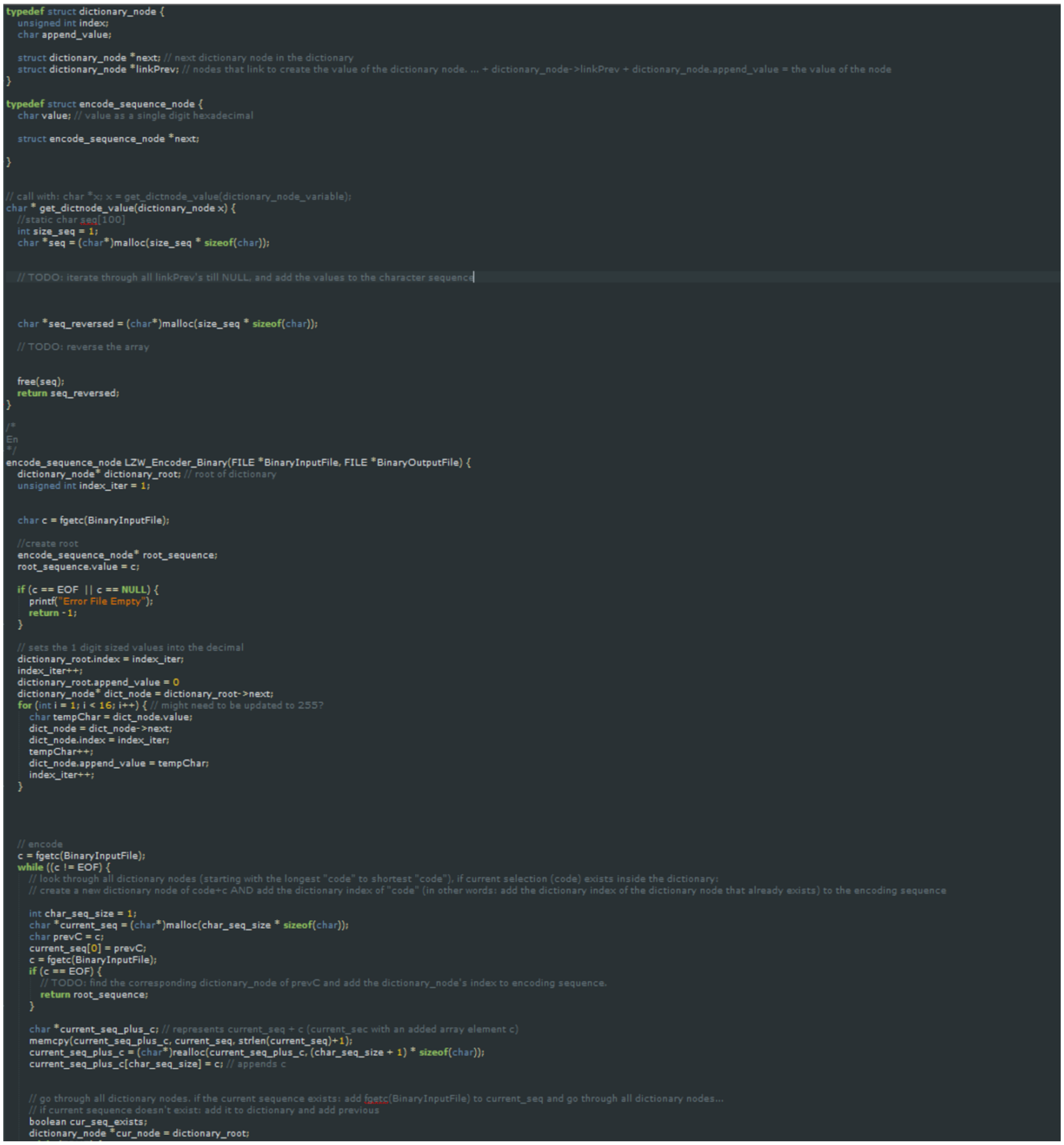
1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string **W** in the dictionary that matches the current input.
3. Emit the dictionary index for **W** to output and remove **W** from the input.
4. Add **W** followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Decoding:

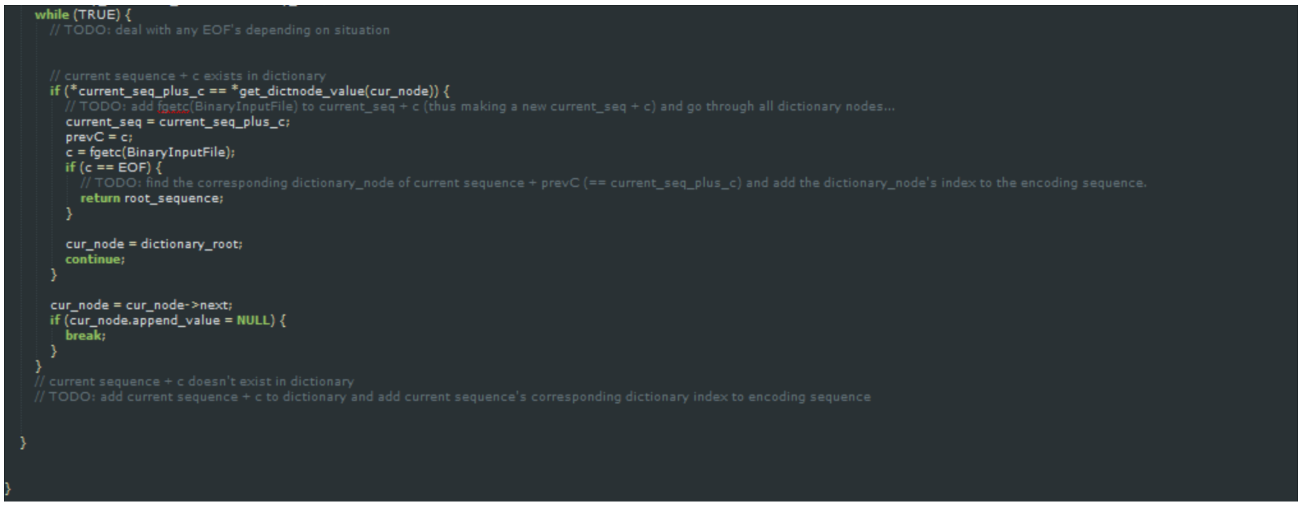


1. Initialize the dictionary to contain all strings of length one.
2. Read the next encoded symbol: Is it encoded in the dictionary?
    1. Yes:
        1. Emit the corresponding string **W** to output.
        2. Concatenate the previous string emitted to output with the first symbol of **W**. Add this to the dictionary.
    2. No:
        1. Concatenate the previous string emitted to output with its first symbol. Call this string **V**.
        2. Add **V** to the dictionary and emit **V** to output.
3. Repeat Step 2 until end of input string

String library:



```c
typedef struct dictionary_node {
   unsigned int index;
   char append_value;

   struct dictionary_node *next; // next dictionary node in the dictionary
   struct dictionary_node *linkPrev; // nodes that link to create the value of the dictionary node. ... + dictionary_node->linkPrev + dictionary_node.append_value = the value of the node
}

typedef struct encode_sequence_node {
   char value; // value as a single digit hexadecimal

   struct encode_sequence_node *next;

}


// call with: char *x; x = get_dictnode_value(dictionary_node_variable);
char * get_dictnode_value(dictionary_node x) {
   //static char seq[100]
   int size_seq = 1;
   char *seq = (char*)malloc(size_seq * sizeof(char));


   // TODO: iterate through all linkPrev's till NULL, and add the values to the character sequence



   char *seq_reversed = (char*)malloc(size_seq * sizeof(char));

   // TODO: reverse the array


   free(seq);
   return seq_reversed;
}

/*
En
*/
encode_sequence_node LZW_Encoder_Binary(FILE *BinaryInputFile, FILE *BinaryOutputFile) {
   dictionary_node* dictionary_root; // root of dictionary
   unsigned int index_iter = 1;


   char c = fgetc(BinaryInputFile);

   //create root
   encode_sequence_node* root_sequence;
   root_sequence.value = c;

   if (c == EOF  || c == NULL) {
      printf("Error File Empty");
      return -1;
   }

   // sets the 1 digit sized values into the decimal
   dictionary_root.index = index_iter;
   index_iter++;
   dictionary_root.append_value = 0
   dictionary_node* dict_node = dictionary_root->next;
   for (int i = 1; i < 16; i++) { // might need to be updated to 255?
      char tempChar = dict_node.value;
      dict_node = dict_node->next;
      dict_node.index = index_iter;
      tempChar++;
      dict_node.append_value = tempChar;
      index_iter++;
   }




   // encode
   c = fgetc(BinaryInputFile);
   while ((c != EOF) {
      // look through all dictionary nodes (starting with the longest "code" to shortest "code"), if current selection (code) exists inside the dictionary:
      // create a new dictionary node of code+c AND add the dictionary index of "code" (in other words: add the dictionary index of the dictionary node that already exists) to the encoding sequence

      int char_seq_size = 1;
      char *current_seq = (char*)malloc(char_seq_size * sizeof(char));
      char prevC = c;
      current_seq[0] = prevC;
      c = fgetc(BinaryInputFile);
      if (c == EOF) {
         // TODO: find the corresponding dictionary_node of prevC and add the dictionary_node's index to encoding sequence.
         return root_sequence;
      }

      char *current_seq_plus_c; // represents current_seq + c (current_sec with an added array element c)
      memcpy(current_seq_plus_c, current_seq, strlen(current_seq)+1);
      current_seq_plus_c = (char*)realloc(current_seq_plus_c, (char_seq_size + 1) * sizeof(char));
      current_seq_plus_c[char_seq_size] = c; // appends c


      // go through all dictionary nodes. if the current sequence exists: add fgetc(BinaryInputFile) to current_seq and go through all dictionary nodes...
      // if current sequence doesn't exist: add it to dictionary and add previous
      boolean cur_seq_exists;
      dictionary_node *cur_node = dictionary_root;
```



```
    while (TRUE) {
      // TODO: deal with any EOF's depending on situation


      // current sequence + c exists in dictionary
      if (*current_seq_plus_c == *get_dictnode_value(cur_node)) {
        // TODO: add fgetc(BinaryInputFile) to current_seq + c (thus making a new current_seq + c) and go through all dictionary nodes...
        current_seq = current_seq_plus_c;
        prevC = c;
        c = fgetc(BinaryInputFile);
        if (c == EOF) {
          // TODO: find the corresponding dictionary_node of current sequence + prevC (== current_seq_plus_c) and add the dictionary_node's index to the encoding sequence.
          return root_sequence;
        }

        cur_node = dictionary_root;
        continue;
      }

      cur_node = cur_node->next;
      if (cur_node.append_value = NULL) {
        break;
      }
    }
    // current sequence + c doesn't exist in dictionary
    // TODO: add current sequence + c to dictionary and add current sequence's corresponding dictionary index to encoding sequence


  }


}
```

Algorithm Name: Cooley–Tukey FFT algorithm

Link/s: https://en.wikipedia.org/wiki/Fast_Fourier_transform
https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm

Description:

Let $x_0, \ldots, x_{n-1}$ be complex numbers. The DFT is defined by the formula

$$X_k = \sum_{m=0}^{n-1} x_m e^{-i2\pi km/n} \qquad k = 0, \ldots, n-1,$$

where $e^{i2\pi/n}$ is a primitive n'th root of 1.

Evaluating directly requires O(n^2)
Could be O(nlogn) in best case.

Use a Divide and Conquer algorithm to decrease runtime.

```
X_{0,...,N-1} ← ditfft2(x, N, s):             DFT of (x_0, x_s, x_{2s}, ..., x_{(N-1)s}):
    if N = 1 then
        X_0 ← x_0                                      trivial size-1 DFT base case
    else
        X_{0,...,N/2-1} ← ditfft2(x, N/2, 2s)          DFT of (x_0, x_{2s}, x_{4s}, ..., x_{(N-2)s})
        X_{N/2,...,N-1} ← ditfft2(x+s, N/2, 2s)        DFT of (x_s, x_{s+2s}, x_{s+4s}, ..., x_{(N-1)s})
        for k = 0 to N/2-1 do                          combine DFTs of two halves into full DFT:
            p ← X_k
            q ← exp(-2πi/N k) X_{k+N/2}
            X_k ← p + q
            X_{k+N/2} ← p - q
        end for
    end if
```

8.4.1 Team Contract

Team Name sdmay24-12. Pipelined Hardware Video Compressor & Decompressor

Team Members:

1) _Colsen Selk____________________ 2) _Caleb Rock_____________________
3) _Ben Meinders___________________ 4) _Logan McDermott_______________
5) _Kareem Eljaam_________________

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings: **Wednesdays 2-3 Face-To-Face Team Meetings**
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face): **F2F, Discord**
3. Decision-making policy (e.g., consensus, majority vote): **Consensus**
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived): **Logan McDermott: Shared Google Doc**

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings: **Everyone should always attend meetings punctually. If a conflict occurs, let teammates know beforehand.**
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines: **All deadlines should be met unless issues hinder progress. If issues arise, team members should be notified as soon as possible.**
3. Expected level of communication with other team members: **Should be responsive on Discord within the day unless conflicts arise.**
4. Expected level of commitment to team decisions and tasks: **Team decisions should be upheld unless problems arise in the implementation of the decision. If problems arise, a solution should be decided upon by the team.**

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.): **All members are expected to communicate with each other concerning what new features they have added and are expected to collaborate where the different "sections" of work interact.**
    a. **Kareem Eljaam: Client Interaction**
    b. **Caleb Rock: Faculty Advisor Interaction**
    c. **Colsen Selk: Leading Testing of Software.**
    d. **Logan McDermott: Leading Testing of Hardware**
    e. **Ben Meinders: Leading Component Design of Software**
2. Strategies for supporting and guiding the work of all team members: **Teammates should communicate often about their work so that team members understand enough about each other's work to give feedback to others.**
3. Strategies for recognizing the contributions of all team members: **Team members should share their contributions weekly at the team meetings.**

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    a. **Kareem Eljaam: CPRE, worked with John Deere on FPGA applications.**
    b. **Caleb Rock: EE, understands hardware and software interaction.**
    c. **Colsen Selk: Software Engineer, with good knowledge of algorithms and coding.**
    d. **Logan McDermott: CPRE, knowledge of VHDL and FPGA simulation tools,**
    e. **Ben Meinders: SE, worked with CNH Industrial on automated data transfer**
2. Strategies for encouraging and supporting contributions and ideas from all team members:
   Provide time at weekly meetings for people to give updates on how they are doing with the project. Ensure all members are getting and completing tasks.
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
   Provide time at weekly meetings for feedback about the team environment and time to brainstorm ways to create a better team environment.

Goal-Setting, Planning, and Execution

1. Team goals for this semester: **design a comprehensive plan and design for the project such that implementation in the next semester is as easy as possible.**
2. Strategies for planning and assigning individual and teamwork: **planning and assigning individual work will be done**
3. Strategies for keeping on task: **before each meeting we will record a general plan of the topics we want to cover so we can focus on our topics during the meeting.**

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
   Logan will record any infractions of the team contract. We will also let the individual know about the infraction(s).
2. What will your team do if the infractions continue?
   Contact TA/Professor about the infractions

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I must abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Logan McDermott____________________________________ DATE 9/8/2023_____________
2) Ben Meinders____ DATE 9/8/2023_____________
3) Colsen Selk____ DATE 9/8/2023 ___________
4) Kareem Eljaam___________________________ ____________ DATE 9/8/2023____________
5) Caleb Rock____ DATE 9/8/2023____________