Shivam Vyas 400076462
Karthik Kaushik 400050868

# FINAL PROJECT

COMP ENG 3DQ5

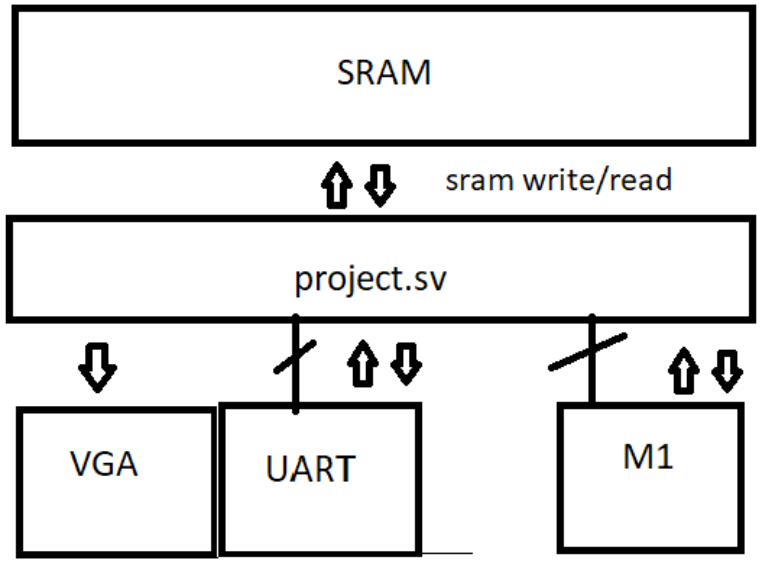
NOVEMBER 30, 2020

DR. NICOLA NICOLICI
Shivam Vyas 400076462 - Karthik Kaushik 400050868

## Introduction

This project is a hardware implemented image decompressor. It can be separated into 3 main milestones. Firstly, decompression and dequantization (Milestone 3). Secondly, inverse discrete cosine transform. Lastly, upsampling and colourspace conversion. Our report will be focusing on Milestone 1 as we were able to successfully test and deliver the requirements. We will be going over design structure and implementation details.

## Design Structure




On the left you can see a non-strict version of the project structure that we have implemented in this project.

Milestone 1 required us to implement an upscaler and a colourspace conversion scheme. In project.sv file, the top-level FSM was modified, and combinational logic was added to move from top-level state to milestone 1 state.

To implement the changed mentioned above, the M1_start and M1_finish signals communicate with the milestone 1 module so that it can work with the top-level project.sv file. This enable signal starts the milestone 1 process and when the finish signal is received the M1 output is done so that the FSM can go back to the top-level state.

## Implementation Details

To implement the color space conversion and the upscaling stages we used .mic14 standard. Our milestone 1 takes data from post-IDCT data output then interpolates values to attain the full range of U and V samples. To implement colourspace conversion and interpolation we first attained the outputted U and V values as required by the project specifications. Then we performed interpolation on this data using the given requirements in the project specification document. We computed partials products in an accumulator register to meet the efficiency requirements and limitations of having 4 multipliers. To calculate the colourspace conversions we used the formula given in the project guidelines. To reduce the use of multipliers we made sure to store the reused values in buffers so that we do not need to instantiate the values every time. Finally, to calculate the RGB values we used the matrix multiplication formula given in the specification document.

$$U'[j] = \begin{cases} U\left[\frac{j}{2}\right] & j\ even \\ (int)\frac{1}{256}\left(21U\left[\frac{j-5}{2}\right] - 52U\left[\frac{j-3}{2}\right] + 159U\left[\frac{j-1}{2}\right] + 159U\left[\frac{j+1}{2}\right] - 52U\left[\frac{j+3}{2}\right] + 21U\left[\frac{j+5}{2}\right] + 128\right) & j\ odd \end{cases}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = (int)\frac{1}{65536}\left(\begin{bmatrix} 76284 & 0 & 104595 \\ 76284 & -25624 & -53281 \\ 76284 & 132251 & 0 \end{bmatrix}\left(\begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}\right)\right)$$

The memory layout of the RGB Segment:

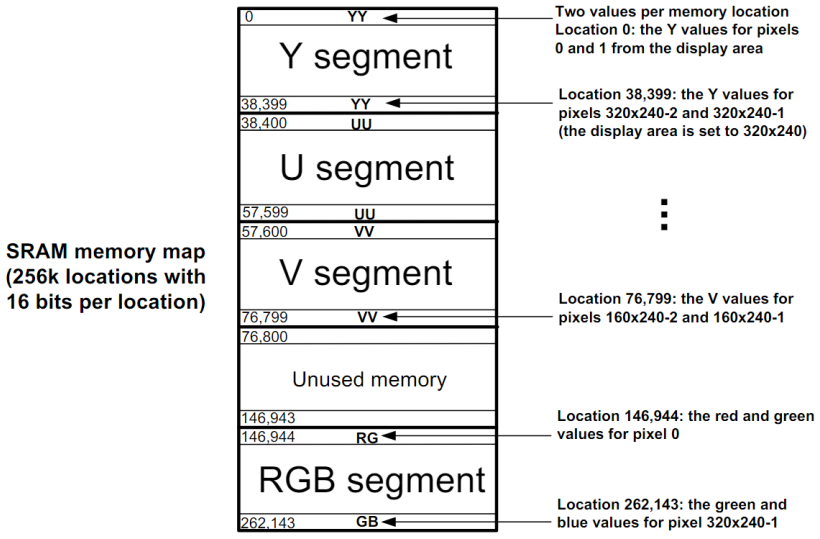



**Figure 17. Memory layout for milestone 1.**

We tried to follow an open approach to our coding style. We were not worried about keeping the code length small as we focused on the readability aspect of our code implementation. The most optimized design we could think of was implemented.

We had a small lead-in of 10 clock cycles to set up the common case. The common case ran for 6 clock cycles and after each row of the display was complete there was a 6 clock cycles lead-out. We tried to match the transitions of our states so that we could handle the write from our current clock's input as well as our previous iteration's output. Thus every common case produces sets of RGB values and stores them into the SRAM as specified by the project document. We have flags set in place for lead-in/lead-out/common_case_transitions to ensure that the correct values are being used as operands for our calculations. The formula to calculate runtime for Milestone 1 is as follows:

((320 (Number of pixels in each row) x 6 (common case))/2 (per value) + 10 (lead-in) + 6 (lead-out)) x 240 (number of columns) = 234240 (total clock cycles)

234240 (total clock cycles) x 20ns (duration of each clock) = 4684800 ns

This value matches our testbench simulation results as well. We required a multiplier utilization of at least 85% minimum usage. To check for this we can use the following equation:

(6+6+5+5)/4(per multiplier) = 5.5 (average usage of each multiplier)

(5.5 x 320 x 240) (total pixels) /2 (per value) = 211200 (total clock cycles of multipliers used)

211200 / 234240 (total clock cycles) ≈ 0.9016 = 90.16 % Utilization

This shows our design has a multiplier utilization of 90% which matches the requirements for the project.

Our state table was a rough idea of what our entire design should have been. Due to this fact, we ran into a lot of coding and timing issues. A lot of the coding additions we made were on the fly which resulted us to encounter issues like mismatched variables and incorrect state progression. Most of these issues were solved by using "brute-force" approach and also trial and error. We also had an issue with our multipliers receiving the operands a clock cycle late. To solve this we decided to create a separate an always comb block to control these values. Lasty, we had issues at the start when renaming modules and having them in separate folders. We circumvented this issue by doing most of the work in the RTL folder and later moving the testbench, simulation, and data files to their respective subfolders.

Note: All of the work for the project was **only** done during meetings with both partners involved at all stages.

| **Timeline** | **Description** |
|---|---|
| Week 1 | Project outline and initial work plan created. Rough state table created and coding of milestone 1 begins. |
| Week 2 | Milestone 1 coding and debugging continues. Files moved from RTL folder to correct sub folders. Testbench runs successfully. Meeting for milestone 2. |
| Week 3 | No progress made |
| Week 4 | No progress made |
| Week 5 | Report finalized. |

## Conclusion

To summarize our learning experience, we learned to focus on time management and hardware description is something that does not come easy to us. Due to the online nature of this year's courses, all coursework increased exponentially. This resulted in multitude of meetings for each course that caused a lot of conflicts to arise. Due to this fact, our group could not complete all requirements of the project as each component required several days to brainstorm, debug, and design solutions. We learned some useful strategies in systems design through the progress of this course. This includes combinational logic, usage of state machines, and broader understanding of image compression and processing.

## References

Comp Eng. 3DQ5 Lab manual 4

Comp Eng. 3DQ5 Lab manual 5

Comp Eng. 3DQ5 Lab instructions and recordings

Comp Eng. 3DQ5 Project instructions and recordings

Comp Eng. 3DQ5 Project spec document