COE3DQ5 – Project Report

Huner Dogra & Sarbjot Ghotra

Group-68

November 27, 2023

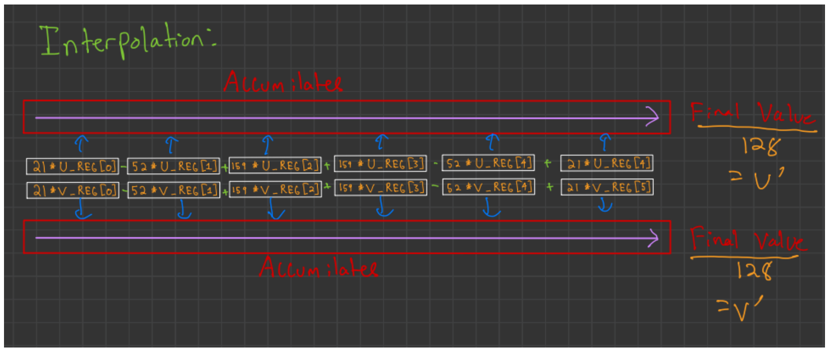
Introduction

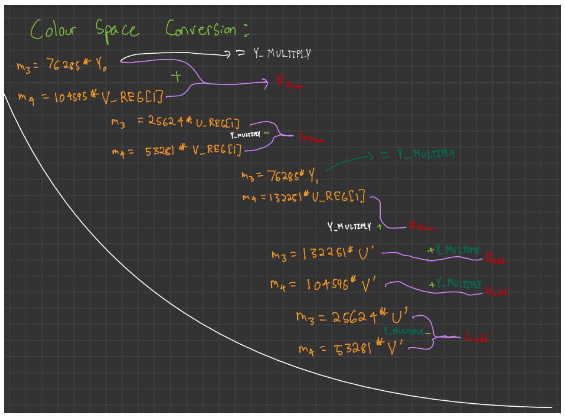
The main objective of this project is to decompress a 320 x 240 pixel image using a custom digital circuit. This is done by using a UART interface to have our PC communicate with the Altera DE2 board, using an SRAM to store the decompressed image, and using a VGA controller to read it and display it to a monitor. The various milestones of the project complete this process using lossless decoding, dequantization, signal transformation, interpolation and colour space conversion. Signal transformation (milestone 2) uses IDCT (Inverse Discrete Cosine Transform) to recover the downsampled image after dequantization. This uses matrix multiplications with multiple DPRAMs to store and compute S', T, and S values concurrently. The S values are stored as Y, U, and V values in the SRAM. These YUV Values are then used for interpolation and colour space conversion (milestone 1). In this stage we use interpolation to generate the odd columns of the image, as the even columns are taken from the downsampled data. After this upsampling process, 3 equations are used to calculate the red green and blue values for each pixel. As specified in the requirements of the project, each step of the image decompression process uses its multipliers efficiently and accounts for clock cycle delays in all processes.

Implementation Details

Upsampling and Colour Space Conversion (Milestone 1)

Interpolation and Calculation Figure





The above figures show a brief rundown of how the calculations for interpolation and colour space conversion are organized. For interpolation the values are accumulated throughout the clock cycles, then divided by 128 to get a final value. For colour space conversion, each value is calculated using the previous multipliers, and in some cases the Y_MULTIPLY value (stored from previous calculations) is used as well.

Register and Purposes

Module	Register Name	Bits	Description
M1_interface	R_VALUE, G_VALUE, B_VALUE	16	Stores final even and odd values of the colour for the pixel
M1_interface	R_CALC_E, R_CALC_O, G_CALC_E, G_CALC_O, B_CALC_E, B_CALC_O	32	Stores the calculated value for the even and odd colours separately, and saves it to be clipped later (and stored in the R, G, or B_VALUE register)

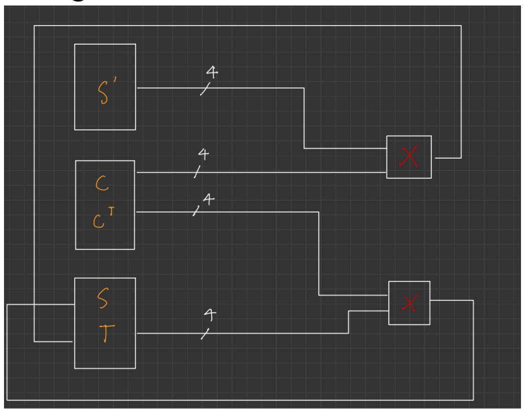
M1_interface	U_REG, V_REG	6 x 16	Holds 6 even values of U, and used to interpolate the odd values (U')
M1_interface	U_PRIME_CALC, V_PRIME_CALC	32	Accumulates the calculations for U' and V'
M1_interface	U_PRIME, V_PRIME	8	Stores the final calculated V' value
M1_interface	Y_STORE, U_STORE, V_STORE	16	Buffer to store Y, U, or V value that is read in from the SRAM (U and V are used to shift their respective registers in the next cycle)
M1_interface	op11, op12, op21, op22, op31, op32, op41, op42, m1, m2, m3, m4	32	Multipliers and operator registers to compute any multiplications necessary
M1_interface	Y_MULTIPLY	32	Stores the constant calculation of Y*76285 (reads Y value from SRAM)
M1_interface	COL_NUM	9	Keeps track of which column the pixel calculations are being done for (important for keeping track of when to move to lead out)
M1_interface	flag	1	Differentiates between the two different common case cycles (reading Y,U,V vs. reading Y only)

Latency Analysis

Based on our waveforms simulation, there are 235682 total clock cycles. This means that it takes 235682 clock cycles to completely upsample and perform colourspace conversion for every pixel of the given image. We know that we use the same number of multipliers every row of the image. Using this knowledge we know that the utilization of the multipliers for the entire milestone will be the same as that of a single row of the image. Based on our code our lead in case runs for 24 states in total, while using 8.5 of the total multipliers. The lead out runs for 28 states, and uses 19 of the total multipliers. By viewing the waveform on ModelSim we can see that our common case runs for 929 states per row, and it uses 5.5 multipliers for every 6 states. Based on this we can compute that it uses 851.58 multipliers for the common case states. By adding the multiplier usage and dividing it by the total states: $\frac{851.58+19+8.5}{981} = 0.896 = 89.6\%$ usage. This is much more than the 75% requirement from the project specification.

Inverse Discrete Cosine Transform (Milestone 2)

Embedded RAM Matrices Figure



In the figure shown above the DRAM stores the S', C, C^T, S, and T values. 4 values are extracted at once and used in the appropriate multipliers, then fed back into the DRAM to be used in future calculations (or to be stored in SRAM).

Register and Purpose

Module	Register Name	Bits	Description
M2_interface	op11, op12, op21, op22, op31, op32, op41, op42, m1, m2, m3, m4	32	Multipliers and operator registers to compute any multiplications necessary
M2_interface	A1_ADDR_COUNTER, B1_ADDR_COUNTER, A2_ADDR_COUNTER, B2_ADDR_COUNTER, A2_ADDR_COUNTER_CT, B2_ADDR_COUNTER_CT	6	DPRAM1, and DPRAM2 read counters, as well as the counters for reading C transpose
M2_interface	T_W_ADDR, T_R_ADDR_A, T_R_ADDR_B	6	T read and write addresses for DPRAM3
M2_interface	S_RC	6	S read counter from DPRAM3
M2_interface	Y_BLOCK, U_BLOCK, V_BLOCK	18	Holds the addresses for the start of each type of block
M2_interface	address_a, address_b	3 x 7	Holds addresses for DPRAMs
M2_interface	write_data_b, write_enable_b, read_data_a, read_data_b	3 x 32	Variables to initialize when to read/write data and to define what data to read/write
M2_interface	S_PRIME_WC	18	Write counter for S'
M2_interface	S_PRIME_BUF	16	Buffer for S'
M2_interface	S_PRIME_COMPUTE_BUF	8 x 16	Buffer for the computation of S'
M2_interface	T_COMPUTE_BUF	8 x 32	Buffer for the computation of T
M2_interface	T_CALC	32	Accumulator for calculating T
M2_interface	S_CALC	40	Accumulator for calculating S
M2_interface	flag, flag2	1	Flags for when initial write should be skipped
M2_interface	ROW_COUNTER, ROW_COUNTER2	4	Keeps track of current common case iterations

Latency Analysis

Unfortunately, due to the fact this milestone was not completed we could not check how many clock cycles the entire milestone would take. However, based on the usage of our multipliers in each of our planned states we can do the following calculation (we neglect the first 13 lead ins due to the fact that they do not repeat and therefore hardly affect the overall multiplier usage):
$128\ Common\ Case\ with\ 100\%\ usage + 4\ Stretch\ with\ 0\%\ usage + 128\ common\ case\ with\ 100\%\ usage = \frac{256}{260} = 0.9846$
Therefore, we can estimate that our multiplier usage would be 98.46%. This meets the 95% usage requirement.

Resource Usage and Critical Path

In our project, we use a total of 1856 logic elements according to Quartus. This is larger than the 616 logic elements from the starting point of the project (lab 5 experiment 4). This is in line with our expectations, as the number of logic elements required for this project is much more than in lab 5. However, after taking another look at our code we have identified parts of the design that could be improved. In our design we chose to use one register for each even RGB pixel and one register for each odd even RGB pixel, for a total of 6 registers to store RGB values. If we were to revise this design in the future we could use 3 registers to store both even and odd values, and simply allocate half the register for even and the other half for odd values. We also chose to make our U_REG and V_REG 6 x 16 bits, but we have now realized we could have reduced this to 6 x 8 bits. The critical path for our project is when going from node op31[3] to G_CALC_E[30] in the M1 module, This has a data delay of 14.253 ns, and with a slack of 5.629. According to the Fmax summary in Quartus this gives us a maximum frequency of 69.58 MHz, which is much more than the required 50 MHz. This critical path makes sense for our design, and can be explained by the fact that the op31 is the first multiplication that is used to calculate the even G value (so it must be the starting node for the critical path). The even G value also requires a computation involving three values, which means that the time taken for the system to receive the data will be longer than for the other colours (which only involve two values). This is further described in the following steps: After computing the op31*op32 multiplication (for m3), the op41*op42 multiplication (for m4) must also be computed. Then, G_CALC_E uses these values, as well as the stored Y_MULTIPLY value to compute the final value of the even G value. Considering this explanation, it is expected that going from node op31 to G_CALC_E would be the critical path.

Weekly Activity and Progress

Week	Progress	Group Member Contribution
1	- Read the project specifications to gain an understanding of the project and its requirements.	*Huner and Sarbjot*
2	- Started the state table for milestone 1.	*Huner and Sarbjot*
3	- Continued perfecting the state table for milestone 1 to ensure minimal issues when coding.	*Huner and Sarbjot*
4	- Began coding milestone 1 and set up project.sv file. - Finished coding and began debugging.	*Sarbjot:* Began coding milestone 1 *Huner:* Finished coding milestone 1 *Huner and Sarbjot:* Began debugging milestone 1
5	- Continued debugging milestone 1. - Finished milestone 1. - Started working on milestone 2 state table. - Finished milestone 2 state table and began coding.	*Huner and Sarbjot:* Continued debugging milestone 1 *Huner and Sarbjot:* Starting working on milestone 2 state table. Finished milestone 2 state table *Sarbjot:* Began coding milestone 2

Conclusion

This project has provided us with a very thorough learning experience. It gave us the knowledge necessary to complete a project of this type in the future. We learned the importance of time management as well as how to implement computer engineering concepts in a practical way. We learned about how to read and write from both SRAMs and DRAMs (and the clock cycle delays associated with them), how to implement UART communication, and how to effectively debug code that we created. This project also emphasized the importance of communication, as we needed to make sure that both of us understood how we planned to approach each milestone. We needed to work together to solve everything in an effective way, as well as take responsibility for any individual work that was necessary. Unfortunately we were not able to complete all the milestones. The last completed milestone was milestone 1 and the commit made was called "MILESTONE 1 FINISHED - November 24, 2023". As stated in the commit message, this was done on November 24, 2023.

References

Jason Thong, Adam Kinsman, and Nicola Nicolici, "COE3DQ5 Project Description 2023
Hardware Implementation of an Image Decompressor," McMaster University, Accessed: Nov. 3, 2023. [Online]. Available: https://avenue.cllmcmaster.ca/d2l/le/content/554050/viewContent/4381835/View