COE3DQ5 – Project Report

Jibin Mathew, Ronit Ahuja, Biswajit Saha | Group 70 – Thursday | November 27th, 2023

1) Introduction:

The COE 3DQ5 project focuses on implementing and integrating an image decompressing design in hardware while specifically considering Lossless Decoding, Dequantization, Inverse Signal Transform, Interpolation, and Color-space conversion in three defined Milestones. The project is completed with specific constraints and maximized efficiency of hardware components. The final product is aimed to be a full integration of a compressed image being lossless decoded and restoring the missing details of the quantized image, with matrix multiplication for the IDCT, allowing for a down sampled YUV image representation. And through upscaling and colouring conversion in Milestone 1, the converted and upscaled RGB image values are sent into the SRAM for image viewing. All three milestone implementations considered the multiplier usage requirements as discussed in the implementation details. We can take this implementation and look at the bigger picture of the Digital Systems Design course, where the created Verilog design specification is driven and monitored by the testbench for testing design precision and errors, CAD tools are utilized to simulate the circuit behavior, timing analysis, and circuit mapping, all to be configured and loaded to the FPGA for project application.

2) Implementation Details

2.1) Milestone 1:

The first Milestone consists of reading from the external SRAM, utilizing 4 multipliers with a minimum of 75% utilization to perform operations for U and V even and odd calculations and RGB matrix calculations. The milestone takes the input of YUV data and is focused on converting data from YUV to RGB in the order of R0G0, B0R1, G1B1, and repeated.

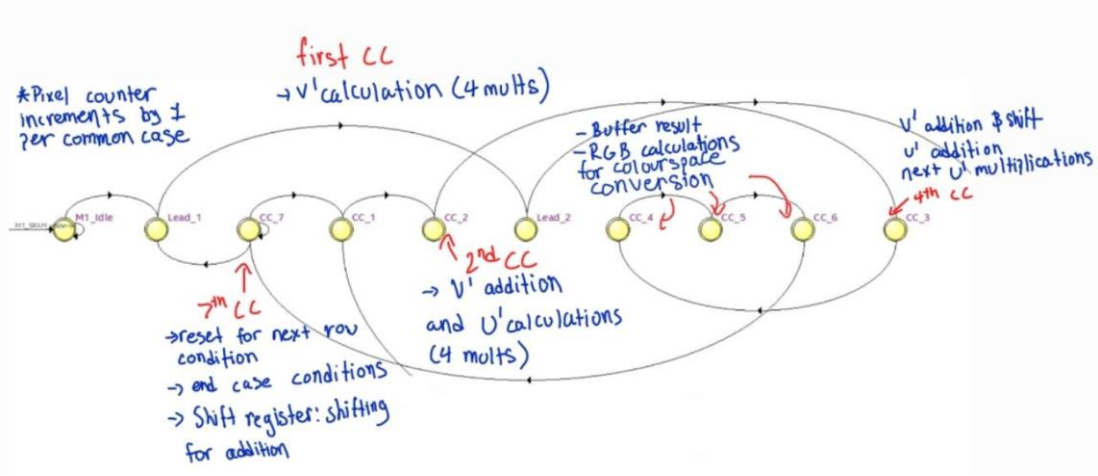

Figure 1: FSM with Common Cases Colour Space Conversion and Interpolation

Algorithm: The general approach for Milestone 1 focused on first requesting the SRAM addresses for U, V, and Y values, which would read two values at a time from the SRAM (e.g. U0 and U1 in lead_4), these values were stored in a register to be utilized in the consequent common case cycle for calculations with the 4 multipliers. Within the common case the required calculations for the even and odd U and V values, and the RGB matrix calculations with the multipliers were completed. While the calculations were completed during the common case cycles, a similar pattern for reading from the SRAM was followed and the acquired YUV data was stored in registers to be used to immediately begin calculations in the next common case cycle.
Efficiency, Timing and Learning: To create an efficient design, YUV values were read in the cycle previous to when they are utilized and preloaded into registers. This method allowed for a reduction in total clock cycles from previous design iterations, all while considering minimal register usage. Furthermore, utilization of the multipliers in the common case can be calculated as: ((5cc * 4mult) + (1cc * 2mult))/(7*4) = 78.57%. Since the lead in does not repeat, the utilization represented as (8 lead in + (number of common case repeats) *((5cc * 4mult) + (1cc * 2mult))/(7*4)). As seen in figure 3 and 4, the number of registers increases from 369 to 938 registers. Latency can be represented as: (7x 160)) +8 = 1128 x 240 = 260720CC.

Milestone 1 provided a deeper understanding of project setup, debugging experience, and algorithm development. The algorithm development and project setup supplied an in-depth experience of what goes into creating and starting a project. The debugging experience provided experience with indexing, clipping, sign extending and more.

Module	Register	Size	Description
M1	stop_M1	1 bit	Flag for M1 ending
M1	SRAM_write_data	16 bit	SRAM getting RGB values written into
M1	SRAM_write_en	1 bit	SRAM active low activation
M1	SRAM_address_M1	18 bit	Current address of SRAM
M1	Data_counter, Y_counter, RGB_counter, Pixel_counter	18 bit	Counters for data incrementation, Y, RGB calculation, Pixel incrementation,
M1	Y_even, Y_odd, U_e, V_e	8 bit	YUV values, even and odd.
M1	U, V	6x8 bit	U and V values for U and V prime calculations
M1	U_prime, V_prime	32 bit	Calculated U and V prime per equation
M1	U_buff, V_buff	16 bit	U and V buffer for addition
M1	CC tracker, End_case, End_case_y, Last_write	1 bit	Flags for common case, final write, edge cases
M1	Mult_op_x (1 to 8), Mult_result_x (1 to 4)	32 bit	Values being multiplied and the results
M1	Mult_result_long_x (1 to 4)	64 bit	Stores result as 64 bit
M1	Mult_result_x_preclip (1 to 6) Mult_result_x_clip (1 to 6)	32 bit	Sum of mult_result to be clipped, clipping for values out of range
M1	CC_x_Mul_y x: (5 to 7) and y: (1 to 2)	32 bit	U and V shifting

```
assign Mult_result_1_clip = (Mult_result_1_preclip[31] == 1'b1) ? 32'd0 : (|Mult_result_1_preclip[30:24] == 1'b1) ? 32'hFFFFFFFF : Mult_result_1_preclip;
assign Mult_result_2_clip = (Mult_result_2_preclip[31] == 1'b1) ? 32'd0 : (|Mult_result_2_preclip[30:24] == 1'b1) ? 32'hFFFFFFFF : Mult_result_2_preclip;
assign Mult_result_3_clip = (Mult_result_3_preclip[31] == 1'b1) ? 32'd0 : (|Mult_result_3_preclip[30:24] == 1'b1) ? 32'hFFFFFFFF : Mult_result_3_preclip;
assign Mult_result_4_clip = (Mult_result_4_preclip[31] == 1'b1) ? 32'd0 : (|Mult_result_4_preclip[30:24] == 1'b1) ? 32'hFFFFFFFF : Mult_result_4_preclip;
assign Mult_result_5_clip = (Mult_result_5_preclip[31] == 1'b1) ? 32'd0 : (|Mult_result_5_preclip[30:24] == 1'b1) ? 32'hFFFFFFFF : Mult_result_5_preclip;
assign Mult_result_6_clip = (Mult_result_6_preclip[31] == 1'b1) ? 32'd0 : (|Mult_result_6_preclip[30:24] == 1'b1) ? 32'hFFFFFFFF : Mult_result_6_preclip;


logic [31:0] CC_5_Mul_1, CC_5_Mul_2, CC_6_Mul_1, CC_6_Mul_2, CC_7_Mul_1, CC_7_Mul_2;

assign CC_5_Mul_1 = (Mult_result_1_clip >>> 16);
assign CC_5_Mul_2 = (Mult_result_2_clip >>> 16);

assign CC_6_Mul_1 = (Mult_result_3_clip >>> 16);
assign CC_6_Mul_2 = (Mult_result_4_clip >>> 16);

assign CC_7_Mul_1 = (Mult_result_5_clip >>> 16);
assign CC_7_Mul_2 = (Mult_result_6_clip >>> 16);
```

Figure 2: Clipping and Multiplier result shifting.

Flow Status	Successful - Mon Nov 27 14:52:11 2023
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	experiment4
Top-level Entity Name	experiment4
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	616 / 114,480 (< 1 %)
Total registers	369
Total pins	160 / 529 (30 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	1 / 4 (25 %)

Flow Status	Successful - Mon Nov 27 20:47:47 2023
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	project
Top-level Entity Name	project
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	2,182 / 114,480 (2 %)
Total registers	938
Total pins	160 / 529 (30 %)
Total virtual pins	0
Total memory bits	9,216 / 3,981,312 (< 1 %)
Embedded Multiplier 9-bit elements	24 / 532 (5 %)
Total PLLs	1 / 4 (25 %)

Figure 3 and 4: Starting and Ending Register Count

2.2) Milestone 2:

Milestone 2 focuses on the IDCT process through 4 steps: fetching S', computing T, computing S, and writing S to the SRAM. The S' values are fetched from the SRAM and stored in the DP0 as a lead in for 64 clock cycles. The computation of T utilizes the C and S' values and is stored in DP1 for the S' * C = T matrices. The computation of S is like computation of T, $S = C^T * T$. For T, the pattern using the 2 S' values per common case cycle, provides 4 results in 8 cycles; as does for S. The S calculation provides 4 results in 8 cycles with one T value used per common case cycle. After the S calculations are complete, the YUV values are concatenated and written to the SRAM; to be the input to Milestone 1. This design was accomplished with 64 lead ins, and 16 common cases. Additionally, the equation for IDCT requires a shift by 8 and 16 for T and S, respectively. Latency can be represented as: (64 x 160)) + 64 = 10304 x 240 = 2472960CC.

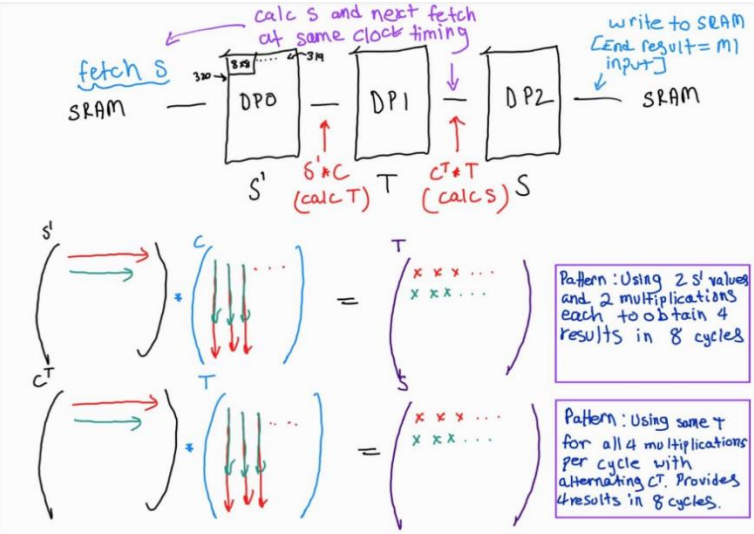

Figure 5: Layout of Design for SRAM, DP Memory, and Matrix Multiplication

Approach and Timing Analysis:
Moreover, our initial Milestone 2 implementation was replaced by a more efficient and component effective method after discussion with the Professor. The initial implementation consisted of excessive register usage and underutilized the provided DP memory. A newer design which had a slightly longer lead in and more consistent and memory effective design was implemented as a method of improved design and learning. The efficiency and timing is: $(16^3)/(16^3 + 64)$ = 99.6% utilization. This calculation aligns with the expectations as there is an extremely large sum of clock cycles, all but approximately 64 lead ins consisting of 4 multiplications.

M2	SRAM_address_M2, SRAM_write_data	18, 16 bit	SRAM address and data for writing
M2	SRAM_write_en, stop_M2, S_calculations	1 bit	SRAM write enable, stop signal, flag for doing S calculations
M2	M2_data_counter, M2_SRAM_line_counter	18 bit	Counters for data and line addressing
M2	M2_SRAM_line_adder, M2_block_counter	18 bit	Address adder and block counter
M2	dp_0_address_counter, dp_0_read_counter	8 bit	Counters for dual-port 0 memory access
M2	dp_1_address_counter, dp_1_read_counter	8 bit	Counters for dual-port 1 memory access
M2	dp_2_address_counter, dp_2_read_counter	8 bit	Counters for dual-port 2 memory access
M2	Total_rows, CC_counter, row_counter	9, 1, and 7 bit	Keeps track of total rows, T calculation values, and row tracking
M2	Lead_out, S_Fetch, S_prime_write	1 bit	Control signals for data flow
M2	SRAM_write_lead, SRAM_write	1 bit	Control signals for SRAM writing
M2	SRAM_address_increment	1 bit	Address increment
M2	Write_row_tracker	8 bit	Row tracking
M2	Buff	128 bit	Buffer array for intermediate data storage
M2	C_array	16 x 64 bit	Array to store C coefficients
M2	Summation	128 bit	Array to store sum values
M2	address_0_A, address_0_B, address_1_A, address_1_B, address_2_A, address_2_B	7 bit	Address registers for DP-RAM 0, 1, and 2 for both ports
M2	data_a, data_b	32 bit	Data registers for port A and Port B
M2	write_enable_a, write_enable_b	1 bit	Write enable signals for port A and port B
M2	read_data_a, read_data_b	32 bit	Read data registers for port A and port B

2.4) Resource Usage and Critical Path
As we progressed through the project, we learned of and found improved methods of implementation. We restarted Milestone 2 as we found register over usage and underutilized DP memory. It created awareness to maximize usage of provided resources, rather than spending on additional and potentially unnecessary resources. Although Milestone 2 was redone, certain flags and buffers could be reused for a more efficient implementation throughout the project. Other than registers, it is likely that more multiplier and delay effective designs could be conceptualized; this would reduce the total number of clock cycles to complete each milestone; providing a high utilization and quick output. The path in the timing analyzer with the highest data delay, excluding inputs and outputs to the RAM top module, is the critical path. This gives V prime as the critical path which is accurate to expectations as the path is designed to involve a large range of logical operators.

3) Weekly Activity and Progress

	Project Progress	Jibin	Biswajit	Ronit
Week 1	-Reading Project document	-Reading Project document	-Reading Project document	-Reading Project document
Week 2	-Understanding the Milestones -Asking TA for project clarification	-Implementing M1 state table, creating common case	-Implementing M1 state table -Implementing lead in idea to state table	-Creating common case algorithms

	-Generating algorithm and state table for Milestone 1 -M1 state table TA verified	-Implementing lead to state table		-Reviewing project document with previous group
Week 3	-Implementing Milestone 1 -Debugging with group and TA	-M1 project set up. -Implementing lead in, common case and usage	-Debugging lead in and common case per state table	-Reviewing final M1 state table -Reviewing M2 -Group change
Week 4	-Debugging and Implementing Milestone 1 -Milestone 1 completed and verified with board -State table for Milestone 2 -Showed M2 state-table to TA -Implementing Milestone 2	-Implementing common cases and clipping -Debugging M1 mismatches -Understanding M2 -Implementing M2 state table -Verifying matrix multiplication and timing analysis -Implementing M2 lead in and common cases	-Debugging M1 mismatches -Understood Milestone 2 -Milestone 1 completed and verified with board -Verifying state table with TA for milestone 2 -Debugging milestone 2 lead in and common cases	-Implementing clipping -Debugging M1 mismatches in common cases -Hand calculation M2 matrix multiplication -Brainstorming M2 state table -Implementing M2 files setup -Implementing M2 lead in
Week 5	-Re-implementing Milestone 2. -State table and common case idea verified by Professor Nicolici. -Accepting the instructors *challenge* to amend M2 implementation idea 😊	-Implementing and debugging M2 -Re-creating M2 state table -Computing T and S manually -Implementing and debugging common case, clipping, and indexing -Debugging SRAM errors -Debugging SRAM row switch	-Debugging M2 common case -Re-creating M2 state table -Working on equations for computing T and S -Debugging errors with incorrect lead in	-Implementing and debugging M2 lead in. -Re-creating M2 state table. -Computing T and S matrix. -Implementing some of new M2 state table common case -Debugging lead in indexing and clipping

4) Conclusion

To conclude, the project for the course COMPENG 3DQ5: Digital System Design has provided a great opportunity for learning and experiencing from a meaningful engineering project. The Image Decompressor Hardware Implementation required a great deal of thinking, designing, re-designing, debugging, learning, and working with one another to complete. Not only have we improved our fundamental knowledge of digital systems, but we have all learned about time management, how to divide combinable thinking tasks, and how to communicate effectively as a team. Provided more time and improved time management skills, we believe we could have fully completed the project with integration and developed a greater understanding of digital systems design. Although the time commitment was intense, the learning experience and opportunity made this project a great accomplishment. The completed Milestone 1 submission was submitted 18-11-2023 labelled “Official Done Milestone 1 (FR)”. For Milestone 2, all of Y calculations were correct with no mismatches, and U and V calculations consisted of some mismatches due to indexing issues after the first block of U. This can be verified through the testbench and simulation results with the file from latest push to GitHub.

5) References

[1] N. Nicolici, COMPENG 3DQ5:Digital Systems Design, https://avenue.cllmcmaster.ca/d2l/le/content/554050/Home (accessed Nov. 27, 2023).