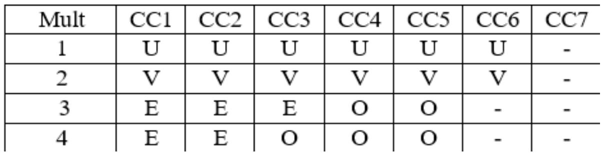
PROJECT REPORT 3DQ5 - Vinay Gajjar – 400236363, Anthony Acosta -400201942

Introduction: The projects' main objective is to create a Hardware Implementation of an Image Decompressor using a reverse engineering approach. The main concepts that are used to build the project is the understanding of VGA, SRAM. UART, finite state machines and learning how to break problems into smaller sections. The steps that are taken are the following in order: Color space conversion, Interpolation, Inverse Signal Transform, Dequantization, and Lossless Decoding. The tasks are breakdown in to three milestones where M1 consists of color space conversion and Interpolation. M2 is where the Inverse Signal Transform is performed and Lastly Milestone three consist of Dequantization, and Lossless Decoding.

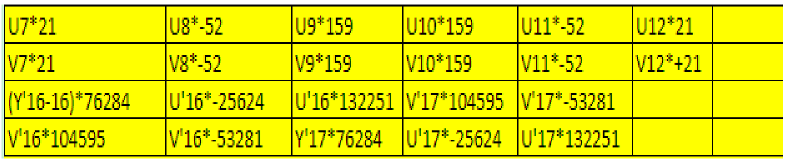
Implementation Details:

2.1 Milestone 1:

Milestone, one entails implementing Upsampling and Color Space Conversion in hardware, as outlined in Figures 1 and 2. Examination reveals that, for processing a pair of RGB values, a minimum of 12 multiplications are needed for upsampling, and at least 10 multiplications are required for CSC to get two RGB_even, and RGB_odd. This implies that within 6 clock cycles, two RGB's with 4 multipliers can be done. The table below illustrates the utilization of the four multipliers with 7CC.



Mult	CC1	CC2	CC3	CC4	CC5	CC6	CC7
1	U	U	U	U	U	U	-
2	V	V	V	V	V	V	-
3	E	E	E	O	O	-	-
4	E	E	O	O	O	-	-



U7*21	U8*-52	U9*159	U10*159	U11*-52	U12*21	
V7*21	V8*-52	V9*159	V10*159	V11*-52	V12*+21	
(Y'16-16)*76284	U'16*-25624	U'16*132251	V'17*104595	V'17*-53281		
V'16*104595	V'16*-53281	Y'17*76284	U'17*-25624	U'17*132251		

Utilization: For the common case the utilization is 22/28 this= 78.6%

LI: (18-4-2) *(4 multipliers)/(18)(4 multipliers) =66.67% , **LO**: (18– 4) *(4 multipliers)/(18* 4)= 78%

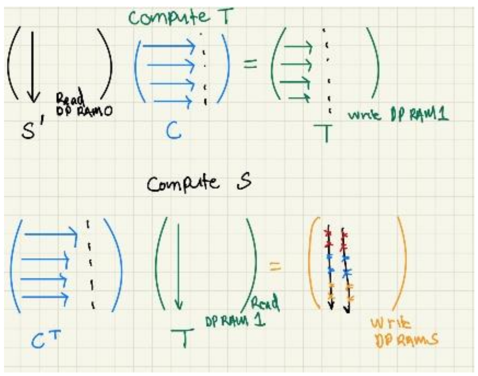
Average Utilization = 18*66.67% + 18*78% + 1082*78.6% / 1118 = 78.4% per row = **78.4% [within design spec >75%]**

Latency: 268,320 CC= (18 + 1082 +18) *240. For LI it takes 18 CC to write 3 RGB's. For LO it takes 18 to write 8 RGB's. Hence, 309 RGB's left to be processed by the common as there 2-RGB values per 7 CC =1082 CC adding all the cases you get 1118 CC per row. Then for image of size 320x240 we end up with Latency ~268320 CC

Module	Register	Size	Description
M1	SRAM_we_n	1 bit	Active low for writing
M1	SRAM_write_data	16 bits	Write to SRAM
M1	SRAM_address	18 bits	SRAM Address
M1	M1_Completed	1 bit	M1 finished indicator
M1	U_prime, V_prime , U/V_prime_buf	6x32 bits	U' V' storage
M1	U/V	6x8 bits	8-bit shift register to store UV
M1	R/G/B_EVEN, R/G/B_ODD	12x32 bits	Store 32-bit RGB values
M1	U/V/Y BUF	3x16 bits	YUV Buffers
M1	Mult_op_X_X	8x 32 bits	Multiplicands/Multipliers
M1	OP_1_8_BUF	8x 32 bits	Multiplier Buffers for LO/LI
M1	YUV/RGB_OFFSET_COUNTER	4x 18 bits	Read YUV, write RGB address
M1	Odd_Even	1 bit	Read U/V every other CC
M1	Special transition flags	2x1 bit	Allows for special conditions
M1	Column_Counter	12 bits	Tracks number of Y reads

2.2 IDCT (Milestone 2)

S Prime	**T**	**S**
Y00	T00	S00
Y01	T01	S01
Y02	T02	S02
DP RAM 0	**DP RAM 1**	**DP RAM 2**



C'00*T00	C'01 * T10	C'02 * T20	C'03 * T30	C'04 * T40	C'05 * T50	C'06 * T60	C'07 * T70
C'10*T00	C'11 *T10	C'12 *T20	C'13 *T30	C'14 *T40	C'15 *T50	C'16 *T60	C'17 *T70
C'20*T00	C'21 * T10	C'22 * T20	C'23 * T30	C'24 * T40	C'25 * T50	C'26* T60	C'27 * T70
C'30*T00	C'31 * T10	C'32 * T20	C'33 * T30	C'34 * T40	C'35* T50	C'36* T60	C'37 * T70

s'00*C00	s'01 * C10	s'02 * C20	s'03 *C30	s'04 * 40	s'05 * C50	s'06 *C60	s'07* C70	s'00*C04
s'00*C01	s'01 *C11	s'02 * C21	s'03 *C31	s'04 *41	s'05 *C51	s'06 *C61	s'07* C71	s'00*C05
s'00*C02	s'01 * C12	s'02 * C22	s'03 * C32	s'04* 42	s'05 *C52	s'06 *C62	s'07 *C72	s'00*C06
s'00*C03	s'01 * C13	s'02 * C23	s'03 * C33	s'04 *43	s'05 * C53	s'06 *C63	s'07*C73	s'00*C07

Overview: The memory was layed out as shown above with 1–32-bit value per location in each DP-RAM, this made read/writing simple, however for the Matrix multiplications coefficients C' and CT we use 4-16 BIT Muxes to read 4 C and CT values each cycle. This allows us to read a single value per cycle from DPRAM. The mux is structured such a way that each value in a mux is unique. When performing multiplications, the primary limitation lies in the fact that, at most, two values can be retrieved from the DPRAM. In our strategy, we opt to acquire one value at a time while processing four C values in a single clock cycle as show by the figure above. For compute T we are calculating in way such as that at 8 CC we have 4 values are ready. For example, at the very first 8 CC you have S01, S02, S03, S04 are ready. As for Compute S S00, S10, S20, S30 will be ready.

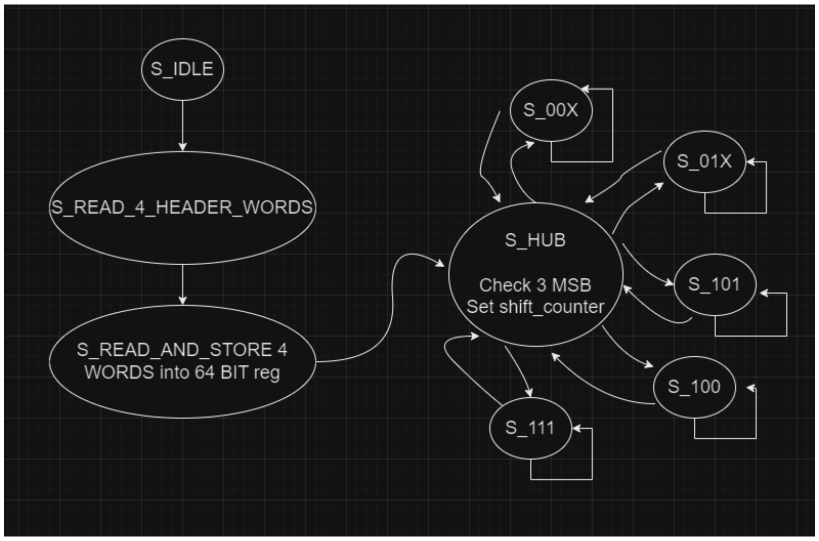
Utilization: 2400(128+128) / (64+(2400*(132+132) +32) = 97%. 64 Fetch S prime, 32 Last S

Latency: Fetch S' = 66 CC, Compute T =132CC, Compute S Fetch S prime = 132 CC, Write S Compute T'=132CC

(132+132) *2398 Blocks+ 66CC Lead In Fetch S prime + 66CC Lead Out Write S = 633,204CC

	Register	Size	Description
M2	row/col_index_DPRAM_S_Prime,	2x3bits	ri,ci for internal elements 0-7
M2	blocks_read/written	2x12 bits	Counter num of reads and writes
M2	address_s_prime_0, address_T_0/1, address_S_0/1	5x6 bits	S prime, T addresses
M2	first_pass	1 bit	Flag for first LI from Fetch S'
M2	write_s_prime_0/1, write_T_0/1, write_S_0/1	6x32 bits	Store S' & T in 32-bit values when read
M2	WE_s_prime_0/We_T/We_S	3x1 bit	Active High write EN for DP
M2	T_0, T_1, T_2, T_3, T_2_BUF, T_3_BUF	6x32 bits	Store computed T values
M2	Write_Sample_counter, Sample_counter	6 bits	8x8 read/write counter
M2	column_block	6 bits	PRE-IDCT column blocks read
M2	row_block	5 bits	PRE-IDCT row blocks read
M2	dram_address_counter_s_prime	7 bits	Dram addressing for Sprime
M2	C_multiplier_row, C_multiplier_column	8 bits	Used as the inputs for the mux
M2	flag_increment_row , flag_increment_column, FS_CC_transition, Lead_Out_Flag, M2_done	1 bit	Flags that are used for special cases
M2	dram_address_counter_T	8 bits	Addressing T from DP ram
M2	mux_counter	4 bits	Selects from 4–16-bit C muxes.
M2	ROW_ADDRESS_T, COL_ADDRESS_T	3 bits	Row/ column address counter for T.
M2	write_column_block, write_row_block	6 bits	POST IDCT blocks written
M2	S_0, S_1, S_2, S_3, S_2_BUF, S_3_BUF	6x32bits	Store S values

2.3 Lossless Decoding and Dequantization (Milestone 3)



Idea/Design: In this approach we have a 64 bit register that stores 4 words at a time, then we have a pointer that allows us to traverse the 64-bit register based on the header code, and the number of reads we need to do. The pointer logic was done using a combinational block that stored a copy of the 64-bit shift register that would be shifted each time based on the reads (meaning the original shift register is always static). We then have two registers to track number of shifts called “shift_counter” and “shift_sum” that based on the header would tell us how many bits to shift and how many shifts we’ve done so far respectively. This “shift_sum” would allow us to check if we have read more than 16 bits (1 Word) we can initiate a new read from SRAM and two clock cycles later we would need to feed this into the 16 most MSB of our original (unshifted) 4 word register. Then every subsequent 16 bit “shift_sum” increment we would read and fill in the registers from left to right starting with MSB, such that after 4 subsequent 16 bit reads all our 64 registers would have new values within them.

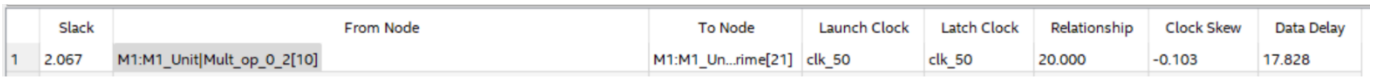
With regards to dequantization, we use purely combinational logic with a mux to track the zig zag counter based on number of writes/reads. The dequantization shifting is also controlled by the zig zag counter, where the sum of the row+col index gives us the select line that we feed into a mux that gives us shifts based on the position we are in the zig zag counter. These values are then written directly to DP-RAM (for this implementation).

Progress State/Issue: Dequantization and ZigZag pattern writing works, however lossless decoding fails after first 5 writes. We observed that the first few reads and writes to the SRAM/DPRAM (tested with DP ram as test) were correct, however once we started shifting in new values from the SRAM reads from the bitstream, these were being placed into our 64-bit register at the wrong time, meaning that the subsequent headers, hence lossless decode and quantization would fail. This could mean that our timing and control of when to feed in data was incorrectly setup for the current test code.

2.4 Resource Usage and Critical Path

With regards to resource usage in M1 we are using about 8000 LE, and similar for M2 meaning that we have have about 8000 – 4 input LUT’s allowing for 8000 outputs to do our functions. This is an order of magnitude smaller than Lab 5 experiment 4.

The worst-case timing can be seen as the path between the 32-bit multiplier and the V_prime (as shown in figure x) for both M1 and M2. This is justifiable since each embedded 9-bit multiplier has built in logic that would need to propagate and scaled for 32 bits, meaning that we would have the longest delay compared to the rest of our hardware where we don’t have such long concatenated paths (specifically with registers). Also, we think that since the multiplier would have additional combinational logic and is has synchronous data, this would lead to the longest critical path in our design, compared to other simple logic, that may not go through multiple logic every clock cycle.



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	2.067	M1:M1_Unit|Mult_op_0_2[10]	M1:M1_Un...rime[21]	clk_50	clk_50	20.000	-0.103	17.828

3. Weekly Activity and Progress:

Week Number	Together	Anthony	Vinay
Week 0	Brainstorming	Brainstorming	Brainstorming
Week 1	Common Case	Started Coding Lead-out	Started Coding Lead in
Week 2	Debug milestone one together	Finish coding the lead out	Coded the Common case and lead in
Week 3	Brainstormed M2 together		
Week 4	Coded M2 together and Debug Together Finished M2	Got M1 and M2 tested on the board	Final Debugs and Clipping Check.
Week 5	Brainstorm Together Coded together and debug together. M3 partially completed.		

4.Conclusion

To conclude Project 3DQ5, we have gained significant insights into digital design, engineering methodologies, learning strategies, and, most importantly, the art of perseverance. We successfully navigated obstacles by breaking down the project into manageable sections. Additionally, we learned the importance of taking a step back, consistently questioning our approach, and understanding the overarching purpose of our actions. By doing so we are able conquer the project with the best of our abilities and leave no stone unturned. It was truly a pleasure to work with you this year Dr. Nicolici you have provided us a glimpse of what life which is to preserve no matter what. Please and continue to inspire young engineers.

5 . GIT COMMITS

- M2 COMPLETED – DATE: 11-25-2023, NAME: "M2 GOLDEN VERSION"
- M3 PARTIAL - DATE: 11-27-2023, NAME: "M3_PARTIAL_CODE_PUSHED"

6 . REFERENCES

- Phillip Krivor - High Frequency Image
- TA's for verifying state table for M1.