COMPENG 3DQ5
Project Report

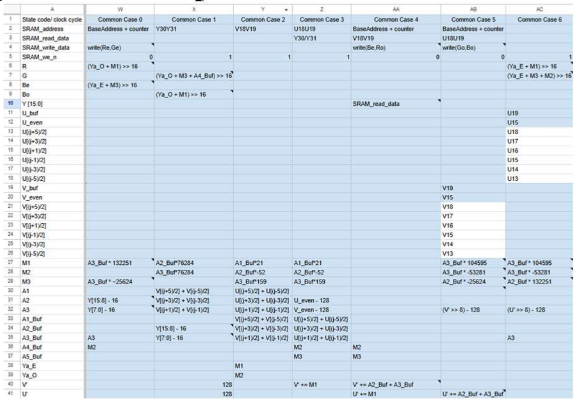
Course Instructor: Dr. Nicolici

Makarios Samwail, Umar Javaid
November 25th, 2024
Group 80
samwailm
400455949
javaim7
400455837

Introduction

This project is centered around designing a hardware decompression system for the custom McMaster Image Compression (mic18) format. The system first receives the mic18 file over the UART serial communication protocol and stores the data in the external SRAM. The decompression system then decodes this data into it's corresponding RGB data through the use of custom-made digital sub-systems. The RGB image data is then outputted to the screen using a VGA controller.

Milestone 1: Upsampling and Colour Space Conversion

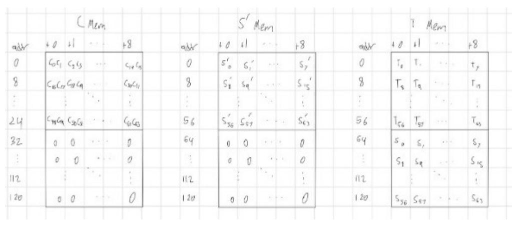


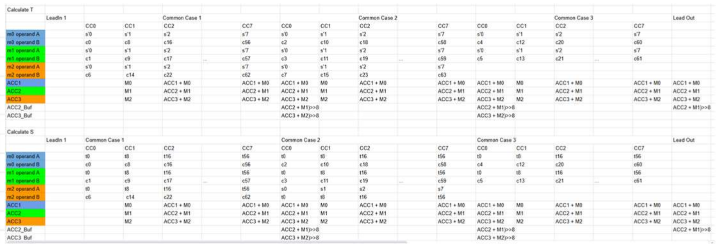
	A	W	X	Y	Z	AA	AB	AC
1	State code/ clock cycle	Common Case 0	Common Case 1	Common Case 2	Common Case 3	Common Case 4	Common Case 5	Common Case 6
2	SRAM_address	BaseAddress + counter	Y30Y31	V18V19	U18U19	BaseAddress + counter	BaseAddress + counter	
3	SRAM_read_data				Y30/Y31	V18V19	U18U19	
4	SRAM_write_data	write(Re,Ge)				write(Be,Ro)	write(Go,Bo)	
5	SRAM_we_n	0	1	1	1	0	0	1
6	R	(Ya_O + M1) >> 16						(Ya_E + M1) >> 16
7	G		(Ya_O + M3 + A4_Buf) >> 16					(Ya_E + M3 + M2) >> 16
8	Be	(Ya_E + M3) >> 16						
9	Bo		(Ya_O + M1) >> 16					
10	Y [15:0]					SRAM_read_data		
11	U_buf							U19
12	U_even							U15
13	U[(j+5)/2]							U18
14	U[(j+3)/2]							U17
15	U[(j+1)/2]							U16
16	U[(j-1)/2]							U15
17	U[(j-3)/2]							U14
18	U[(j-5)/2]							U13
19	V_buf						V19	
20	V_even						V15	
21	V[(j+5)/2]						V18	
22	V[(j+3)/2]						V17	
23	V[(j+1)/2]						V16	
24	V[(j-1)/2]						V15	
25	V[(j-3)/2]						V14	
26	V[(j-5)/2]						V13	
27	M1	A3_Buf * 132251	A2_Buf*76284	A1_Buf*21	A1_Buf*21		A3_Buf * 104595	A3_Buf * 104595
28	M2		A3_Buf*76284	A2_Buf*-52	A2_Buf*-52		A3_Buf * -53281	A3_Buf * -53281
29	M3	A3_Buf * −25624		A3_Buf*159	A3_Buf*159		A2_Buf * -25624	A2_Buf * 132251
30	A1		V[(j+5)/2] + V[(j-5)/2]	U[(j+5)/2] + U[(j-5)/2]				
31	A2	Y[15:8] - 16	V[(j+3)/2] + V[(j-3)/2]	U[(j+3)/2] + U[(j-3)/2]	U_even - 128			
32	A3	Y[7:0] - 16	V[(j+1)/2] + V[(j-1)/2]	U[(j+1)/2] + U[(j-1)/2]	V_even - 128		(V' >> 8) - 128	(U' >> 8) - 128
33	A1_Buf			V[(j+5)/2] + V[(j-5)/2]	U[(j+5)/2] + U[(j-5)/2]			
34	A2_Buf		Y[15:8] - 16	V[(j+3)/2] + V[(j-3)/2]	U[(j+3)/2] + U[(j-3)/2]			
35	A3_Buf	A3	Y[7:0] - 16	V[(j+1)/2] + V[(j-1)/2]	U[(j+1)/2] + U[(j-1)/2]			A3
36	A4_Buf	M2			M2	M2		
37	A5_Buf				M3	M3		
38	Ya_E			M1				
39	Ya_O			M2				
40	V'		128		V' += M1	V' += A2_Buf + A3_Buf		
41	U'		128			U' += M1	U' += A2_Buf + A3_Buf	

Module (instance)	Register Name	Bits	Description
Milestone1	V_Buf_5p:V_Buf_5n U_Buf_5p:U_Buf_5n	8x6 8x6	These registers are set up in a shift register configuration to perform the interpolation for odd U and V pixels
Milestone1	A1:A3	32x3	These registers are used to store and accumulate the results of our colour space conversion
Milestone1	[31:0]operand[5:0]	32x6	These 6 registers are used to store the operands that we wish to multiply together.
Milestone1	Mult_result_long1: Mult_result_long3 Mult_result1: Mult_result3	64x3 32x3	These 3 Registers are used to store the long result of our multiplication These 3 Registers are used to store the first 32 bits of our long multiplication result
Milestone1	Adder1_Buf: Adder5_Buf	32x5	These 5 registers are used to buffer data needed for colourspace calculation to prevent multiplying and adding the same value in one clock cycle. This approach was implemented to reduce the combinational logic depth to prevent timing violations.
Milestone1	RedE RedO GreenE GreenO BlueE BlueO	31x5	These 6 registers store the calculated RGB values for odd and even pixels. Using separate registers for odd and even pixels prevent the values from being overwritten.
Milestone1	Ycounter, VUcounter, RGBcounter	16x1 15x1 18x1	These registers keep track of the address of YUV and RGB pixels in order to read YUV and write RGB data to the SRAM.

Milestone 1 took 271203 cycles to complete. The colorspace conversion and upsampling is done within 7 clock cycles. This means that each row has 1120 (7x160) common cases. Furthermore, each row has 14 LEADIN states and 3 LEADOUT states resulting in 1137 clock cycles per row in total. During our 7 common cases, we use our multipliers 16 times out of the 21 possible times which gives us a multiplier utilization of 76%. Hence, the total utilization of milestone 1 is $\frac{1120*76.19\%}{1137} = 75.05\%$ which meets the project specifications.

Milestone 2: Inverse Discrete Cosine Transform





Calculate T																	
	Leadin 1			Common Case 1					Common Case 2					Common Case 3			Lead Out
		CC0	CC1	CC2		CC7	CC0	CC1	CC2		CC7	CC0	CC1	CC2		CC7	
m0 operand A		s'0	s'1	s'2		s'7	s'0	s'1	s'2		s'7	s'0	s'1	s'2		s'7	
m0 operand B		c0	c8	c16		c56	c2	c10	c18		c58	c4	c12	c20		c60	
m1 operand A		s'0	s'1	s'2		s'7	s'0	s'1	s'2		s'7	s'0	s'1	s'2		s'7	
m1 operand B		c1	c9	c17	...	c57	c3	c11	c19	...	c59	c5	c13	c21	...	c61	
m2 operand A		s'0	s'1	s'2		s'7	s'0	s'1	s'2		s'7						
m2 operand B		c6	c14	c22		c62	c7	c15	c23		c63						
ACC1			M0	ACC1 + M0		ACC1 + M0	ACC1 + M0	M0	ACC1 + M0		ACC1 + M0	ACC1 + M0	M0	ACC1 + M0		ACC1 + M0	ACC1 + M0
ACC2			M1	ACC2 + M1		ACC2 + M1	ACC2 + M1	M1	ACC2 + M1		ACC2 + M1	ACC2 + M1	M1	ACC2 + M1		ACC2 + M1	ACC2 + M1
ACC3			M2	ACC3 + M2		ACC3 + M2	ACC3 + M2	M2	ACC3 + M2		ACC3 + M2	ACC3 + M2	M2	ACC3 + M2		ACC3 + M2	ACC3 + M2
ACC2_Buf							ACC2 + M1)>>8					ACC2 + M1)>>8					ACC2 + M1)>>8
ACC3_Buf							ACC3 + M2)>>8					ACC3 + M2)>>8					

Calculate S																	
	Leadin 1	Common Case 1					Common Case 2					Common Case 3					Lead Out
		CC0	CC1	CC2		CC7	CC0	CC1	CC2		CC7	CC0	CC1	CC2		CC7	
m0 operand A		t0	t8	t16		t56	t0	t8	t16		t56	t0	t8	t16		t56	
m0 operand B		c0	c8	c16		c56	c2	c10	c18		c58	c4	c12	c20		c60	
m1 operand A		t0	t8	t16		t56	t0	t8	t16		t56	t0	t8	t16		t56	
m1 operand B		c1	c9	c17	...	c57	c3	c11	c19	...	c59	c5	c13	c21	...	c61	
m2 operand A		t0	t8	t16		t56	s0	s1	s2		s7						
m2 operand B		c6	c14	c22		c62	t0	t8	t16		t56						
ACC1			M0	ACC1 + M0		ACC1 + M0	ACC1 + M0	M0	ACC1 + M0		ACC1 + M0	ACC1 + M0	M0	ACC1 + M0		ACC1 + M0	ACC1 + M0
ACC2			M1	ACC2 + M1		ACC2 + M1	ACC2 + M1	M1	ACC2 + M1		ACC2 + M1	ACC2 + M1	M1	ACC2 + M1		ACC2 + M1	ACC2 + M1
ACC3			M2	ACC3 + M2		ACC3 + M2	ACC3 + M2	M2	ACC3 + M2		ACC3 + M2	ACC3 + M2	M2	ACC3 + M2		ACC3 + M2	ACC3 + M2
ACC2_Buf							ACC2 + M1)>>8					ACC2 + M1)>>8					ACC2 + M1)>>8
ACC3 Buf							ACC3 + M2)>>8					ACC3 + M2)>>8					

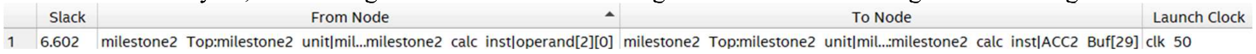
Module (instance)	Register Name	Bits	Description
Milestone2Top	calculateT_en, calculateS_en, fetch_Sprime_en, write_S_en	1x4	These Registers are used to Control what we want to preform
Milestone2Top	calculateT_finish, calculateS_finish, m2_finishFS_top, m2_finishWS_top	1x4	These registers are used to store and accumulate the results of our colour space conversion
Milestone2Top	c_address, c_write_data, c_WE, c_read_data s_prime_address, s_prime_write_data, s_prime_WE, s_prime_read_data t_address, t_write_data, t_WE, t_read_data	75x2 75x2 75x2	All these registers are used to drive the embedded memories in the milestone 2 top
Milestone2Top	DP_..._Calc	304	All these registers are connected to capture the outputs/send inputs into the milestone 2 calc module.
Milestone2Top	DP_..._FW	160	All these registers are connected to capture the outputs/send inputs into the milestone 2 fetch write module.
Milestone2Calc	A1:A3 ACC2_Buf ACC3_Buf	32x3	These registers are used to store and accumulate the results of our calculation. These buffers hold values that get overwritten.
Milestone2Calc	[31:0]operand[5:0]	32x6	These 6 registers are used to store the operands that we wish to multiply together.
Milestone2Fetchandwrite	SampleCounter1, SampleCounter2	6 bits, 5 bits	The SampleCounter manages the tracking of the i,j indices for reading S′ values from the DualPort S memory and writing S values into the SRAM.
Milestone2Fetchandwrite	CA, CA_WS, RA, RA_WS	9 bits, 8 bits, 8 bits, 8 bits	Computes the column and row address for fetching S' and writing S.

Milestone2Fetchandwrite	column_count [1:0], row_count[1:0]	6 x 2 , 5 x 2	Tracks the column and row position during S' fetching and S writing.
Milestone2Fetchandwrite	ADDR_FS, ADDR_WS	18 bits, 18 bits	Calculates SRAM address for fetching S' and writing S.
Milestone2Fetchandwrite	M2_addr_counter	32	Tracks the current address in the SRAM during Fetch and Write operations.
Milestone2Fetchandwrite	address_en	1	Ensures that the correct memory address is used during read and write operations

For the latency analysis for milestone 2, it should be taking 66+2400(197*2)+66= 945732. The 66 clock cycles represent the initial read and write elements. The middle term represents the clock cycles for the mega states.

Resource Usage and Critical Path

For the most recent commit, Quartus is reporting 2579/114480 total logic elements. This is compared to the 616/114480 total logic elements from experiment 4. This is obviously a big jump which makes sense since the decompression system takes a lot more resources to implement digital circuits like the colour space converter and IDCT. In Milestone 1, we implemented 5 buffers to store values temporarily to prevent simultaneous multiplication and addition within the same clock cycle to avoid timing errors. However, I realized that by restructuring our calculations to avoid overwriting values, we can sequence the operations so that additions occur after multiplication in the next clock cycle, eliminating the need for additional registers while maintaining correct timing.



	Slack	From Node	To Node	Launch Clock
1	6.602	milestone2 Top:milestone2 unit|mil...milestone2 calc inst|operand[2][0]	milestone2 Top:milestone2 unit|mil...:milestone2 calc inst|ACC2 Buf[29]	clk 50

The critical path for the most recent commit is operand [2] to ACC2_Buf. This makes sense since hardware multiplication is a difficult process within an FPGA. It requires a lot of digital logic to implement which translates to a higher propagation delay when calculating multiplication.

Weekly Activity and Progress

	Umar	Makarios
Week 1 Planning	Attended lectures and made notes on concepts that are essential to the project.	Made notes and started thinking of how to implement milestone 1
Week 2	Planned out how to do the colorspace conversion and interpolation within 7 clock cycles. Then, implemented the common cases for Milestone 1 state table.	Reviewed the state table and found areas of concerns for the state table. Implemented the lead in and lead out cases as well as the counting logic.
Week 3 Debugging	Programmed interpolation and colorspace conversion using the state table. Started debugging the calculations by comparing to theoretical calculations and by checking hex viewer.	Helped with debugging the milestone and analyzing the waves. Especially for the lead out and lead in case.
Week 4 Implement	Me and my partner completed Milestone 1 with no mismatches. Then I created state tables for Fetch S' and Write S and started working on implementing the code.	After completing the milestone 1, I came up with the overall plan and architecture for milestone 2. I delegated the work between me and Umar.
Week 5 Implement	Finished Fetch S' and Write S. Performed unit testing to reduce bugs before top level integration. Implemented the top level integration with my partner but we were not able to get it fully working.	Began by developing the top level design for milestone 2. Debugged calculate T and S module and finished the implementation this week. Not able to complete the debugging for this in simulation.

We interacted with Group 31 during the initial planning phase of milestone 2 but implemented our milestones differently.

Conclusion (Commit- Message: "Finished milestone 1 code", Date: 20 November)

To summarize, there was a lot of new learning experiences we had during this project. We learned how to tackle complex problems into smaller problems that can be easily completed. We also learned how to modularize code into separate modules that get instantiated in top level files. The most important thing we learned though was learning how to more efficiently create state tables and debug using modelsim.

[1] N. Nicolici, "3DQ5 project report guidelines," Login - McMaster University, https://avenue.cllmcmaster.ca/d2l/le/content/633401/viewContent/4901064/View (accessed Nov. 25, 2024).