Final Project Report

Omar Abdulrahman 400371849, Ali Shariq 400377855
COMPENG 3DQ5: Digital Systems Design
McMaster University
Monday, November 27th, 2023

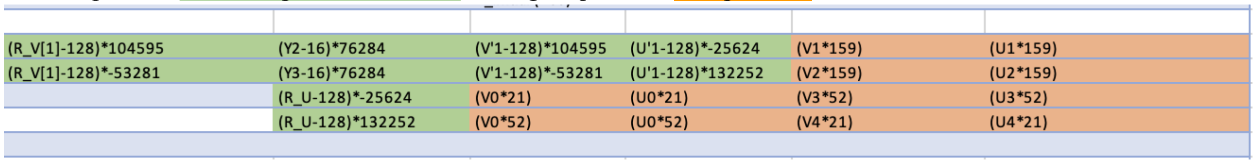
Introduction

The objective of the hardware implementation project is to design and implement a custom image decompressor based on the McMaster Image Compression revision 17 (.mic17) specification. The process begins with the reception of compressed data for a 320x240 pixel image through a UART interface from a personal computer, storing it in external SRAM on an Altera DE2 board. The decompression circuitry reads and processes the compressed data, recovering the image using a DCT and quantization approach. The resulting decompressed image is stored back in SRAM and displayed on a monitor through a VGA controller. The overall goal is to solidify our current knowledge in digital system design, reinforcing concepts such as image compression, hardware implementation, and real-time image display.

Implementation

Milestone 1: Upsampling and Colour Space Conversion

In this milestone, we use the method of interpolation and color space conversion to convert our YUV values to RGB. We begin with the even YUV values by using two sets of registers in which the R_initial, G_initial, and B_initial values are stored. These values are clipped and then stored within 3 different respective registers, R, G, and B. These registers now include the post-clipped values. To deal with the odd values, interpolation first occurs. A multiplier is used in combinational logic with six U and six V registers, resulting in the required U and V values. We then repeat the color space conversion process on these odd numbers, storing them in the R, G, and B registers. Below is a figure of the process of interpolation and color space conversion through the common case:
Green represents Colour Space Conversion, Orange represents Interpolation



(R_V[1]-128)*104595	(Y2-16)*76284	(V'1-128)*104595	(U'1-128)*-25624	(V1*159)	(U1*159)
(R_V[1]-128)*-53281	(Y3-16)*76284	(V'1-128)*-53281	(U'1-128)*132252	(V2*159)	(U2*159)
	(R_U-128)*-25624	(V0*21)	(U0*21)	(V3*52)	(U3*52)
	(R_U-128)*132252	(V0*52)	(U0*52)	(V4*21)	(U4*21)

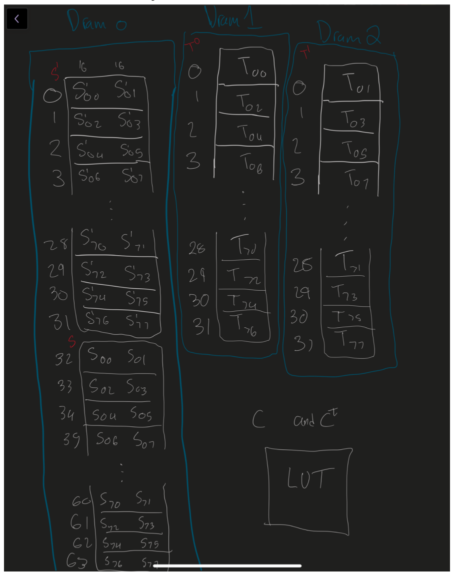
*Note: All registers are a part of the milestone 1 module except the last one, which is a top-level instance

Register Name	Bits	Description
U_shift_register, V_shift_register	48	Hold the six U and V values that are used for upsampling from U' and V' to U and V (j-5, j-3, j-1, j+1, j+3, j+5)
U_buff, V_buff	8	Holds the second half of the SRAM read for U and V for one clock cycle before sending the value into the register
Y_register[1:0], V_register[1:0]	32x2	Registers are used as temporary hold storage as we compute the different multiplications before upsampling / colour conversion
U_register	32	
R_initial[1:0], G_initial[1:0], B_initial[1:0]	32x2	Pre Clipping RGB storage register
R[1:0], G[1:0], B[1:0]	8x2	Post-Clipping RGB storage register, used to write in SRAM
Product[3:0]	32x4	Holds the Product of Op0 * Op1

Op0[3:0], Op1[3:0]	32x4	Operator registers that holds the values we are planning to multiply
eol	1	Stands for “Every Other Loop”, this is a flag so we only set V and U address every other loop (in order to not call too many V and U address’ compared to Y)
Y_counter, X_counter	18	X and Y counter to track where we are in the image
Y_address, U_address, V_address,RGB_address	18	Hold respective SRAM address values that are iterated through, constantly changing.
M1_SRAM_address	18	Sets SRAM address
M1_SRAM_write_data	15	SRAM write data for M1
M1_status	1	M1 flag to track whether M1 is completed or not
M1_SRAM_we_n	1	M1 SRAM write enable
M1_start	1	Indicate the beginning of milestone 1

To calculate the number of clock cycles milestone 1 goes through, we complete the following latency analysis: First we consider the fact that there are 6 states that occur in the common case. We know that the common case is repeated 160 times in a row, so we multiply these values, furthermore we have to consider 9 for the lead in values, so we add these in. From this we have now acquired the number of clock cycles per row. The image is a size of 240 rows, so we multiply this by 240, where we've finally acquired the number of clock cycles in milestone 1. This relates to the way we've utilized multipliers in this milestone as we complete the same process overall in order to acquire our converted RGB values. **((6*160) + 9)*240 = 232 560 cycles**

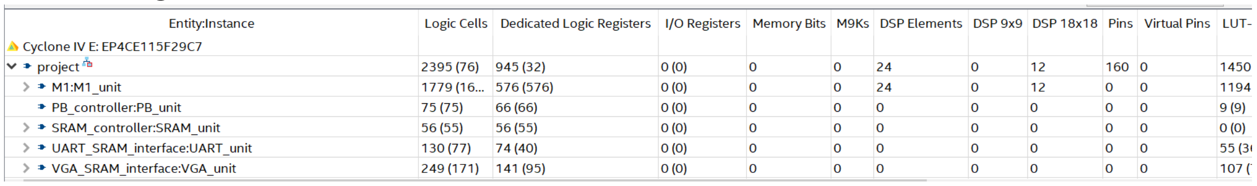
Milestone 2: Inverse Discrete Cosine Transform
*NOTE: Analysis is done to the best of our ability with how much we have completed. Latency analysis incomplete.



Register Name	Bits	Description
TCCloopflag, TCCflag, TCC_OP_flip, TCCrightflag	1	1 bit flags to indicate the start/ stop of specific areas in the code

RCIndex, Rowtrack, Coltrack	6	RCindex counts what row and col we are in for each block, rowtrack and coltrack track what black we are in
SC_counter	7	Used to calculate what S and C values we need to find T
Product[3:0]	32x4	Holds the Product of Op0 * Op1
Op0[3:0], Op1[3:0]	32x4	Operator registers that holds the values we are planning to multiply
sp_register	128	Holds one row of S' to compute
T_hold_register	28	Temp holds T value until we get second half of T
rowRegister	12	Counts for what row we are in
colRegister	9	Counts for what col we are in
YUVflag	2	Used to indicate what area of the SRAM we reading in
baseAddress	18	Register to hold base address value

Resource Usage and Critical Path

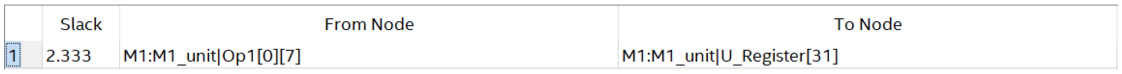


Entity:Instance	Logic Cells	Dedicated Logic Registers	I/O Registers	Memory Bits	M9Ks	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins	LUT-
Cyclone IV E: EP4CE115F29C7											
project	2395 (76)	945 (32)	0 (0)	0	0	24	0	12	160	0	1450
M1:M1_unit	1779 (16...	576 (576)	0 (0)	0	0	24	0	12	0	0	1194
PB_controller:PB_unit	75 (75)	66 (66)	0 (0)	0	0	0	0	0	0	0	9 (9)
SRAM_controller:SRAM_unit	56 (55)	56 (55)	0 (0)	0	0	0	0	0	0	0	0 (0)
UART_SRAM_interface:UART_unit	130 (77)	74 (40)	0 (0)	0	0	0	0	0	0	0	55 (3
VGA_SRAM_interface:VGA_unit	249 (171)	141 (95)	0 (0)	0	0	0	0	0	0	0	107 (

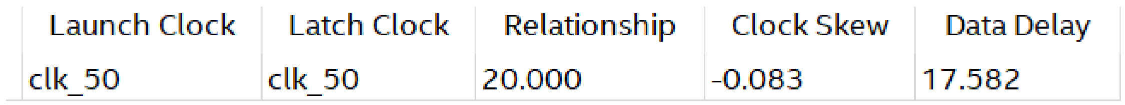
We have used a total of 2395 logic elements according to Quartus, which is due to the complexity and large number of elements required for this project in comparison to something like lab 5. It can be seen that the majority of the elements came from milestone 1, which handled a large number of registers along with logic elements in order to complete the required data manipulation, clipping and shift through multiple registers for memory allocation. UART communication as well as VGA interfacing required logic elements in order to properly integrate and display the desired RGB values onto the display.

During our project analysis, we noted that the register V_register[0] is currently used to temporarily store the V*104 595 values essential for the G calculation in the color space conversion. A potential optimization involves modifying the code to store these values directly in the G_initial register. In the subsequent state, the G color space conversion could then be accomplished by adding G_initial to the other multiplied values obtained in that state, getting rid of the necessity for the intermediate use of the register V_register[0]. This adjustment enhances efficiency in handling the color space conversion process.

To the best of our understanding, the reason for this critical path occurring is due to different factors. The overall complexity of the combinational logic within the project has a big influence on the length, with a large number of logic elements such as registers, multiplexers, adders, and multipliers. This can be seen from the resource usage analysis that was completed. As we can see, the path starts from the beginning of M1, at the unit operand 1, and ends at M1 U_register[31]. This displays the overall longest critical path occurring as being between these two points in the process. Furthermore, the number of states in this process adds to the length of the path, with further delays occurring. Overall propagation delays also increase, as more input and outputs are initialized and used. Finally, the number of registers in this portion of the project is very high, meaning there is a need for specific constraint settings to be considered such as the hold time, setup time and clock-to-q delays.



	Slack	From Node	To Node
1	2.333	M1:M1_unit|Op1[0][7]	M1:M1_unit|U_Register[31]



Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
clk_50	clk_50	20.000	-0.083	17.582

Weekly Progress

Week	Progress & Contribution
Week 1	Revise previous labs. Reading and understanding the project document. Concluding project requirements and objectives. **Ali:** Read document and previous labs, understand project scope and flow **Omar:** Read documents and previous labs, understand continuity between milestones. Input and output organization
Week 2	Began milestone 1. Planned out and worked on the state table. Started coding. **Together:** Implemented a state table including common case and lead-out case situations. Begin planning how to implement within code
Week 3	Continued milestone 1, completed the state table and worked on implementing elements within the code such as FSM logic, and simulation. **Omar:** Worked on state table implementation, debugged and verified results in the lab using the board. **Ali:** Worked on state table implementation. Aided in testbench modifications and testing/debugging
Week 4	Continued milestone 1. State table entered multiple iterations, due to different mistakes and errors that were found. **Together:** Worked on getting the state table finalized correctly. Completed in code implementation and tested on the FPGA. Continuous verification and debugging process.
Week 5	Completed milestone 1. Began milestone 2. Consistently worked on and completed required state tables. Work on code implementation. **Omar:** Solidified understanding of milestone 2 expectations. Worked on lead-in/lead-out state tables. **Ali:** Solidified understanding of milestone 2 expectations. Worked on common case state tables. **Together:** Programmed and tested chosen state tables on the FPGA, with consistent testing and verification occurring.

Conclusion

Embarking on the hardware implementation image decompressor project has been a transformative learning experience. Delving into digital system design, we gained a comprehensive understanding of image compression processes, from colorspace conversion to quantization and lossless coding. The hands-on nature of the project, involving the FPGA and real-world application of concepts such as discrete cosine transform, allowed us to bridge theoretical knowledge with practical implementation. In general, the challenges that we faced during the design process allowed us to discover new avenues and hone our skills in troubleshooting and critical thinking. Even through the incompletion of the project, we gained a comprehensive view of the development lifecycle, actively engaging in project management and documentation. Overall, this project has not only enhanced our technical skills in digital system design but has also highlighted the importance of optimized hardware solutions in addressing real-world challenges, specifically in image compression and decompression. Milestone 2 was incomplete but some verilog implementation was done. The commit for milestone 1 completion is the following: “m1 complete” November 23rd 2023

References

Nicolici, Nicola. (2023). COMPENG 3DQ5: 3DQ5 project description [Course Document]. Fall 2023, McMaster University. https://avenue.cllmcmaster.ca/d2l/le/content/554050/viewContent/4381835/View