FPGA JPEG Standard Hardware Image Decompressor

Authors:

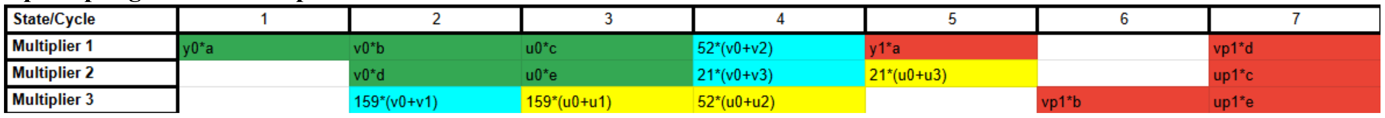
Ali Naqvi

Daniel Wong

Introduction

This project entails the design and implementation of a system capable of decoding compressed image data for a 320x240 pixel image. The compressed data is transmitted via a UART interface to an Altera DE2-115 FPGA board, where it is stored in external static random access memory (SRAM). A custom digital circuit is developed to decompress the data, reconstruct the image, and store the output back into the SRAM. The VGA controller then reads the decompressed image from the SRAM to display it on a screen. This project integrates various components, including finite state machines, SRAM, VGA, and UART interfaces, and applies key principles of hardware-based image processing such as lossless decoding, dequantization, signal transform, interpolation and colour space conversion.

Upsampling and Colour Space Conversion



State/Cycle	1	2	3	4	5	6	7
Multiplier 1	y0*a	v0*b	u0*c	52*(v0+v2)	y1*a		vp1*d
Multiplier 2		v0*d	u0*e	21*(v0+v3)	21*(u0+u3)		up1*c
Multiplier 3		159*(v0+v1)	159*(u0+u1)	52*(u0+u2)		vp1*b	up1*e

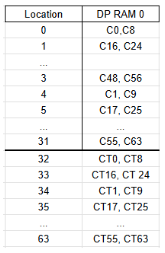
Figure 1: Overview of how the calculations for Colour Space Conversion & Interpolation are organized.

The milestone 1 state machine is responsible for performing YUV to RGB conversion via interpolation and colour space conversion. Using a fixed constrained amount of three multipliers, the calculations for both interpolation and colour space conversion are completed in 7 clock cycles of the common case, highlighted in Figure 1.

Module (instance)	Register name(s)	Bits	Description
M1_Unit	lead_out_cycle	2	Increments with each lead out cycle and is the select signal for a priority encoder that chooses which multiplier operands to use for each cycle.
M1_Unit	lead_out_counter	8	Enabled when is equal to 158 indicating the transition to lead out states.
M1_Unit	U_temp, V_temp, Y_temp, Y_reg	8/16	Temperorary register to hold YUV values to be used later
M1_Unit	RGB_temp	32	Even R and G values are written together while the even B value is put in the temp register to be written later on when R odd is calculated.
M1_Unit	Y_counter, UV_counter, RBG_counter	32	Counters to increment the SRAM address every time a value is extracted (in the case of YUV) or if a value is being written (RGB data).
M1_Unit	R_reg, B_reg, G_reg, V_prime, U_prime	32	Hold the accumulated values of the calculations for RGB and U' and V'.
M1_Unit	R_write, B_write, G_write, RGB_temp_write	8	The accumulators above are clipped and transformed into these 8 bit registers holding the final clipped RGB values.
M1_Unit	UV_read_cycle	1	Every other common state cycle, values of U and V are read to update the shift register, this flag is raised anytime a read cycle is to occur.
M1_Unit	U_shift, V_shift	48	Shift registers holding consecutive U and V values.

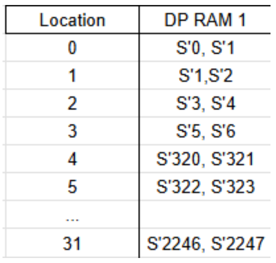
Latency Analysis: It takes a total of 156 commons, 3 lead out cases and 1 lead in case adding up to 160 for one row to complete. Since there are 240 rows, we can multiply these values by 240: One common case uses 7 clock cycles (240*156*7 = 262,080 clock cycles total), the lead in states take 12 clock cycles (240*1*12 = 2880 clock cycles total), the lead out states take 7 clock cycles (240*3*7 = 5040 clock cycles total). Adding these up we get a total of 270,000 cc for milestone 1 to complete. In terms of multiplier usage, calculations are as follows: The common states and lead out states both use 16/21 = 76% utilization while the lead in states use 16/36 = 44.4% utilization. (76% of 262080 common cycles) + (76% of 5040 lead out cycles) + (44.4% of 2880 lead in cycles) = 199180 + 3830 + 1280 = 204290. Since 204290 clock cycles out of the 270000 total are using multipliers, this yields a multiplier usage of 75.6% (204290/270000) meeting the project constraints.

Inverse Discrete Cosine Transform



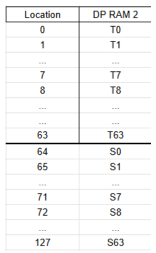
Location	DP RAM 0
0	C0,C8
1	C16, C24
...	
3	C48, C56
4	C1, C9
5	C17, C25
...	...
31	C55, C63
32	CT0, CT8
33	CT16, CT 24
34	CT1, CT9
35	CT17, CT25
...	...
63	CT55, CT63

Figure 2: C and CT matrix layout



Location	DP RAM 1
0	S'0, S'1
1	S'1,S'2
2	S'3, S'4
3	S'5, S'6
4	S'320, S'321
5	S'322, S'323
...	
31	S'2246, S'2247

Figure 3: S' matrix layout for block 1



Location	DP RAM 2
0	T0
1	T1
...	...
7	T7
8	T8
...	...
...	...
63	T63
64	S0
65	S1
...	...
71	S7
72	S8
...	...
127	S63

Figure 4: T and S matrix layout

Following the matrix organization highlighted above, the first matrix multiplication between C, in DPRAM 0, and fetched S' values, in DPRAM 1, is conducted. The matrix multiplication is done by accumulating partial products for each row of T at a time. These computed T values are then put into DPRAM 2 with one value per location in row major format. For the second matrix multiplication between T and C^T, however, calculations are done by accumulating the partial product for each column of S at a time, these S values, however, are put back into the DPRAM in row major format causing there to be jumps of 8 between write locations.

Module	Register name(s)	Bits	Description
M2_Unit	m2_wr_enable,m2_ct_enable, m2_cs_enable,m2_fs_enable, M3_enable, Done	1	Enable signals to start or stop the various submodules as well as the M2 module itself via the done flag.
M2_Unit	FS_base_address, WR_base_address	18	SRAM base address registers that change based on every block that is fetched by FS or every block that is written by WR.
M2_Unit	FS_Jump_Offset, WR_Jump_Offset	9	Jump offsets to manage how many locations to jump by due to DPRAM memory orientation (i.e, if DPRAMs are in row major or column major format.
M2_Unit	FS_row_index, WR_row_index, FS_col_index, WR_row_index	8/18	Keep track of the specific block indices for when 8 by 8 blocks are fetched or are written to SRAM memory.
M2_fs	row_index	4	Keeps track of which row the currently fetched block is at.
M2_fs	DP_counter	7	Address counter for the SP DPRAM which increments everytime data is written to the DPRAM.
M2_fs	jump	18	Rows are a certain value apart (320 for Y) (160 for U/V) this counter increments depending on the row and is used to fetch from the SRAM.
M2_fs	Buff_reg	16	Holds a fetched S value and waits for the next one to be fetched before being written to the SP DRAM as a pair.
M2_fs	wren_SP	1	Enable port for top port of the SP DPRAM.
M2_ct	SP_counter, C_counter, T_address_counter	7	Counters that increment each time the address for SP/C/T DPRAMS are called.
M2_ct	T_accumulator	32	Accumulates the partial products for computing T
M2_ct	SP_buffer, C_buffer	16	3 values are needed, 4 are received from SP and C so one is buffered.
M2_ct	Jump	7	Used to jump locations in the SP DPRAM.
M2_cs	Flag, flag2, flag3, flag4, write_flag, write_flag2	1/3	Enable bits for priority encoder setup in the common case. They are enabled and disabled based on whether or not a value is to be written and if a series of writes are to occur.
M2_cs	S_accumulator[2:0]	32	Accumulators for the three S values that are computed at a time
M2_cs	T_write_counter, T_read_counter, C_addr_counter,	7	Address counters for reading or writing C transpose and T values.
M2_cs	offset, write_offset	6	Offsets to keep track of address jumps due to C transpose and written S values orientation in their respective DPRAMs.
M2_write	T_counter, SRAM_counter	7	Address counters to access SRAM and DPRAM memory.
M2_write	Jump	11	Jump number that keeps track where to write in memory

Latency Analysis: Fetch SP takes ~64 cc, CT takes 197 cc, CS takes 197, WR takes ~32 cc. Since there are 2400 blocks of data to be processed across YUV we can multiply the sum of CT and CS by 2400. FS and WR don't use multipliers, thus the multiplier utilization only involves CT and CS which have 8/9 = 89% utilization. Therefore, to calculate the utilization:

$$\text{Utilization} = [2400*(0.89*197 + 0.89*197)] / [64 + 2400*(197+197) + 32] = 89.6\%$$

Lossless Decoding and Dequantization

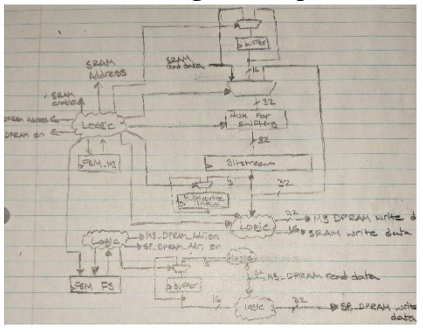


Figure 5: Milestone 3 high level

Lossless decoding is implemented with three 16-bit registers. The first two bitstream registers are constantly being left shifted based on the header code found in the most significant bits of the bitstream registers. The third register stores the next 16 bits of the bitstream and is loaded into the first 2 registers once there are at least 16 bits in the bitstream register unrelated to the bitstream. To determine this, another register is used to track the position of the last bitstream bit in the first 2 bitstream registers (not shown in Fig.5). The M3 FSM determines what data to write to the SRAM and M3 DPRAM by decoding the header, taking the appropriate number of bits from the bitstream register, and dequantizing based on the current write position. In the complete M3 code the SRAM enable is fixed as 1 during milestone 3 to prevent writing to the SRAM. Furthermore, the complete M3 code implements another FSM to fetch the decoded data from the M3 DPRAM to write it to the SP DPRAM 2 values per location. This sub-FSM is a modified version of FS from milestone 2.

Module	Register name(s)	Bits	Description
M3_Unit	quant	1	Stores whether Q0 or Q1 quantization matrix was used
M3_unit	leadInFlag, readFlag, bufferFlag, decodeDone	1	Flags to aid the FSM in determining actions such as how to read a new SRAM address or that decoding is done
M3_unit	SRAM_read_buff	16	3rd bitstream buffer that stores the next 16 bits of the bitstream
M3_unit	Bitstream_buff	32	Stores the incoming bitstream to be decoded and dequantized
M3_unit	dataEnd	5	Stores the last index of the bitstream in bitstream_buff
M3_unit	multiWriteInput	3	Stores the 3 bit number to be written 2nd (header 00) or the number of zeros left to write (header 11)
M3_unit	DPRAM4_buffer	16	Store the decoded M3_DPRAM read value to be combined with another
M3_unit	counters	5-6	Store the address of the DPRAMs and SRAM read and write locations

Latency Analysis: The worst-case scenario for milestone 3 clock cycles is when a new SRAM address is read as often as possible. By only decoding 9-bit values at total of 13 bits must be shifted per value decoded. Since reading SRAM data gains 16 bits of the bitstream, the worst-case scenario would involve 5 SRAM reads for 6 values decoded. Reading from the SRAM (and decoding and writing to memory) takes 3 cc and only decoding a single value takes 1 cc. Assuming all 64 values are 9 bit numbers, a total of ceil(64*5/6)=54 values would take 3 cc. The remaining 10 values would take 1 cc to decode. Overall, the number of cc for decoding 64 values would be 54*3 + 10 = 172 cc. Accounting for the lead in would make it 176 cc.

Resource Usage and Critical Path

According to the compilation report, the resource usage is as follows: 5312 logic elements and 2082 registers. In contrast to Lab 5, these values are much higher which makes sense due to the much more extensive scope of the project. Throughout the project, various unused registers were marked, moreover, many bit widths of various registers, especially counters, could have been reduced to make use of the bit width overflow. Making these changes would have improved our resource usage in terms of logic elements and register usage. According to the timing analyzer the worst-case path starts in the milestone 2 during a state change and ends at the computed T value accumulator in the compute T state. This makes sense because the T accumulator is essentially adding up three 64-bit values which are the partial products of the computed T value and is also shifting them. The starting point of this critical path is any state change into another state that contains the compute T module.

Conclusion

The digital systems design project was a profound learning experience that highlighted the importance of integrating theoretical knowledge with practical applications. Through milestones such as upsampling, inverse discrete cosine transformation, and lossless decoding, we gained valuable insight into hardware implementation, finite state machine design, and efficient use of resources. The hands-on approach with the FPGA board reinforced understanding of image processing and data management. Collaborating on complex problems, analyzing clock cycle utilization, and optimizing system performance were key skills gained through the rigorous work done to complete the project.