3DQ5 – Project Report

Group 80: Kai Hughes hughek26@mcmaster.ca

November 25, 2024

1 Introduction

This project implements a hardware image decompressor capable of decoding a bitstream produced by the mic19 codec. The system reconstructs the final image by performing a sequence of operations, including chroma upsampling, colour-space conversion, IDCT, and lossless coefficient decoding with dequantization. All modules were designed in SystemVerilog and deployed using. Input data is streamed through UART, stored in SRAM, processed through the M1–M3 pipeline, and finally written back as RGB data for VGA display.

2 Implementation

2.1 Milestone 1: Upsampling and Colour-Space Conversion

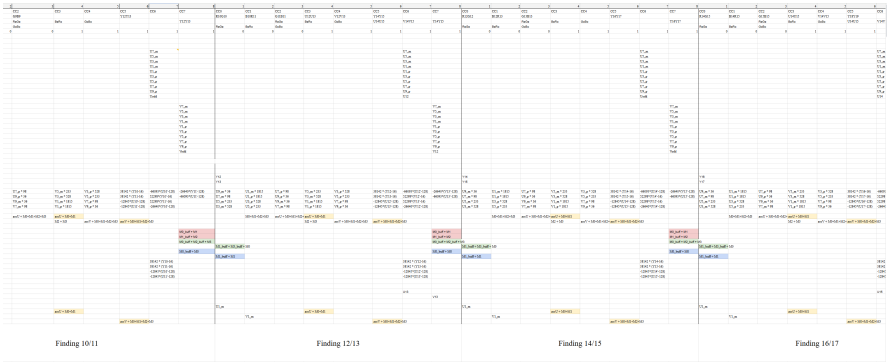

Figure 1: Planning State table. Full exploration available here: Full Spreadsheet.

Selected Register Summary (M1)

Name	Size	Description
M1_S_state	5	FSM state controlling sequencing and pipeline flow.
base_memory_Y,U,V,RGB	14–18	Base addresses for SRAM regions.
memory_Y,U,V,RGB	14–18	Dynamic memory pointers.
memory_offset_Y,UV,RGB	14–17	Offsets added to base pointers.
U/V indexes	$10 \times 8 \times 2$	U/V samples for interpolation.
Uodd, Vodd, Ye, Yo	2×8	Temporary odd-byte and even/odd sample buffers.
U/V_prime_even/odd	2×8	Interpolated chroma values.
M0..M3, M0_buff..M3_buff	4×32	Multiplier operand staging.
accU_prime, accV_prime	2×32	Chroma interpolation accumulators.
Re/o, Ge/o, Be/o	6×32	RGB accumulators.
Re/o_write, Ge/o_write, Be/o_write	6×8	Saturated RGB bytes.
pixel_compute_counter, pixel_column_counter	8	Counters for pixel and row progress.

Table 1: Concise grouped summary of registers used in M1. Size in bits.

M1 Latency Analysis

1. Common-case multiplier utilization:
$$\frac{4 \times 8 - 2}{4 \times 8} = 93.75\%$$
2. General utilization:
$$\frac{192 \times 8 \times 0.9375 - (4 \times 9 + 4 \times 3)}{192 \times 8 + (4 \times 9 + 4 \times 3)} = 87.88\%$$

2.2 Milestone 2: Inverse Discrete Cosine Transform

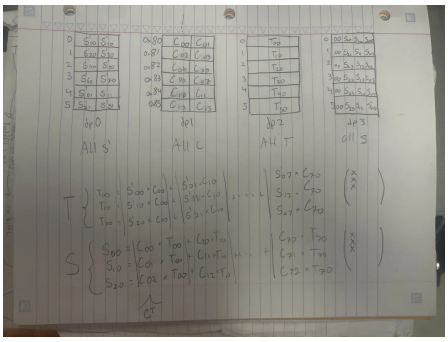

Figure 2: M2 multiplier pipeline and DP-RAM buffer organization.

Selected Register Summary (M2)

Name	Size	Description
M2_S_state	4	Top-level FSM state.
Y_finished_W, U_finished_W	1	Channel completion flags.
leadout_ws_done_waiting_m3	1	Synchronization flag between WriteS and M3.
m3/T/S/WS_done_latched	1	Done signal capture latches (4 total).
m3/T/S/WS_start	1	Module start pulses (4 total).
Rblock, Cblock, Rblock_WS, Cblock_WS	5	Block row/column indices for compute and write.
ComputeT/ComputeS Submodules		
state (CT/CS)	5	ComputeT and ComputeS FSM states.
Addressing_counter	5	Iteration through k (0–15 or 0–7).
column_counter	4	Column index j (0–15 or 0–7).
row_set_counter	3	Row set index (groups of 3 rows).
Acc_T0/T1/T2, Acc_S0/S1/S2	32	Matrix multiply-accumulate registers (6 total).
mult_op1_a/b/c, mult_op2_a/b/c	32	Shared multiplier operands (6 total per module).
write_data_buff (CT)	32	Buffered T result for delayed write.
T/S_DP_RAM_address_a/b	8	RAM address outputs (2–3 per module).
T/S_DP_RAM_write_data_a/b	32	RAM write data outputs (1–2 per module).
Address_C_a/b, Address_T_a, Address_Sp_a/b	8	C/T/Sp RAM address lines (3–5 per module).
Write_en_T_a/b, Write_en_S_a	1	RAM write enable signals.
first_cycle, done	1	Initialization and completion flags.
WriteS Submodule		
WRITE (state)	4	WriteS FSM state.
ri, ci, col_pair	4/3/8	Row, column, and pair counters.
A_data, B_data	32	Column data buffers for packing.
BASE	18	Base address offset (Y/U/V selection).
IDCT_address	18	Computed SRAM write address.
RA, CA	12	Absolute row and column addresses.
Address_Sp_a/b	8	Sp RAM read address outputs (2 total).
done	1	WriteS completion flag.
FetchSP Submodule		
fetch_sp (state)	4	FetchSP FSM state.
ri, ci	4	Intra-block row/column indices.
Se	16	Temporary even-row sample buffer.
DP_RAM_address, DP_RAM_write_data	8/32	DP-RAM write interface.

Table 2: Selected M2 register summary.

M2 Latency Analysis

Y (108 blocks)

Lead-in: 1897

Steady-state: $106 \times 1633 = 173{,}098$

Lead-out: 1783

Subtotal: 176,778

U+V (432 blocks)

Lead-in: 289

Steady-state: $430 \times 217 = 93{,}310$

Lead-out: 259

Subtotal: 93,858

Total: 270,636 cycles = 5.41 ms @ 50 MHz

Total multiplications: 1,465,344

Total available slots: 270,636 cycles × 6 mults = 1,623,816

Average utilization: 90.2%

Module Utilization Summary

Module	Y blocks	UV blocks
ComputeT	94.1%	78.6%
ComputeS	94.1%	88.5%
Overall M2	90.2%	

2.3 Milestone 3: Lossless Decoding and Dequantization

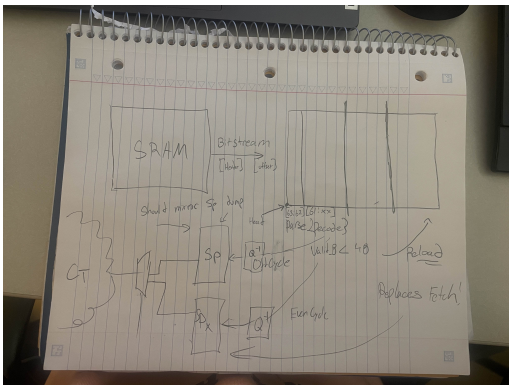

Figure 3: M3 planning and data flow.

Selected Register Summary (M3)

Name	Size	Description
m3_state	3	M3 FSM state (header/decode/done).
finish, decoder_start	1	Module completion and decoder control.
header_read, Q_number	1	Header flag and quantization table selector.
write_toggle	1	Ping-pong buffer selector (Sp/X RAMs).
zz_ri, zz_ci, zz_dir	4/4/1	Zig-zag traversal position and direction.
m3_sram_address	18	SRAM address for header reads.
RAM port addresses	8	address_a/b_Sp, address_a/b_X [7:0 each].
RAM write data	32	write_data_a/b_Sp, write_data_a/b_X [31:0 each].
RAM control signals	1/4	write_enable and byteena for dual ports.
decode_state, reload_state	4/3	Main decoder and prefetch FSMs.
bitstream_buffer	64	Main 64-bit bitstream buffer.
prefetch_shift_reg	64	Secondary 64-bit prefetch buffer.
bits_valid, prefetch_valid_bits	7	Valid bit counters for each buffer.
bits_transferred	7	Communication between buffers.
sram_offset	18	Prefetch SRAM offset pointer.
lead_in_sram_address	18	Lead-in phase address.
lead_in_counter, lead_in_active	3/1	Lead-in sequence control.
bits_to_consume, shift_amount	4	Bit consumption tracking.
decoded_coeff_internal	9	Internal decoded coefficient.
coeff_position, block_size	9	Position counter and block size.
zeros_to_write, zeros_written	9	Zero-run-length tracking.
decoded_coefficient	9	Output decoded value.
decoded_valid, finish	1	Output valid flag and completion.
bits_consumed_flag	1	Consumption tracking flag.

Name	Size	Description
temp_buffer, insert_pos	64/6	Working registers for buffer updates.
bits_consumed/added_this_cycle	7	Net bit change tracking.

Table 3: Selected M3 register summary. Size in bits.

Latency Analysis (M3)
Worst case occurs when all coefficients use 11-bit symbols. 3 cycles per element:

$$3 \times 256 = 768 \text{ cycles per luma block}, 3 \times 64 = 192 \text{ cycles per chroma block}$$

My design takes two cycles for decoding and writing. It appears as parsing, decoding, and writing but because the writing is combinational, it will appear immediately after it is loaded by the decoder, and as M3's top level FSM is processing it - the parsing/decoding state begins for the next one. So including the processing time for loading the bitstream, and the lead out, the total clock cycles for chroma (the upper bound threshold), would be 135 cycles. That is - 6 for loading the stream, 1 lead out, and 128 processing cycles. This is under the required 192 absolute worst case.

3 Weekly Activity and Progress

Week	Tasks Completed
0	Developed the initial state table and explored interpolation architectures for M1.
1	Finalized the state table, refined lead-in/lead-out behavior, and created the initial M1 skeleton. Then began filling in the M1 states as per the state table.
2	Completed M1 and debugged using V0. Transitioned to M2 after reviewing memory partitioning with the Graham.
3	Began with FetchS' module and laid out the top level FSM (M2) and tried to write everything serially to begin with (figured debugging would be easier as enabling and disabling states would be a couple lines at most).
4	Simulated all M2 modules. Debugged matrix multiplications in MATLAB (just copied over memory dump and manually recreated) and redesigned the memory layout after realizing C transpose was not feasible. Completed M2 verification for some modules.
5	Finalized M2 verification and planned for M3 implemented requantizer and zig-zag counter for M3 and planned the full M3 pipeline. Finalized on the weekend/monday. Tested mainly on Sunday/Monday with worst cases.

4 Resource Usage and Critical Path

The final project consumes 5145 logic elements with 1,861 registers, representing a substantial increase from Lab 5's baseline VGA display with around 30. The M1 top-level module uses 1,742 logic elements (672 registers), accounting for 33% of total project usage. M1's high resource consumption stems from implementing the complete YUV-to-RGB color space conversion pipeline with extensive buffering and computation logic. It maintains 10 U/V buffer registers for upsampling interpolation, plus accumulators, Y/U/V prime values, and six RGB output registers. M1 also instantiates four 32-bit multipliers and contains an 8-state FSM managing pixel-by-pixel processing across 192x144 resolution. In the future, I would focus on not using new registers for flags especially as most can be derived from other counters. Specifically for M1 the even_cycle flag was unnecessary (could have indexed pixel counter). Also, during development I did realized how I could have change those dynamic memory elements because all I needed was the base and offset. This strategy was used for M2 and M3.

Even more so, the most taxing module was the M2 DCT computation which accounts for 2,692 logic elements (819 registers), with registers distributed across ComputeT and ComputeS submodules plus M3 and WriteS for SRAM interfacing. This was after integration so of course this would not take into account the FetchSp module, but rather the M3/Decoding modules.

The critical path of the system is dominated by the M2 `ComputeT` accumulation stage, specifically between the register nodes storing intermediate S' values and the final `ComputeT` outputs. This is expected because the accumulation stage performs multiple sequential multiplier additions that cannot be fully parallelized due to data dependencies between successive rows and columns. The measured slack is 11.57 ns, with a total path delay of 8.17 ns. This serves well under our 20 ns budget.

5 Conclusion

The final system successfully implements real-time decompression of mic19-encoded images using a structured hardware pipeline. Each milestone builds toward a complete decoder capable of running within FPGA timing constraints while efficiently using available DSPs, RAM blocks, and logic resources. The learning experience in exploring different implentations and being able to physically implement them has been unmatched.

6 Reference

Nicolici, N. (2025). Digital systems design course: Hardware implementation of an image decompressor.