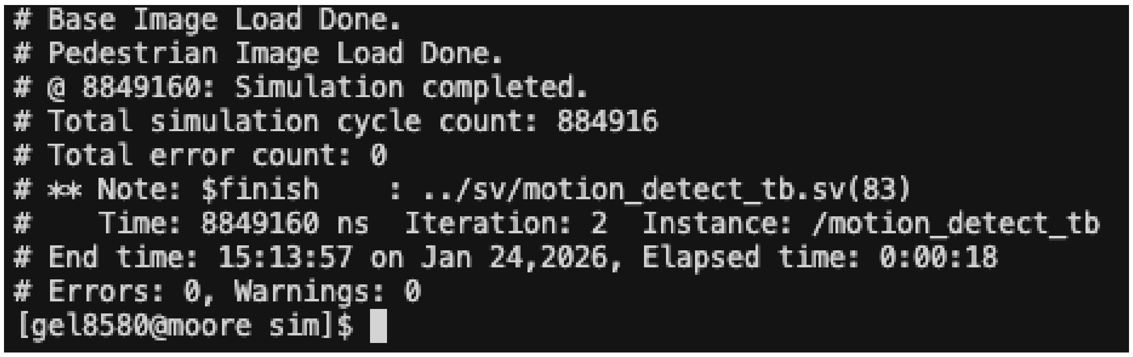
1 Simulation results

1.1 Clock cycle count

Answer:

The simulation results demonstrating the system performance and functional correctness are shown in Fig. 1 and Fig. 2. s



```
# Base Image Load Done.
# Pedestrian Image Load Done.
# @ 8849160: Simulation completed.
# Total simulation cycle count: 884916
# Total error count: 0
# ** Note: $finish    : ../sv/motion_detect_tb.sv(83)
#    Time: 8849160 ns  Iteration: 2  Instance: /motion_detect_tb
# End time: 15:13:57 on Jan 24,2026, Elapsed time: 0:00:18
# Errors: 0, Warnings: 0
[gel8580@moore sim]$
```

Figure 1: Simulation console output showing zero errors and total cycle count.

Fig. 1 shows the final simulation output. The simulation was configured with a clock period of 10 ns (100 MHz) as defined in the testbench parameter `CLOCK_PERIOD`. The design processed the entire image in **884,916 cycles**. Based on the image resolution of 768×576 pixels (total $442,368$ pixels), the performance metric is calculated as:

$$\text{Cycles per Pixel} = \frac{884,916}{442,368} \approx 2.00 \tag{1}$$

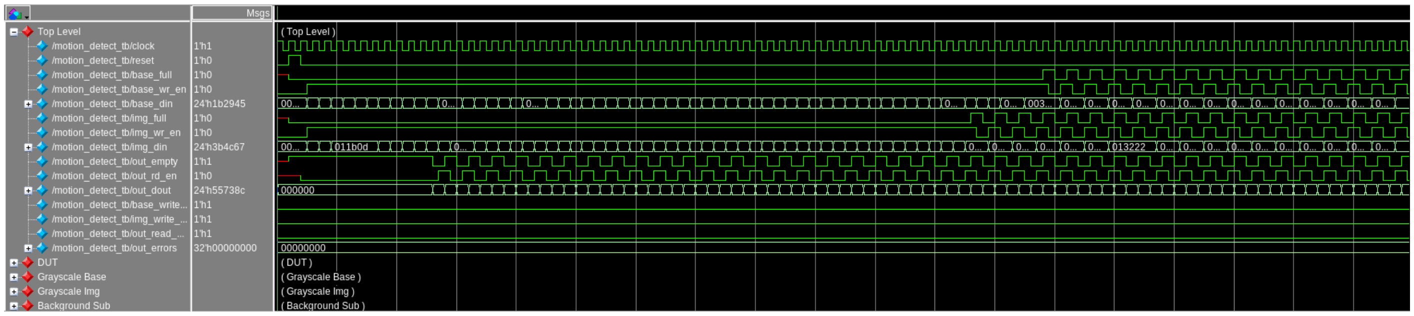
This result matches the theoretical performance of our 2-process FSM architecture. Each processing module (Grayscale, Background Subtraction, Highlight) requires exactly two cycles per pixel: one cycle for reading/computation and one cycle for writing to the output FIFO. The fact that the ratio is almost exactly 2.0 indicates that the pipeline overhead is negligible compared to the large dataset, confirming that the design achieves an algorithmic complexity of $O(N)$.

Fig. 2 presents the simulation waveforms. Fig. 2a captures the initialization phase where modules are activated sequentially as data propagates through the FIFOs.

Fig. 2b illustrates the essence of the streaming architecture: **Full Pipeline Utilization**. As observed in the waveform, both the upstream module (`Background Sub`) and the downstream module (`Highlight`) are toggling and processing data simultaneously. This confirms that:

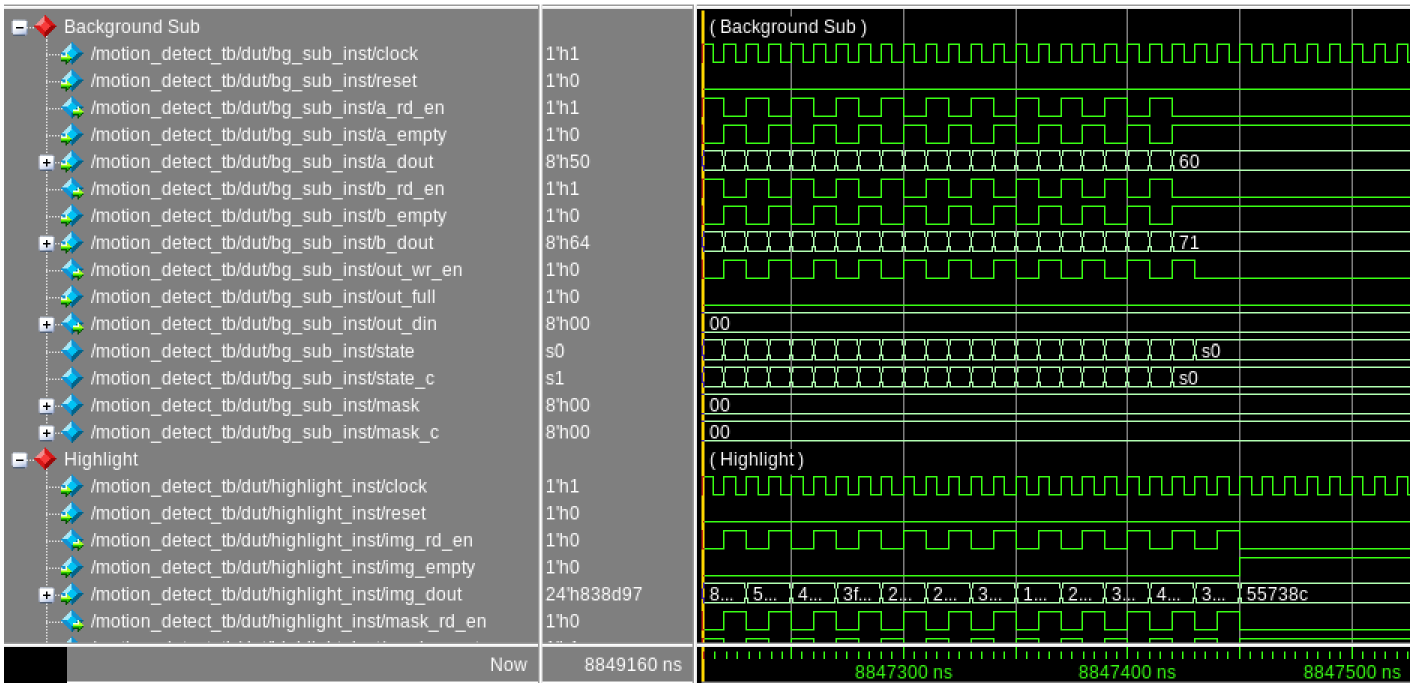
- When the `Grayscale` module processes the N-th pixel, `Background Sub` is processing the $(N-1)$-th pixel, and `Highlight` is processing the $(N-2)$-th pixel.
- The FIFO depth is sufficient to absorb any jitter, preventing pipeline stalls.

This concurrent operation without stalling is the key reason why the system achieves the optimal throughput of 2 cycles per pixel.



```
                                      Msgs
Top Level
  /motion_detect_tb/clock             1'h1
  /motion_detect_tb/reset             1'h0
  /motion_detect_tb/base_full         1'h0
  /motion_detect_tb/base_wr_en        1'h0
  /motion_detect_tb/base_din          24'h1b2945
  /motion_detect_tb/img_full          1'h0
  /motion_detect_tb/img_wr_en         1'h0
  /motion_detect_tb/img_din           24'h3b4c67
  /motion_detect_tb/out_empty         1'h1
  /motion_detect_tb/out_rd_en         1'h0
  /motion_detect_tb/out_dout          24'h55738c
  /motion_detect_tb/base_write...     1'h1
  /motion_detect_tb/img_write_...     1'h1
  /motion_detect_tb/out_read_...      1'h1
  /motion_detect_tb/out_errors        32'h00000000
DUT
Grayscale Base
Grayscale Img
Background Sub
```

(a) Simulation Start: Pipeline filling phase.



```
Background Sub
  /motion_detect_tb/dut/bg_sub_inst/clock             1'h1
  /motion_detect_tb/dut/bg_sub_inst/reset             1'h0
  /motion_detect_tb/dut/bg_sub_inst/a_rd_en           1'h1
  /motion_detect_tb/dut/bg_sub_inst/a_empty           1'h0
  /motion_detect_tb/dut/bg_sub_inst/a_dout            8'h50
  /motion_detect_tb/dut/bg_sub_inst/b_rd_en           1'h1
  /motion_detect_tb/dut/bg_sub_inst/b_empty           1'h0
  /motion_detect_tb/dut/bg_sub_inst/b_dout            8'h64
  /motion_detect_tb/dut/bg_sub_inst/out_wr_en         1'h0
  /motion_detect_tb/dut/bg_sub_inst/out_full          1'h0
  /motion_detect_tb/dut/bg_sub_inst/out_din           8'h00
  /motion_detect_tb/dut/bg_sub_inst/state             s0
  /motion_detect_tb/dut/bg_sub_inst/state_c           s1
  /motion_detect_tb/dut/bg_sub_inst/mask              8'h00
  /motion_detect_tb/dut/bg_sub_inst/mask_c            8'h00
Highlight
  /motion_detect_tb/dut/highlight_inst/clock          1'h1
  /motion_detect_tb/dut/highlight_inst/reset          1'h0
  /motion_detect_tb/dut/highlight_inst/img_rd_en      1'h0
  /motion_detect_tb/dut/highlight_inst/img_empty      1'h0
  /motion_detect_tb/dut/highlight_inst/img_dout       24'h838d97
  /motion_detect_tb/dut/highlight_inst/mask_rd_en     1'h0
                                             Now      8849160 ns
```

(b) Steady State: Concurrent operation of Background Sub and Highlight modules.

Figure 2: Simulation waveforms demonstrating the streaming architecture. (a) shows the sequential activation of modules. (b) demonstrates full pipeline utilization where multiple stages process different pixels simultaneously.

1.2 Errors reported

Answer:

The simulation reported **0 errors**. As demonstrated in the simulation console output (Fig. 1), the error counter `out_errors` remained at zero throughout the verification phase. This confirms that the hardware implementation is bit-true accurate compared to the golden reference (`img_out.bmp`).

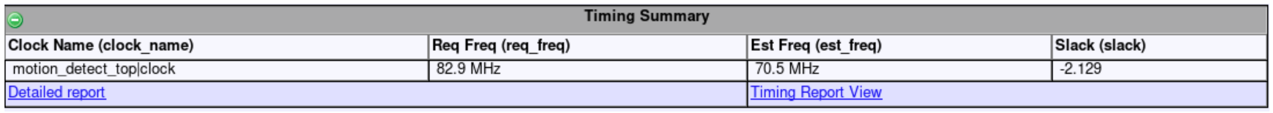
2 Synthesis results

2.1 Maximum frequency

Answer:

The estimated maximum frequency of the design is **70.5 MHz**.

As shown in Fig. 3, the synthesis tool reports an estimated frequency of 70.5 MHz against a requested frequency of 82.9 MHz, resulting in a negative slack of -2.129 ns.



Timing Summary			
Clock Name (clock_name)	Req Freq (req_freq)	Est Freq (est_freq)	Slack (slack)
motion_detect_top|clock	82.9 MHz	70.5 MHz	-2.129
Detailed report		Timing Report View	

Figure 3: Timing summary synthesis report. The estimated frequency is 70.5 MHz.

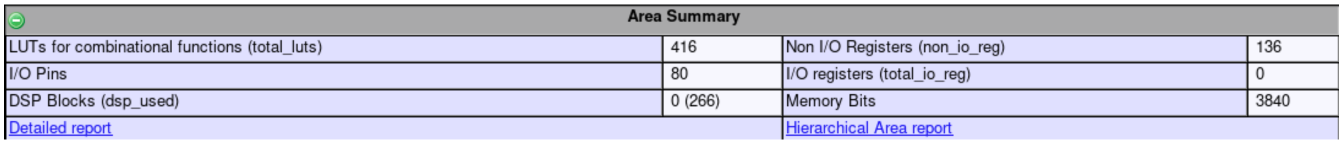
2.2 Registers/LUTs/Logic Elements

Answer:

The resource utilization for the Motion Detection System on the Cyclone IV-E FPGA is summarized below, based on the synthesis area report shown in Figure 4:

1. Total Registers (Non I/O): 136
2. Total Combinational Functions (LUTs): 416
3. Logic Elements: 416 (Based on total LUT utilization)

These resources are utilized to implement the control logic for the 2-process FSMs across four modules, the arithmetic units for pixel processing, and the pointers/control logic for the five internal FIFOs.



Area Summary			
LUTs for combinational functions (total_luts)	416	Non I/O Registers (non_io_reg)	136
I/O Pins	80	I/O registers (total_io_reg)	0
DSP Blocks (dsp_used)	0 (266)	Memory Bits	3840
Detailed report		Hierarchical Area report	

Figure 4: Area summary report detailing the resource usage, showing 416 LUTs and 136 registers.

2.3 Memory utilization

Answer:

The design utilizes **3,840 bits** of memory, as indicated in the "Memory Bits" field of Figure 4. This memory usage corresponds to the storage requirements of the internal FIFOs used for buffering data between pipeline stages. The design instantiates multiple FIFOs (e.g., input buffers, intermediate grayscale buffers, and mask buffer), each configured with a depth of 32 words to handle the latency differences between processing stages.

2.4 Multipliers (DSPs)

Answer:

The DSP block utilization is **0**, as shown in Figure 4. This is expected because the Motion Detection algorithm relies primarily on addition, subtraction, absolute difference, and division (implemented via logic or approximation), rather than intensive multiplication. Unlike matrix multiplication or convolution tasks, the pixel-wise operations in this design do not require dedicated hardware multipliers (DSP slices).

2.5 Worst path(timing analysis)

Answer:

1. **Critical Path Slack:** -2.129 ns (at 82.9 MHz target)
2. **Path Delay:** 14.719 ns
3. **Start Point:** `fifo_base_in.fifo_buf / q_b[16]` (Input FIFO Block RAM output)
4. **End Point:** `gs_base_inst.gs[0] / d` (Grayscale module register input)
5. **Logic Levels:** 24

The critical path analysis identifies the arithmetic logic within the Grayscale module as the primary timing bottleneck. As detailed in the synthesis report, the path originates from the read port of the Base Image Input FIFO (Fig. 6a), propagates through a deep combinational logic cloud, and terminates at the internal register of the Grayscale module (Fig. 6b).

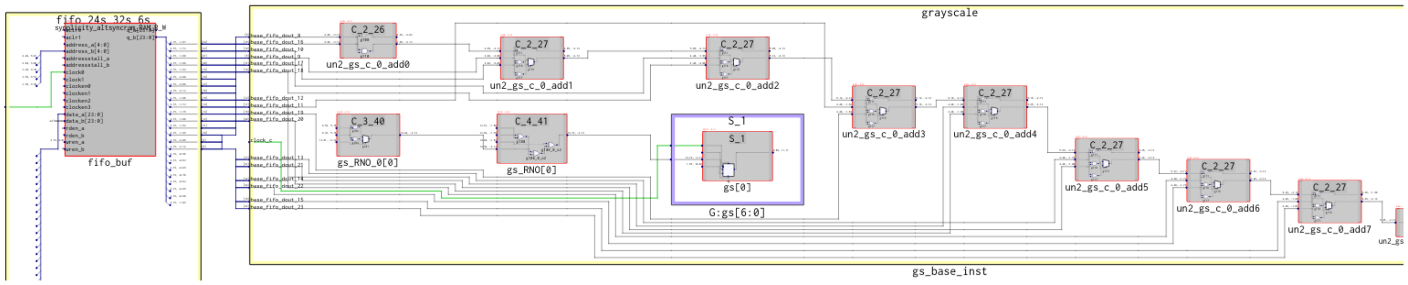
This path, consisting of 24 logic levels, implements the pixel averaging operation: $Gray = (R + G + B)/3$. Figure 5a illustrates the full path from the FIFO memory (left) through the adder tree to the destination register. Figure 5b provides a close-up of the endpoint register (`S_1`), which captures the result. The requirement to complete this arithmetic operation within a single clock cycle limits the maximum frequency to 70.5 MHz.

Figure 6 confirms this is a systematic datapath limitation. The uniform negative slack across multiple bits indicates that the entire RGB-to-Grayscale conversion stage is the critical constraint.

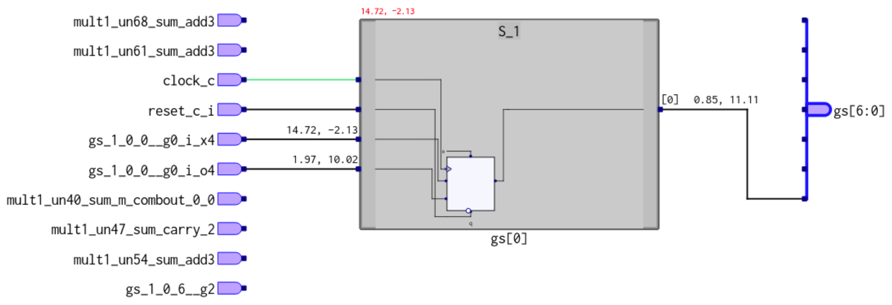
2.6 Schematic architecture(RTL)

Answer:

The RTL architecture implements a streaming data processing pipeline. Unlike the previous matrix multiplication design which relied on a centralized controller, this design adopts a distributed control scheme where each module operates independently, synchronized via FIFO buffers.

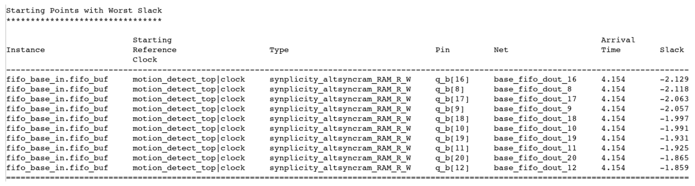


(a) RTL Schematic of the worst timing path: FIFO (left) to Register (right).



(b) Close-up of the path endpoint register (`S_1`).

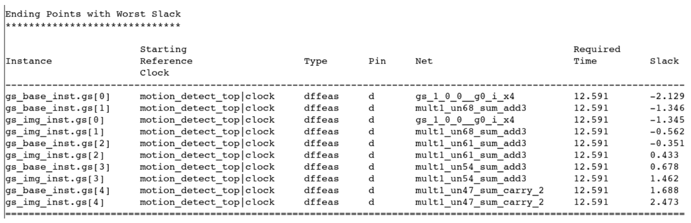
Figure 5: Visual analysis of the critical path showing the combinational delay through the arithmetic logic.



```
Starting Points with Worst Slack
********************************

                          Starting                                                                              Arrival
Instance                  Reference              Type                            Pin         Net                   Time         Slack
                          Clock
-----------------------------------------------------------------------------------------------------------------------------------------
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[16]     base_fifo_dout_16     4.154     -2.129
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[8]      base_fifo_dout_8      4.154     -2.118
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[17]     base_fifo_dout_17     4.154     -2.063
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[9]      base_fifo_dout_9      4.154     -2.057
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[18]     base_fifo_dout_18     4.154     -1.997
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[10]     base_fifo_dout_10     4.154     -1.991
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[19]     base_fifo_dout_19     4.154     -1.931
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[11]     base_fifo_dout_11     4.154     -1.925
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[20]     base_fifo_dout_20     4.154     -1.865
fifo_base_in.fifo_buf     motion_detect_top|clock     synplicity_altsyncram_RAM_R_W     q_b[12]     base_fifo_dout_12     4.154     -1.859
=========================================================================================================================================
```

(a) Starting points (FIFO outputs).



```
Ending Points with Worst Slack
******************************

                        Starting                                                                 Required
Instance                Reference                 Type       Pin     Net                         Time         Slack
                        Clock
---------------------------------------------------------------------------------------------------------------------
gs_base_inst.gs[0]      motion_detect_top|clock     dffeas     d       gs_1_0_0__g0_i_x4           12.591       -2.129
gs_base_inst.gs[1]      motion_detect_top|clock     dffeas     d       mult1_un68_sum_add3         12.591       -1.346
gs_img_inst.gs[0]       motion_detect_top|clock     dffeas     d       gs_1_0_0__g0_i_x4           12.591       -1.345
gs_img_inst.gs[1]       motion_detect_top|clock     dffeas     d       mult1_un68_sum_add3         12.591       -0.562
gs_base_inst.gs[2]      motion_detect_top|clock     dffeas     d       mult1_un61_sum_add3         12.591       -0.351
gs_img_inst.gs[2]       motion_detect_top|clock     dffeas     d       mult1_un61_sum_add3         12.591       0.433
gs_base_inst.gs[3]      motion_detect_top|clock     dffeas     d       mult1_un54_sum_add3         12.591       0.678
gs_img_inst.gs[3]       motion_detect_top|clock     dffeas     d       mult1_un54_sum_add3         12.591       1.462
gs_base_inst.gs[4]      motion_detect_top|clock     dffeas     d       mult1_un47_sum_carry_2      12.591       1.688
gs_img_inst.gs[4]       motion_detect_top|clock     dffeas     d       mult1_un47_sum_carry_2      12.591       2.473
=====================================================================================================================
```

(b) Ending points (Grayscale registers).

Figure 6: Synthesis timing report showing uniform slack violations, confirming the bottleneck.

The architecture consists of the following key components:

a. **Top-Level Streaming Pipeline**: A cascade of processing modules connected by FIFOs to handle data flow and clock domain crossing.

b. **Distributed FSMs**: Each processing module contains a compact 2-state Finite State Machine (optimized to a single-bit register) to manage the Read-Modify-Write cycles.

c. **Grayscale Units**: Datapaths for calculating pixel luminance: $Gray = (R + G + B)/3$.

d. **Background Subtraction Unit**: Logic for computing absolute difference $|Img - Base|$ and thresholding.

e. **Highlight Unit**: Multiplexing logic to overlay the detection mask onto the original image.

Figure 7 illustrates the system level integration. The distinct yellow blocks are the FIFOs that decouple the processing stages, allowing for the "Full Pipeline Utilization" observed in the simulation waveforms. While the logical data flow is sequential (Input → Grayscale → Background Sub → Highlight), the schematic optimizes for connectivity, positioning the `highlight_inst` (left) and `background_sub_inst` (right) around the central `Grayscale` modules.

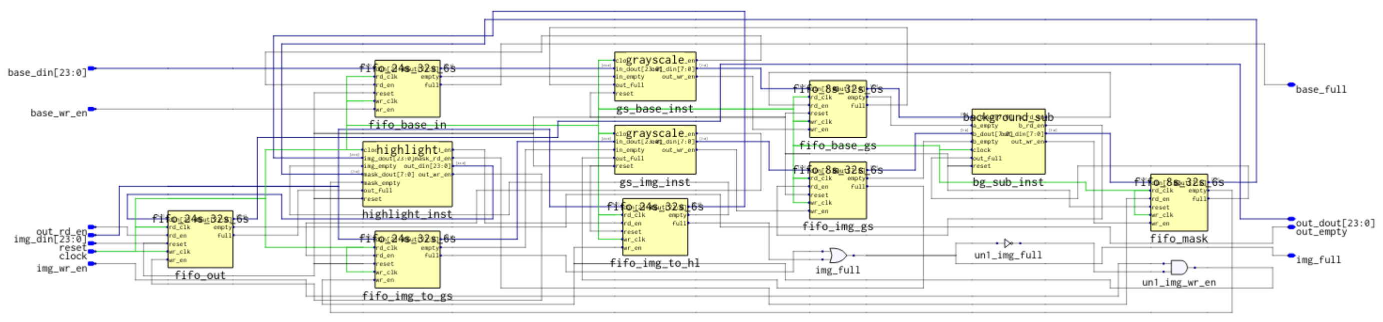


Figure 7: Top-level RTL schematic of `motion_detect_top`.

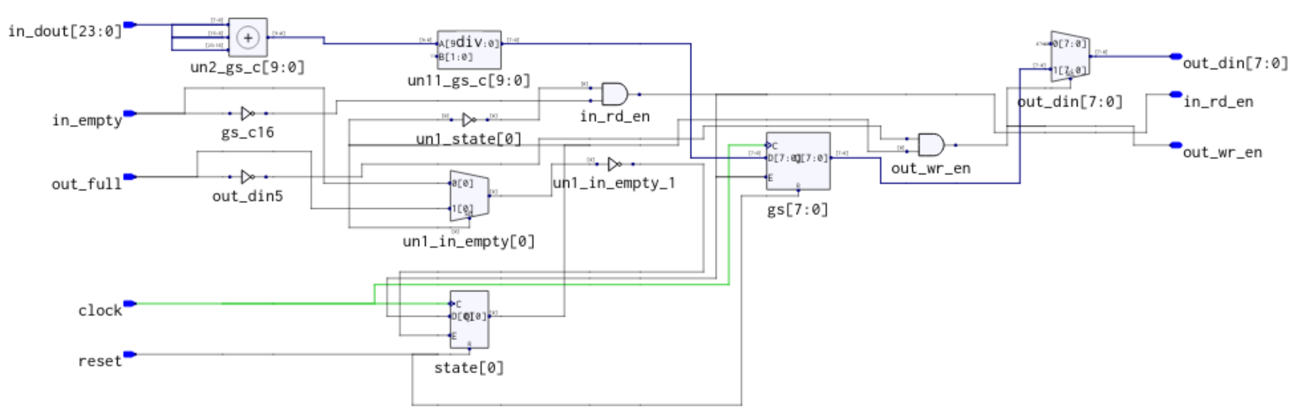


Figure 8: RTL schematic of the `grayscale` module. The adder chain at the top computes the sum of RGB channels. The `state[0]` register (bottom center) implements the optimized 2-state FSM that controls the read/write enable signals.

Figure 8 reveals the internal datapath of the Grayscale module. The synthesis tool has inferred an adder tree (top left) to sum the color channels. Notably, the FSM is implemented as a single flip-flop `state[0]`, which toggles between the Read/Compute state and the Write state.

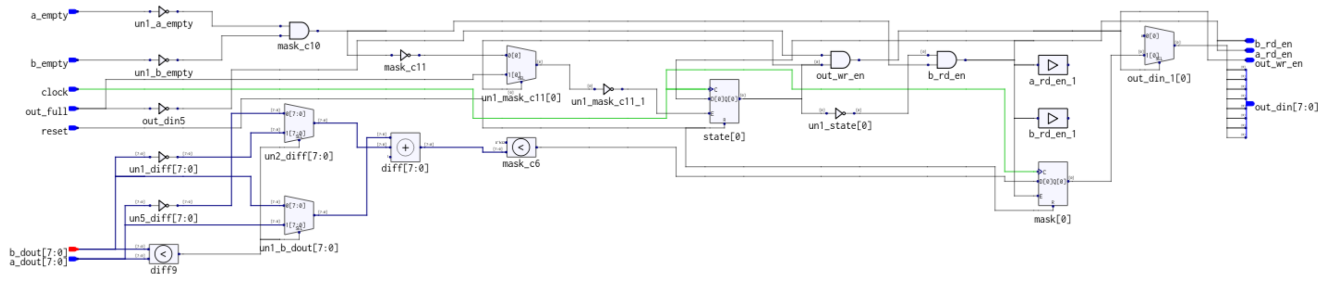


Figure 9: RTL schematic of the **background_sub** module. The logic on the left implements the absolute difference calculation ($|A - B|$) and the threshold comparison used to generate the binary mask.

The Background Subtraction logic is shown in Figure 9. The comparators and subtractors on the left side are responsible for calculating the absolute difference between the base and current frames.

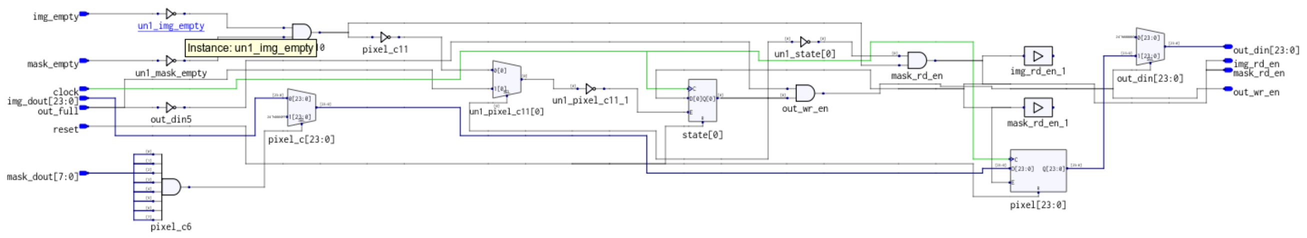


Figure 10: RTL schematic of the **highlight** module. The multiplexer logic selects between the original pixel color and the red highlight color based on the input mask signal.

Finally, Figure 10 depicts the Highlight module. It primarily consists of multiplexing logic that uses the binary mask generated by the previous stage to selectively replace pixels with a solid red color, completing the motion detection pipeline.