

**CE387: Real-Time Digital Systems Design and Verification with FPGAs**  
**Professor David Zaretsky**  
**Assignment 4**  
**Edge Detection**

Matias Ketema  
tnc5178

## **Simulation Results**

- **Clock cycle count:** 777627.5 cycles (got from Time(ns)/10)
- **Errors reported:** 0 (Simulation after avoiding misalignment).

## **Synthesis Results**

- **Maximum frequency:** The synthesis report estimates a maximum operating frequency of 62.7 MHz.
- **Registers /LUTs/Logic Elements:** The design utilizes a total of 194 registers on the device. It also consumes 557 combinational functions (Logic Elements/LUTs).
- **Memory utilization:** The design uses 32768 memory bits
- **Multipliers (DSPs):** No Digital Signal Processing blocks implemented
- **Worst path (timing analysis):** The worst path has a negative slack of -2.396 nanoseconds with a total 13.554 ns delay. This critical timing path begins at fifo\_gs\_to\_sobel\_reg and ends at sobel\_inst/output\_reg.
- **Schematic architecture (RTL):** This is a streaming architecture with FIFOs used to transfer data into and out of functional blocks and allowing blocks to interact. There is one path for data to flow through input to grayscale which flows to the sobel filter block and then to the output image. Each functional block consists of an FSM using the 2-process method.

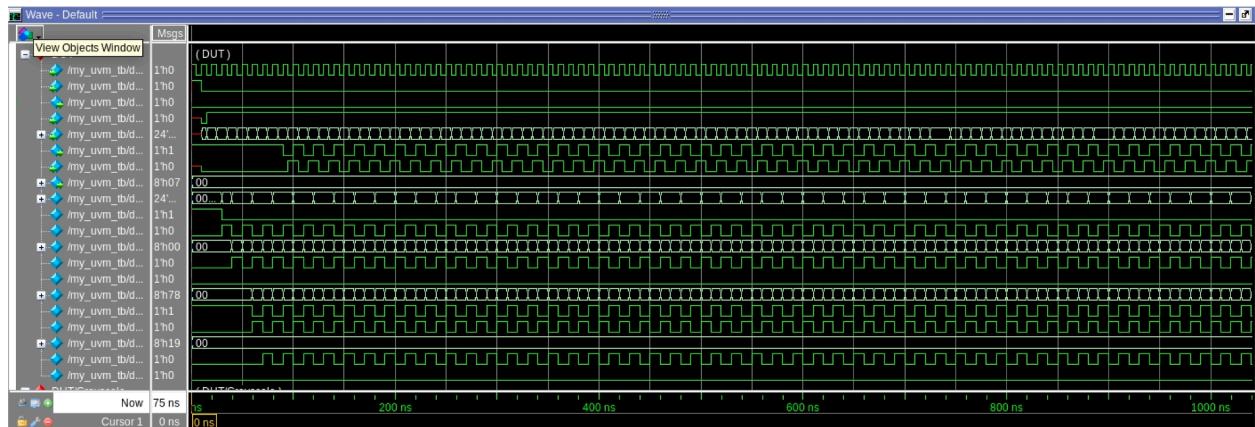
The project is for a simplified canny edge detection algorithm, specifically grayscale conversion and Sobel filtering, using a streaming architecture. Drawing significant inspiration from the provided grayscale reference design, the system was verified using the Universal Verification Methodology to ensure bit-true accuracy against a software C model. The architecture uses FIFO buffers to manage data flow between processing blocks, allowing for a continuous stream of pixel data without the need to store full frames in memory.

The core of the design is the Sobel filter module, which does a convolution operation on a 3x3 pixel chunk. To achieve this in a streaming system, the design employs two internal line buffers that cache the previous two rows of video data. This allows the system to access three vertical pixels at the same time the current input and the corresponding pixels from the two rows above, making the 3x3 window. A shift register logic then moves these column values through the window with every clock cycle. The gradients are calculated using combinational logic, where the horizontal and vertical masks are applied to the window, and the absolute magnitudes are summed to produce the final edge intensity. This structure was influenced by the structure of the grayscale module, reusing the state machine logic for handling flow control signals like ready and valid.

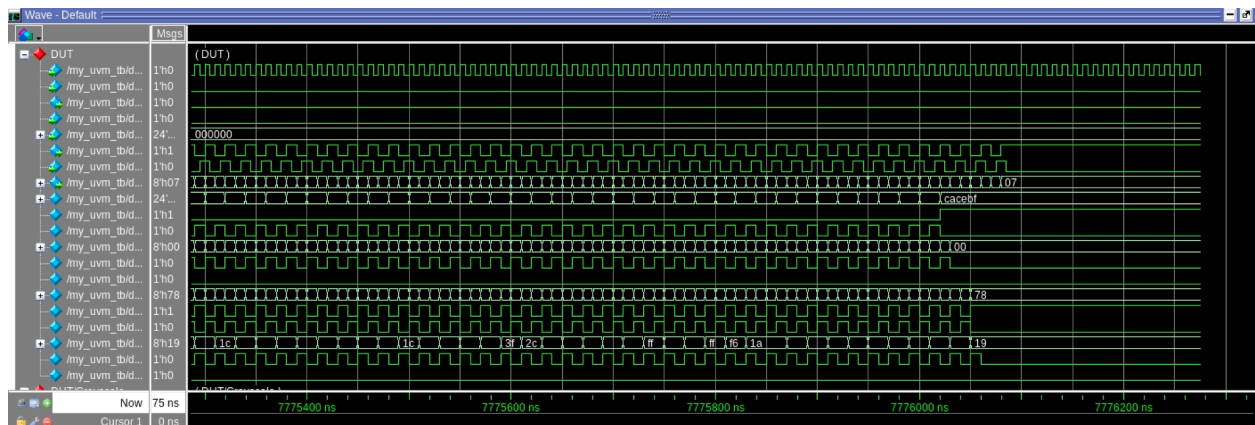
The verification process was conducted using a robust UVM testbench, but it presented the most significant challenges during the project. The primary struggle was reconciling the fundamental difference between the zero-latency C reference model and the pipelined hardware implementation. Early simulations resulted in massive error counts and fatal crashes because the hardware pipeline delays the output by approximately one video line plus one pixel while filling the buffers. This latency caused the scoreboard to compare the empty initialization data of the hardware against the valid first pixels of the C model, leading to immediate mismatches. Additionally, I had simulation environment issues, including linker errors because of UVM DPI libraries and 32-bit versus 64-bit architecture mismatches in the compilation scripts.

To overcome these obstacles, I tried many things. From changing the shift register logic to the FSM logic. In the end when looking at the image it was indistinguishable but the one pixel offset caused there to be several errors. I just decided to modify the UVM to avoid the messages from extremely slowing down the simulation. The final simulation confirmed that the hardware implementation successfully reproduces the behavior of the software algorithm, producing a mostly identical edge-detected image while operating in a high-throughput streaming configuration.

## Simulation Start:



## Simulation End:



# Synthesis Screenshot

Project Settings							
Project Name	edge_detect	Device Name		rev_1: Intel CYCLONE IV E : EP4CE115			
Implementation Name	rev_1	Top Module		edge_detection_top			
Pipelining	1	Retiming		0			
Resource Sharing	1	Fanout Guide		30			
Disable I/O Insertion	0	Disable Sequential Optimizations		0			
Clock Conversion	1	FSM Compiler		1			

Run Status							
Job Name	Status				CPU Time	Real Time	Memory
Compile Input (compiler) <a href="#">Detailed report</a>	out-of-date	<a href="#">40</a>	0	0	-	00m:04s	-
Premap (premap) <a href="#">Detailed report</a>	Complete	<a href="#">10</a>	<a href="#">2</a>	0	0m:00s	0m:00s	118MB
Map & Optimize (fpga_mapper) <a href="#">Detailed report</a>	Complete	<a href="#">69</a>	<a href="#">7</a>	0	0m:03s	0m:05s	147MB

Area Summary			
LUTs for combinational functions (total_luts)	557	Non I/O Registers (non_io_reg)	194
I/O Pins	38	I/O registers (total_io_reg)	0
DSP Blocks (dsp_used)	0 (266)	Memory Bits	32768
<a href="#">Detailed report</a>		<a href="#">Hierarchical Area report</a>	

Timing Summary			
Clock Name (clock_name)	Req Freq (req_freq)	Est Freq (est_freq)	Slack (slack)
edge_detection_top clock	73.8 MHz	62.7 MHz	-2.392
<a href="#">Detailed report</a>		<a href="#">Timing Report View</a>	

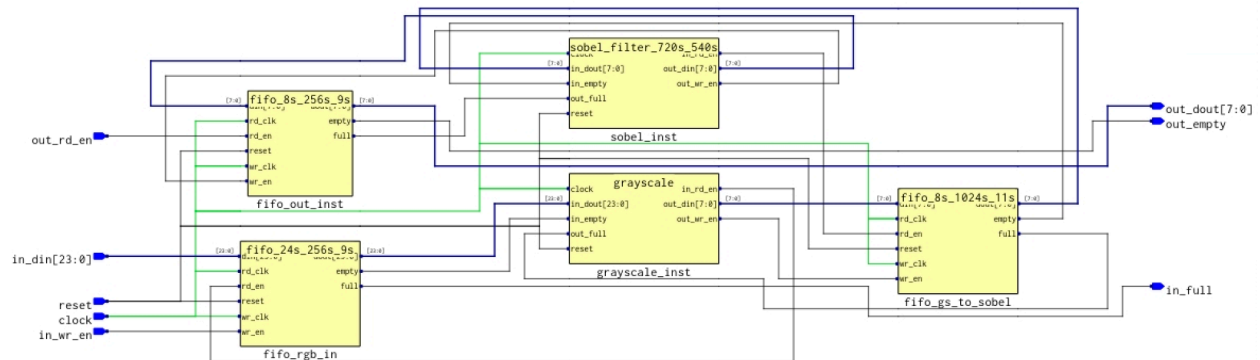
  

Optimizations Summary			

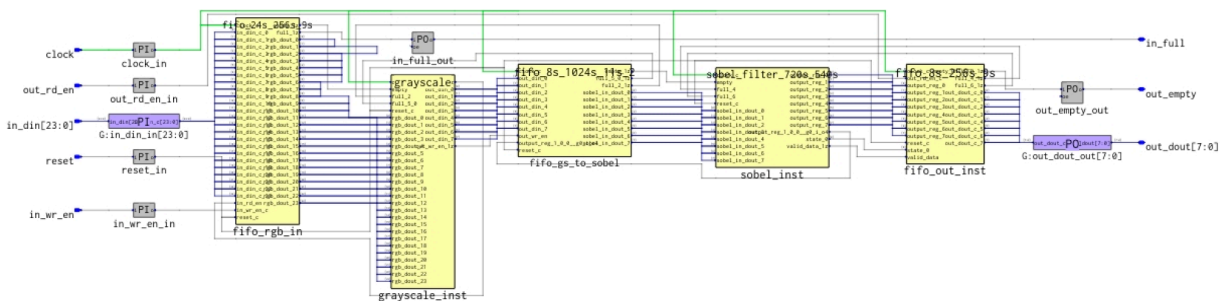
## Hierarchical Area Report:

Module name	ATOMS	ARITHMETIC MC	REGISTERS	SYNC RAMS	MACs
edge_detection_top	205	0	194	5	0
fifo_24s_256s_9s	19	0	19	1	0
fifo_8s_1024s_11s_2	16	0	23	1	0
fifo_8s_256s_9s	20	0	19	1	0
grayscale	17	0	9	0	0
sobel_filter_720s_540s	133	0	124	2	0

## RTL Hierarchical:



## Technology Hierarchical:



## RTL Flattened:

