California State University, Northridge

ECE 524 – Advanced FPGA Design

Fall 2022

Final Project Report

Sobel Edge Detection Using Zybo Z7000

December 16th, 2022

Instructor: Dr. Shahnam Mirzaei

Report by: FNU HARDIK

Table Of Contents

Introduction

This project implements a video processing pipeline that captures video data from an incoming HDMI stream and applies a Sobel Edge Detection filter to create a black and white, edge detected output video stream. The figures below show an image before and after a Sobel filter was applied to it.

Figure 1: Before Sobel (left) and After Sobel (right).

A general high-level Block Diagram of the algorithm used to implement this project is shown in the figure below.

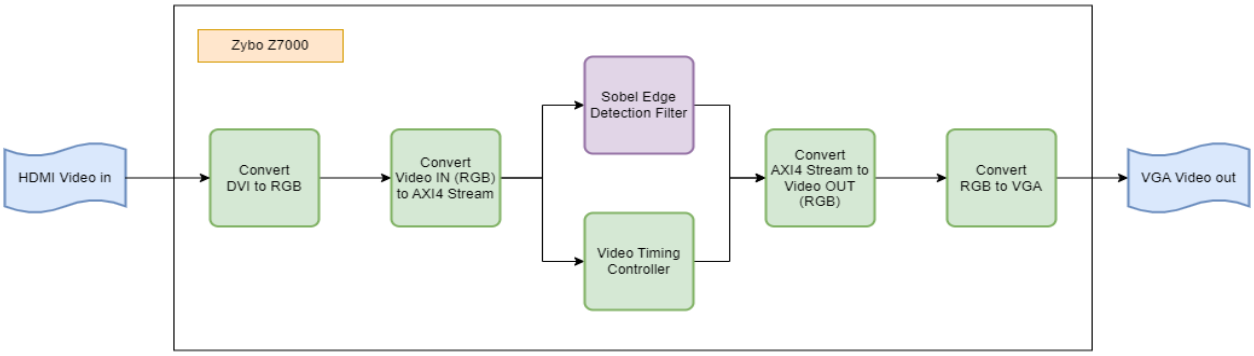


Figure 2: Block Diagram of Sobel Edge Detection Algorithm.

Procedure

We start by creating the required Sobel function module using the HLS workflow and after that we will create a new block design in Vivado to integrate the RTL module we imported from the HLS workflow with other Digilent provided IP modules that will help us in creating the streaming video processing system.

- **Step 1:**Creating the Sobel C++ functions, one thing we need to keep in mind in this step is to utilize the hls_video.h library functions and to avoid using the

much easier hls_opencv.h library functions because the opencv library functions are not designed to be synthesizable when converted into hardware RTL modules. On the other hand the hls_video.h library has most of the same functions as the opencv library and are designed to be optimized for conversion to hardware RTL modules.

- **Step 2:** Creating a Testbench in C++ for Vivado HLS to confirm if our Sobel Edge Detection module works as expected or not.
- **Step 3:** Now we are ready for the RTL Co-Simulation step in which we can create the actual RTL package, in this step we also get the performance indicators like the maximum clock frequency that the synthesis tool has determined for our RTL Package and the Pipelining results from using the HLS Dataflow directives.
- **Step 4:** Now we are ready to start working in Vivado to create the actual block design. We will start by adding all the required IP packages, downloading them from the Digilent IP Interface if they are not provided in the default Vivado version, the IP's we need for this step are Clocking Wizard, DVI to RGB Video Encoder, Video in to AXI-4 Stream, Video Timing Controller, AXI-4 Stream to Video out, RGB to VGA output and Constant.
- **Step 5:** Now we will import the RTL package we created using the HLS workflow and make all the necessary interconnections between all the modules mentioned above, for reference the finished block diagram has been shown in figure 3.
- **Step 6:** This step involves generating the bitstream and making the necessary connections on the board to connect a monitor via the VGA port, and to connect an input stream via the HDMI port, for input you can use any laptop to stream video data from the HDMI port.
- **Step 7:** All that's left is to program the Zybo Z7000 with the bitstream we generated and observe the VGA monitor to see the Sobel Output.

Testing Strategy

The Testing strategy for this project included a testbench for the Sobel Edge Detection Module we created via the HLS workflow and then finally physical testing by uploading the bitstream onto the Zybo Z7000 board and observing the Sobel Edge Detected output on the VGA monitor.

Results & Analysis

The Objective of this project was successfully achieved. The desired functionality was showcased in the presentation and demonstration in class lecture.

The final block diagram created for our implementation has been shown in figure 3 below.

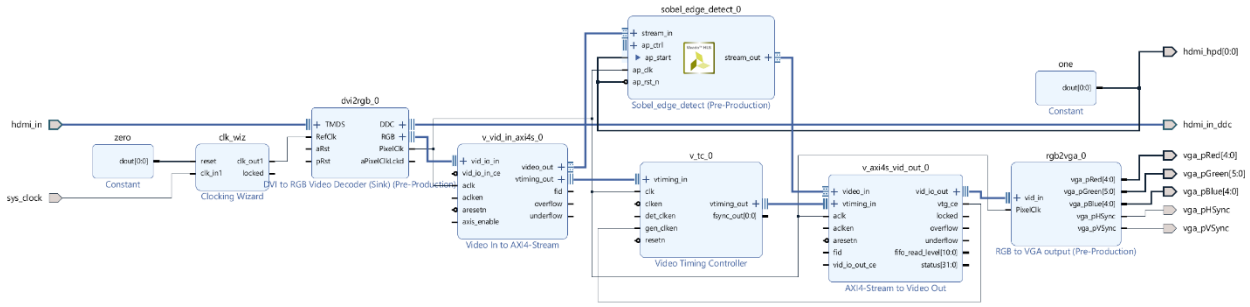


Figure 3: Vivado Block Diagram.

The HLS testbench results have been shown below in figure 4. An input image of a Mars Rover served as the subject here and an output image of a sobel edge detected version of the same rover was generated as the output of the sobel module.

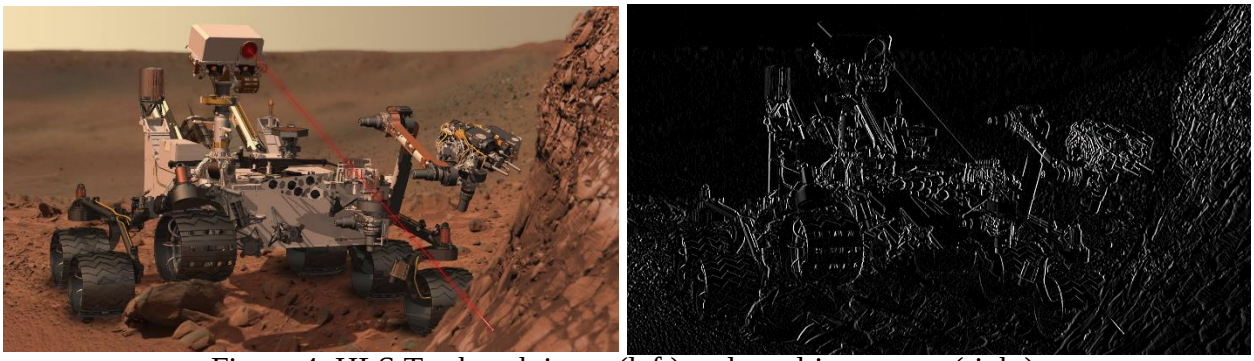


Figure 4: HLS Testbench input (left) and resulting output (right).

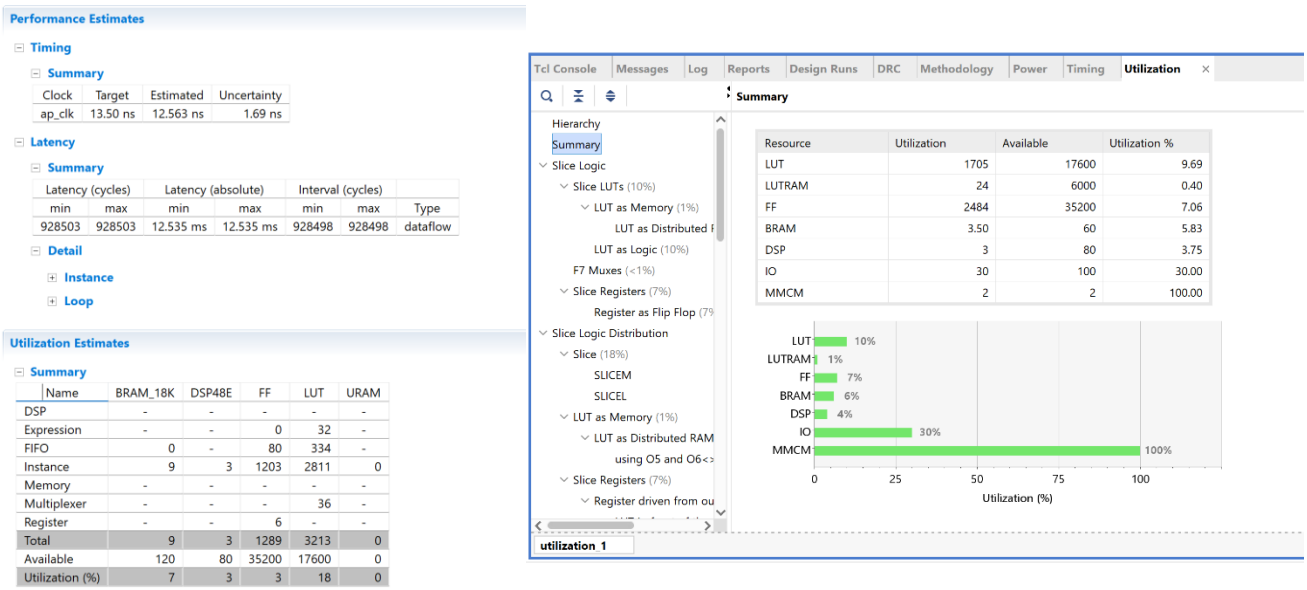
The entire Vivado project has been compressed (.zip) and uploaded onto Github with all the design files and constraints, the HLS (C++) files are uploaded in a separate folder in the same Github repository. The link to the Github repository is: https://github.com/hardiksingh933/Sobel_Edge_Detection_using_Zybo_Z7000.

Conclusions

In this project we strengthened our understanding of how to utilize the HLS workflow to create RTL modules using C/C++, and how to integrate them into the normal Vivado design flows. I also demonstrated the proper functioning of the designed system in a lecture presentation.

Implemented Design Schema

The Implementation Resource Utilizations and the HLS performance estimates for our design are shown in the figures below.



Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	13.50 ns	12.563 ns	1.69 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
928503	928503	12.535 ms	12.535 ms	928498	928498	dataflow

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	32	-
FIFO	0	-	80	334	-
Instance	9	3	1203	2811	0
Memory	-	-	-	-	-
Multiplexer	-	-	-	36	-
Register	-	-	6	-	-
Total	9	3	1289	3213	0
Available	120	80	35200	17600	0
Utilization (%)	7	3	3	18	0

Figure 5: HLS Performance estimates (left) & Resource Utilization Summary (right).

Appendix

Link to Design Files and Vivado Project on Github: https://github.com/hardiksingh933/Sobel_Edge_Detection_using_Zybo_Z7000