CS211 Final Project: Sobel Edge Detection based on Vivado HLS

Overview

Design and implement a Sobel Edge Detection accelerator using Vivado High-Level Synthesis (HLS) tools. Load a picture from memory and display the processed image through a VGA port. Some advanced processing application will earn higher assessment.

The project requirements are divided into the basic part and advanced part. You can get most of the score after finishing the basic part. However, you have to complete some (not all) challenging tasks in the advanced part so as to get the full score.

Background

An important application of HLS is image and signal processing. Apart from simple logic design, the tool also supports us to use some open-source industrial standard framework such as OpenCV. Edge detection is a basic application in Digital Image Processing (DIP) and Computer Vision (CV), the same convolution method can be also applied in feature extraction in Convolutional Neural Network (CNN). If we just use CPU to compute the convolution sequentially, the throughput is limited in some underperforming systems, however, we can use FPGA to accelerate the specific calculation in parallel, which will greatly increase the performance.

In this project, you will learn how to use Vivado HLS tool to generate and optimize your RTL design, then assemble with VGA display and AXI/Video interface in Vivado.

Basic Requirements (75 pts)

1. Load image from Flash or ROM in EGO1, and display on the screen in VGA. **(15 pts)**
    (1) Complete a VGA driver to display the picture in 640x480 resolution.
    (2) Read the picture from ROM or Flash on board.
2. Complete Sobel edge detection module using HLS, and configure the interface in AXI-Stream format. You can use OpenCV libraries in the basic part, optimization will earn higher points in the advanced part. **(20 pts)**
    (1) The module processes a gray-scale image, duplicate into 2 images, and detect the horizontal and vertical edge using the corresponding operators (convolution kernels).
    (2) Add Gaussian filter and Threshold into the module for better display.
    (3) Input and output ports are colored RGB images in AXI-Stream format, easy to integrate with other IP blocks.
3. Run C-Simulation in HLS, synthesize and find the highest clock frequency. **(10 pts)**
    (1) Run C-simulation, compare the HLS processed image and software processed images.
    (2) Find the minimum clock frequency of the solution for best performance.
4. Assemble the above components in Block Design or HDL code, work out timing sequence and dataflow. **(20 pts)**
    (1) Integrate the components in Vivado Block Design (recommended) or HDL wrapper.
    (2) Generate bitstream and implement on EGO1 board.

5. Report. **(10 pts)**

Advanced Requirements (25 pts)

1. Write your own module functions and optimize using HLS directives. (**5~20 pts** based on the module latency and utilization, show your detailed efforts in the report).
2. Switch different pictures by buttons on EGO1 board. **(5 pts)**
3. Use switches/buttons to control the threshold of edge. **(5 pts)**
4. Display information such as threshold, frame rate, image size with 7-seg tube or UART. **(5 pts)**
5. Implement edge detection in other methods such as Canny, state the advantages. **(10 pts)**
6. Implement some advanced application such as Hough Transform Algorithm, Image Correction using HLS based on your edge detection. **(10~20 pts)**
7. Something else you think is interesting and challenging.

Presentation

1. One person for each group, finish independently.
2. You need to show your work in the class at 7-8 classes on Friday of the 6th week.

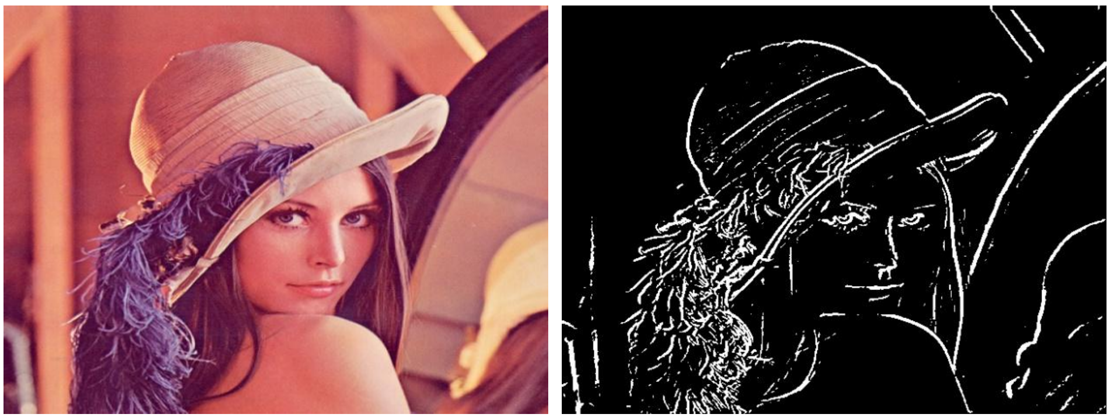
Report

See the template in Sakai.

C Simulation results, comparison between HLS IP processed and software processed images should also be concluded in the report.

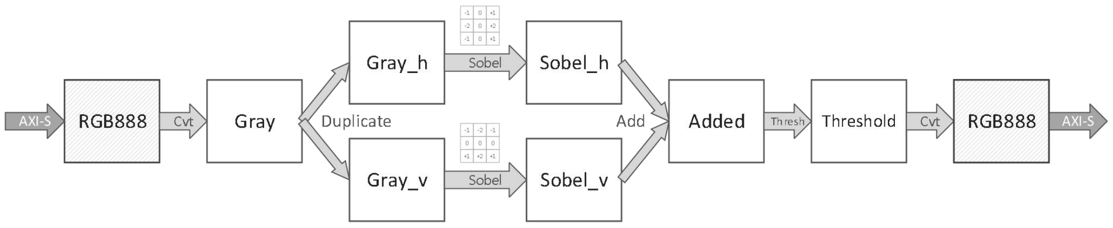
Your optimization and advanced work should also be included in the report.

Example Result:



Reference

1. Sobel Edge Detection



Sobel Operator: a discrete first-order differentiation operator, used to calculate the gradient of a grayscale image, the larger the gradient, the more likely it is likely to be an edge.

Sobel edge detection works by detecting a change in the gradient of the image in both the horizontal and vertical directions. To do this, two convolution filters are applied to the original image, the results of these convolution filters are then combined to determine the magnitude of the gradient.

Gx, Gy, $G = \sqrt{G_x^2 + G_y^2} \approx Gx + Gy$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

To work in higher level of abstraction, we will use Vivado HLS and its HLS_OpenCV and HLS_Video libraries. (Refer to UG902 Chapter 4, part 5 for guidance). The first library *hls_opencv* allows us to work with the very popular OpenCV framework. While the *hls_video* library provides several image processing functions which can accelerate in programmable logic.

As we need our Sobel IP to support AXI-Stream protocol used in video processing, while opencv use matrix form of the image, we will need *HLS::AXIvideo2Mat* and *HLS::Mat2AXIvideo* to convert between them.

HLS project:

Src: hls_sobel.h, hls_sobel.cpp;

Tb: hls_sobel_tb, test.jpg;

Functions needed:

hls_sobel.cpp: *hls::AXIvideo2Mat, hls::CvtColor, hls::Duplicate, hls::Sobel, hls:AddWeighted, hls::Mat2AXIvideo…*

hls_sobel_tb: *IplImage2AXIvideo, cvLoadImage, cvCreateImage, cvReleaseImage…*

Hints:

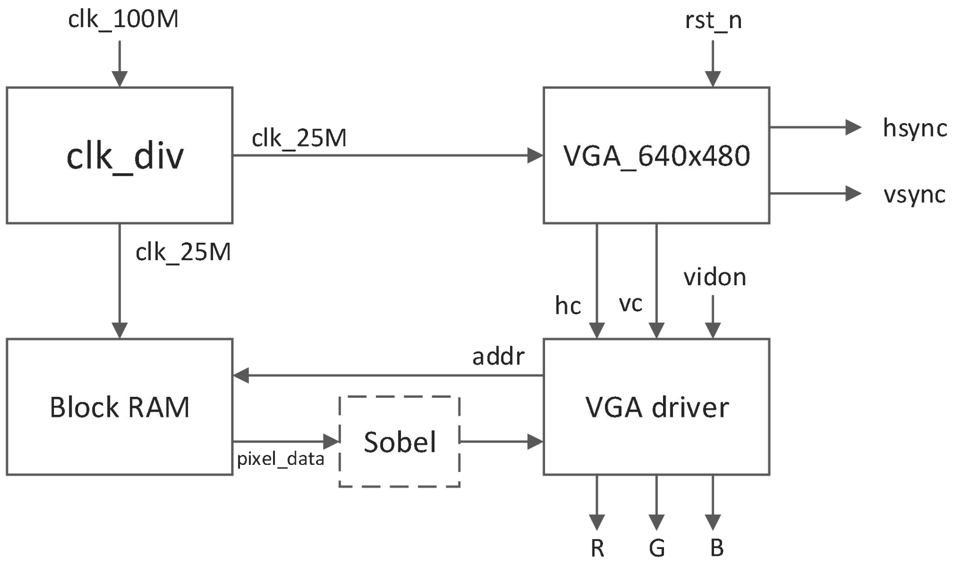
1. HLS “Export RTL error” A known Bug
https://blog.csdn.net/qq_43580646/article/details/123372437

2. Vivado HLS C/RTL co-simulation function does not support structures or classes that contain *hls::stream<>* members in the top-level interface. But Vivado HLS supports synthesis of these structures or classes. (Cannot be co-simed, but can be synthesized)

3. AXI Interface: The I/O port of HLS_Sobel IP should be AXI-Stream format.
Refer to UG902 Chapter 1, part 5.

4. Export RTL and generate IP, include the IP location in Vivado—Project Manager—Settings—IP—Repository, then the HLS IP will be scanned and added into IP catalog.

2. VGA display



1. clk_div

A module to convert the 100MHz clock to 25MHz in VGA display. This can be written as a counter in Verilog, or just customize the “Clocking Wizard” IP.

2. Block RAM

Created by IP catalog, in RAM or ROM, load image from initial “coe” file, the address is given by VGA driver, and output pixel data (in RGB565 format) are input to the VGA driver.

3. vga_640x480

Refer to the lab tutorial, the picture displayed size is 640 pixels in width and 480 pixels in height. Higher resolution and fps will need different set of parameters.

4. VGA driver

This module reads the pixel data from ROM and distribute to R, G, B channels. It needs to calculate the address of the image pixels, and pass to the BRAM.

5. Top design

Assemble these modules together, using a Verilog file, or initialize and connect them in BD design.

Hints:

1. This flow is reading a “coe” image in ROM, and output in VGA form, in this project, you can use IP “Video to AXI Stream”, “Video Timing Controller” and “AXI Stream to Video Out” in an easier way.
2. The “coe” can be generated by MATLAB code, transforming from jpg pictures into memory file.
3. Sobel IP is added between the pixel data and VGA driver, the interface should be customed into AXI stream form to cope with the HLS interface.
4. The total block RAM resource is not available for a full 640x480 picture in 24-bit, you need to figure out a proper way.

3. Video Timing Path

Configuration of Video Timing Controller (VTC), it is recommend to convert video data to AXI-Stream and process in HLS IP core, then output AXI-Stream data will also be converted to Video format using VTC. For more information, refer to ug934 (axi_videoIP), pg043 (v_vid_in_axi4s) and pg044 (v_axis_vid_out).

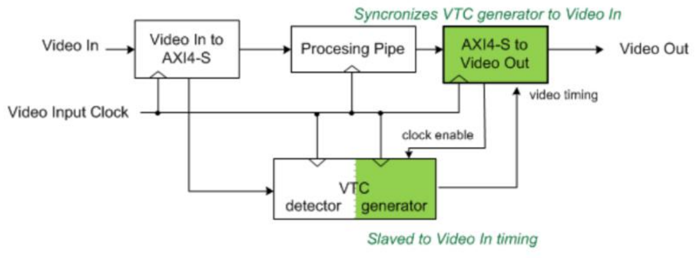


Figure 3-2: **Without VDMA - Slave Timing Mode**

Appendix:

jpg2coe.m

hls_sobel.h, hls_sobel.cpp, hls_sobel_tb.cpp (Not complete)

UG902-vivado-high-level-synthesis_v2017_4.pdf

Some Useful Pages:

HLS——opencv 图像处理知识点

https://blog.csdn.net/weixin_50988214/article/details/115401405

HLS pragma

https://blog.csdn.net/jerwey/article/details/103678275

使用 Vivado HLS 实现 OpenCV 的开发流程

http://www.chinaaet.com/article/217492

HLS #3 Sobel 算例实例应用

https://blog.csdn.net/sements/article/details/102788842

(Note: Some Zynq series chip have integrated ARM cores, which can use PS+PL and SDK development flow, however, EGO1 only has Programmable Logic, we will not use a CPU core for simplicity)

Bridge from "Video In to AXI4-stream" to "AXI4-stream to Video Out" directly with VTC without VDMA

https://blog.csdn.net/pigyyf/article/details/104706429

基于 ROM 的 VGA 图像显示

https://blog.csdn.net/Bunny9__/article/details/120085735

matlab 处理 jpg 转 coe 并载入 BRAM

https://blog.csdn.net/qq_44404407/article/details/115793256