Gateware Engineer Assignment

Image Processing Accelerator for Edge Detection

Introduction

Image processing is a critical application in embedded systems, particularly in **computer vision**, and **edge computing**. One of the fundamental operations in image processing is **edge detection**, which identifies the boundaries of objects within an image. A commonly used method for edge detection is the Sobel filter, which applies a convolution operation to highlight gradients in intensity.

In this assignment, you will design an **FPGA-based hardware accelerator** for edge detection using a **3x3 Sobel filter** and a **pooling layer** to downsample the output. The project will test your skills in **HDL programming (Verilog), clock domain crossing (CDC), timing analysis, and testbench design.

---

1. Background: Convolution & Edge Detection

1.1 Convolution and Sobel Filter

Convolution is a fundamental operation in image processing, where a small matrix (kernel) is applied to an image to extract specific features. The **Sobel filter** is a set of convolutional kernels used to detect edges by calculating gradients in both the **X and Y directions**.

If A is defined as the source image, then the image derived after a Sobel filter convolution is defined as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A$$

$$G = \sqrt{G_x^2 + G_y^2}$$

Where, $*$ represents a convolution operation and G is the final output image. The higher the gradient magnitude (G), the stronger the edge at that pixel.

1.2 Pooling Layer for Downsampling

Pooling reduces the size of an image while preserving key features, making further processing more efficient. You will implement an **average pooling (2×2 window)** layer over the output of the convolutional layer in the previous step (G) that computes the average of a 2×2 region.

---

2. Assignment

2.1 Tools and Target Device

The assignment is meant to be implemented using Xilinx (AMD) Vivado targeting any FPGA device with BRAM capability (Artix 7 is a valid example). Vivado can be downloaded for free from Xilinx's website.

The use of Xilinx provided RTL core libraries (such as BRAM interfaces, FIFOs, etc.) is allowed and encouraged.

2.2 Requirements

A solution will be considered complete if the following requirements are met. Partial solutions are acceptable but will be assessed accordingly.

2.2.1 Input and Output Formats

- The input image must be stored in a **BRAM** instance in an 8-bit unsigned format. You can assume the image size is a maximum of 64 X 64 pixels.
- The output should be an 8-bit image that should be sent out serially using the FPGA pins with **proper handshaking including valid (output) and ready (input) signals**.

2.2.2 Clock frequencies and Clock Domain Crossing (CDC)

- The **BRAM memory interface should run at 100 MHz**.

- The **processing pipeline (Sobel filter + pooling layer) should run at 200 MHz**
- Therefore CDC synchronization is required between these two elements. The choice of CDC implementation is left to you but will be evaluated as per the criteria set out below.

2.2.3 Image Processing

- Implement the **3x3 Sobel filters** using **fixed-point arithmetic**.
- Implement an **average pooling layer** after the Sobel filters.

2.2.4 Timing Analysis, Constraints, and Power

- Ensure setup/hold timing closure using Vivado's Static Timing Analysis (STA) tools.
- Provide IO constraints for the signals coming in and going out from the FPGA.
- Report power consumption using Vivado tools.

2.2.5 Testbench & Simulation

- Simulate a **small 64 × 64 pixel grayscale image** to verify the output.
- Use Vivado's in-built simulator for functional simulation.
- Ensure **correct CDC synchronization** using waveform analysis.

---

3. Evaluation Criteria

1. **Correctness** – Does the design produce expected edge detection results?
2. **Quality** – Is the code well-structured and readable?
3. **Performance** – Is the design fully pipelined and does it achieve the best possible throughput?
4. **Efficiency** – Does the design have optimal timing and resource consumption?
5. **Low-Power Optimizations** – Are clock gating and resource sharing applied effectively?
6. **Simulation & Verification** – Are testbenches well-structured with meaningful validation?
7. **Differentiation** – Based on your own experience in this field, include any relevant improvements to the design which are worth consideration.

4. Deliverables

The deliverables for this assignment are:

1. HDL Source Code (Verilog) for Sobel filter, pooling, and CDC synchronization.
2. Testbench Code & Simulation Waveforms.
3. Timing Report & CDC Analysis.
4. Power Optimization Report.
5. Short Write-Up on challenges and optimizations applied.

Good luck, and happy coding! 🚀