README: FPGA-Based Sobel Edge Detection with Approximate Multipliers

# Project Overview

This project implements a fully synthesizable Sobel edge detection algorithm using Verilog HDL, suitable for FPGA deployment on platforms such as the Xilinx Spartan-7. The design uses a custom-built approximate signed multiplier to significantly reduce power consumption, while maintaining edge detection accuracy. It is capable of detecting edges from 8-bit grayscale images in real-time.

# 🧱 Modular Architecture

The project is built using a modular RTL architecture consisting of the following components:

* pixel\_window\_generator.v – Generates a 3×3 pixel window using FIFOs and shift registers
* sobel\_core.v – Computes Gx and Gy gradients, absolute values, and applies thresholding
* Adder\_16bit.v – Custom structural adder (CSA or RCA based)
* AppMul\_8bitSigned.v – Approximate unsigned multiplier for power-aware convolution
* line\_buffer.v – FIFO-based scanline buffering
* tb\_sobel\_image.v – Testbench for simulating full image-based pipeline
* MATLAB Scripts – For image preprocessing and output visualization

# Features

* Modular, synthesizable Verilog design
* Uses approximate signed multipliers to reduce dynamic power
* Fully pipelined 3×3 Sobel edge detection core (Gx, Gy)
* Threshold-based binary edge map generation
* Pixel window generation using FIFOs and shift registers
* Simulation testbench with image input/output support
* MATLAB integration for pre/post-processing
* Support for 256×256 grayscale images

# Advantages & Perks

* Reduces overall on-chip power by using approximate logic in arithmetic blocks
* Flexible design: change threshold, kernels, or multipliers easily
* Improves hardware efficiency by replacing standard multipliers with approximate units
* RTL-level image processing pipeline is useful for learning and prototyping
* Suits low-power edge devices and FPGA-based camera systems

# Simulation & Image Flow

To simulate the system and generate edge-detected output:

1. Convert image to grayscale and resize to 256×256 using MATLAB
2. Flatten image to CSV using `writematrix(img(:), 'image\_data.csv');`
3. Run simulation in ModelSim or Vivado using tb\_sobel\_image.v
4. Write output to 'sobel\_output.csv' and reshape using MATLAB
5. Display result using `imshow(uint8(reshape(...)))`

# Future Work

* Live video stream processing from camera module
* VGA or HDMI output display
* Gradient magnitude and direction visualization
* Reconfigurable kernel convolution block
* Power analysis and optimization on real FPGA board