Project Proposal: "Design and ASIC Implementation of a CNN Accelerator for Lightweight Object Detection"

1. Project Title

Design and ASIC Implementation of a CNN Accelerator for Lightweight Object Detection

---

2. Objective

The primary objective of this mini-project is to design and implement a hardware accelerator for convolutional neural network (CNN) inference—specifically optimized for lightweight object detection models such as YOLOv8-tiny. The accelerator will target high-performance convolution operations with optimized resource usage, suitable for ASIC implementation using Synopsys tools.

---

3. Motivation

CNN-based object detection models such as YOLOv8-tiny are widely used in embedded AI systems for real-time applications. However, the convolution operations in these models are computationally intensive. While GPUs are commonly used for inference, they are power-hungry and unsuitable for energy-constrained systems. A dedicated hardware accelerator offers a low-power, high-efficiency alternative, especially when implemented as an ASIC.

The Final Year Project I am working on is “Clear Sight for Drivers: Robust Road and Traffic Sign Recognition Algorithm for Advanced Driver Assistance System Under Adverse Weather Conditions.” As part of this, we are developing a hardware solution that includes a camera, a processing unit (planned to be a Raspberry Pi module), and a user interface for the driver. Currently, we are using a Raspberry Pi 4 as the processing unit. However, for more reliable and faster traffic sign detection, we need to improve the hardware setup used for object detection. My motivation for this mini-project is to optimize the hardware as much as possible using the knowledge I’ve gained from the Digital Design and Synthesis module, and to tapeout a chip that functions as a CNN accelerator.

---

4. Project Scope

The project focuses on:

- Implementing a CNN accelerator on RTL using SystemVerilog.
- Focusing on the convolutional layers, activation functions (ReLU), and optional max-pooling.

- Interfacing with a controller and UART for communication with external processors (e.g., Raspberry Pi).
- Preparing the RTL files for ASIC tapeout using Synopsys Design Compiler and other EDA tools.

---

5. System Architecture

The CNN accelerator consists of the following RTL modules:

Module	Description
`input_buffer`	Stores incoming image or feature map
`weight_buffer`	Holds kernel weights for convolution
`conv2d_core`	Performs 3×3 convolution using parallel MAC units
`mac_unit`	Single multiply-accumulate unit
`relu`	Implements ReLU activation (`y = max(0, x)`)
`max_pool`	Optional downsampling layer
`controller`	FSM-based controller to orchestrate the dataflow
`uart_interface`	UART RX/TX logic to communicate with a host processor
`top_module`	Top-level module connecting all blocks together

---

6. Design Flow

1. RTL Design (SystemVerilog)
2. Functional Simulation (Vivado for FPGA prototyping / Synopsys VCS for ASIC flow)
3. Synthesis using Synopsys Design Compiler
4. Place and Route (using Synopsys IC Compiler)
5. Power, Timing, and Area Analysis
6. Tapeout Preparation

---

7. Target Specifications

- Image Input: 32×32 or 64×64 pixels (test case)
- Kernel: 3×3
- Word Length: 8-bit fixed point (Q format)
- Operating Frequency: ~50–100 MHz (depends on synthesis)
- Area/Power: To be evaluated post-synthesis

---

8. Tools

- Language: SystemVerilog
- RTL Simulation: Synopsys VCS
- Synthesis: Synopsys Design Compiler
- PnR: Synopsys IC Compiler
- Visualization: Synopsys DVE (Not sure)

---

9. Expected Outcome

- Fully functional RTL design of a CNN accelerator
- Synthesized netlist optimized for area and power
- Ready-for-tapeout layout using Synopsys toolchain
- Demonstration with test image and model weights

---

10. Gantt Chart