Bharat-AI-SoC-Student-Challenge

Team Members

- Hari Kesavaraj J
- Iniyavan S
- Karthiswaran R

3rd Year B.E Electronics Engineering
(VLSI Design & Technology)
K. S. Rangasamy College of Technology

GitHub: https://github.com/Karthiswaran-R/Bharat-AI-SoC-Student-Challenge
Project Focus: Object detection on Xilinx PYNQ-Z2

1. Abstract

This project presents a CPU-only object detection system implemented on the Xilinx PYNQ-Z2, developed under Problem Statement 5 of the Bharat AI-SoC Student Challenge.

The system performs pedestrian detection using a classical computer vision pipeline based on:

- Histogram of Oriented Gradients (HOG)
- Support Vector Machine (SVM)
- Non-Maximum Suppression (NMS)

All computations are executed on the ARM Cortex-A9 processor of the Zynq-7020 SoC without FPGA acceleration.

This implementation establishes a quantitative baseline for future hardware acceleration using the programmable logic (PL) fabric.

2. Introduction

Embedded object detection is critical in:

- Smart surveillance
- Automotive systems
- Edge AI applications
- Smart city infrastructure

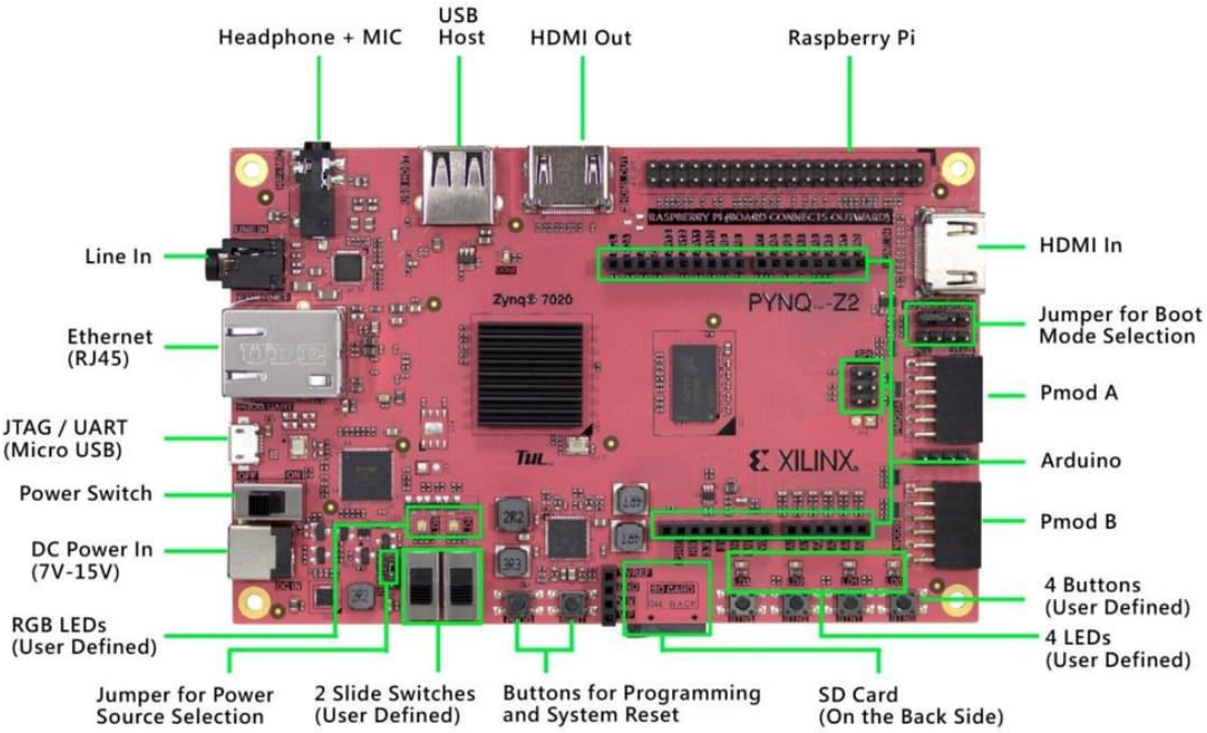
However, real-time object detection on resource-constrained SoCs is challenging due to:

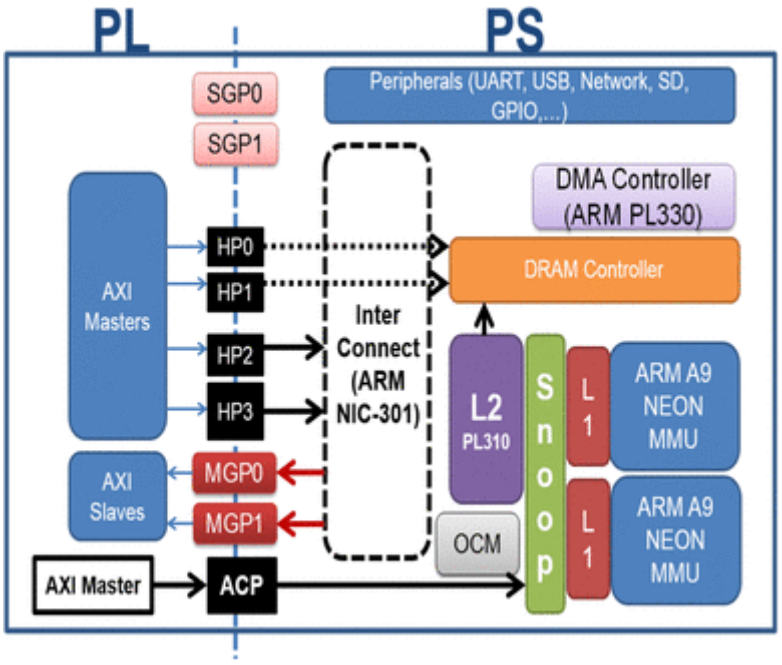
- Limited processing power
- Memory bandwidth constraints
- Power limitations

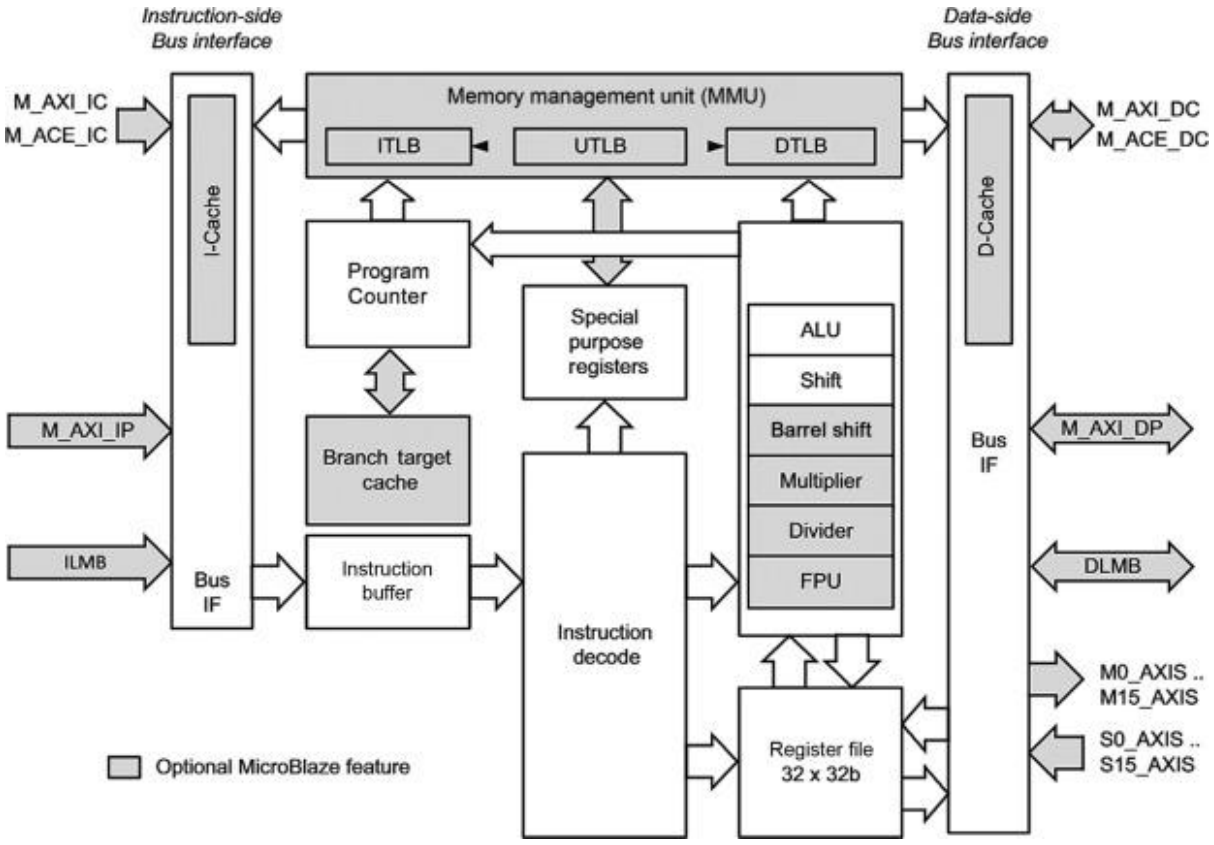
This project evaluates the performance limitations of a CPU-only implementation before introducing FPGA acceleration.

3. System Architecture

3.1 Hardware Platform




PL
PS
SGP0
SGP1
Peripherals (UART, USB, Network, SD, GPIO,...)
DMA Controller (ARM PL330)
AXI Masters
HP0
HP1
HP2
HP3
Inter Connect (ARM NIC-301)
DRAM Controller
L2 PL310
Snoop
L 1
ARM A9 NEON MMU
L 1
ARM A9 NEON MMU
AXI Slaves
MGP0
MGP1
OCM
AXI Master
ACP


Instruction-side Bus interface
Data-side Bus interface
M_AXI_IC
M_ACE_IC
I-Cache
Memory management unit (MMU)
ITLB
UTLB
DTLB
D-Cache
M_AXI_DC
M_ACE_DC
Program Counter
Special purpose registers
ALU
Shift
Barrel shift
Multiplier
Divider
FPU
M_AXI_IP
Branch target cache
Bus IF
M_AXI_DP
ILMB
Bus IF
Instruction buffer
Instruction decode
DLMB
M0_AXIS .. M15_AXIS
S0_AXIS .. S15_AXIS
Register file 32 x 32b
Optional MicroBlaze feature

Hardware Specifications

Component	Description
SoC	Zynq-7020
Processor	Dual-core ARM Cortex-A9
FPGA	Artix-7 Programmable Logic
RAM	512 MB DDR3
OS	PYNQ Linux

4. Methodology

4.1 Image Preprocessing

- Resize image to fixed width (640 px)
- Preserve aspect ratio
- Improve detection consistency

4.2 Feature Extraction – HOG

HOG computes gradient orientation histograms over localized regions.

Advantages:

- Lightweight
- Deterministic runtime
- Suitable for embedded CPU baseline

Limitations:

- Computationally expensive sliding window
- Not real-time on Cortex-A9

4.3 Classification – SVM

Uses OpenCV's pre-trained pedestrian detector.

Binary classification:

- Person
- Background

5. Experimental Setup

- Images stored in /home/xilinx/jupyter_notebooks/images
- Python 3 + OpenCV
- Detection timing measured using time.time()
- Visualization excluded from latency measurement

6. Performance Evaluation

Metrics Used

- Average Latency (ms)
- Minimum Latency (ms)
- Maximum Latency (ms)
- Frames Per Second (FPS)
- Total Execution Time

Output:

Figure 1

Figure 2

```
===== CPU-ONLY PERFORMANCE (PYNQ-Z2) =====
Total images processed : 8
Average latency        : 8437.73 ms
Min latency            : 3854.21 ms
Max latency            : 10118.12 ms
Throughput (FPS)       : 0.12
Total execution time   : 79.71 s
```

Figure 3

Observed Performance

Metric	**Observed Value**
Average Latency	~ 4000 – 5000 ms
FPS	~ 0.2
Real-Time Capability	Not Achieved

7. Analysis

7.1 Why Performance is Low

- Sliding window scanning
- Image pyramid scaling

- CPU-only execution
- No SIMD optimization
- No hardware acceleration

The Cortex-A9 processor lacks sufficient computational throughput for real-time object detection using HOG.

8. Importance of Baseline

This CPU-only implementation provides:

- Reference latency
- Reference FPS
- Engineering comparison point

9. Future Work – FPGA Acceleration Plan

Planned hardware offloading:

- Gradient computation module (PL)
- Sliding window engine (PL)
- Feature histogram computation (PL)
- AXI-based PS-PL communication

Target:

- ≥ 5× performance improvement
- ≥ 2 FPS real-time threshold
- Optimized memory transfers

10. Conclusion

This work successfully:

- Implements CPU-only object detection on PYNQ-Z2
- Benchmarks real embedded performance
- Establishes quantitative baseline

- Prepares for hardware acceleration phase

The results confirm that CPU-only detection is insufficient for real-time embedded applications, justifying FPGA-based acceleration.

12. References

[1] Xilinx Inc., “PYNQ: Python Productivity for Zynq,” *PYNQ Documentation*, 2024. [Online]. Available: https://pynq.readthedocs.io/en/latest/.

[2] OpenCV Team, “Open Source Computer Vision Library (OpenCV) – Version 4.x Documentation,” 2024. [Online]. Available: https://docs.opencv.org/4.x/.

[3] Xilinx Inc., *Zynq-7000 SoC Technical Reference Manual*, UG585, 2023.

[4] Arm Ltd., *ARM Cortex-A9 Technical Reference Manual*, Arm Documentation, 2023.