# FPGA-Based YOLOv4-Tiny Accelerator on AMD Kria KV260

***Abstract***

This project presents the implementation of a hardware-accelerated YOLOv4-Tiny object detection system on the AMD Kria KV260 platform. The accelerator is designed using Vitis HLS and integrated through the Vitis kernel flow. The system offloads convolution-intensive computations from the ARM processor to the FPGA fabric to improve inference performance.

Experimental results show that the FPGA accelerator achieves a **2.31× speedup** over the CPU baseline implementation.

## *Introduction*

Real-time object detection using deep neural networks requires significant computational resources. When executed purely on embedded CPUs, inference latency becomes a bottleneck.

This project aims to:

- Reduce inference latency using FPGA acceleration
- Optimize convolution operations using parallel hardware execution
- Improve throughput compared to CPU-only execution

## *System Architecture*

### Overall Processing Flow

- ➢ Camera/Input Frame
- ➢ Preprocessing
- ➢ FPGA Accelerator
- ➢ Postprocessing
- ➢ Output Display

## *Hardware Implementation*

### *Target Platform*

- AMD Kria KV260
- Device: *xck26-sfvc784-2LV-c*
- Design Flow: Vitis Kernel Flow

### Accelerator Features

- Custom HLS-based accelerator

- AXI memory-mapped interface
- Loop pipelining optimizations
- On-chip BRAM utilization
- Parallel multiply-accumulate (MAC) operations
-

## 5. Resource Utilization

| Resource | Utilization |
|---|---|
| **LUT** | 18% |
| **FF** | 8% |
| **BRAM** | 13% |
| **DSP** | 13% |

The design uses FPGA resources efficiently while leaving room for scalability.

### Performance Evaluation

| Metric | Value |
|---|---|
| **Average Latency** | **850 ms** |
| **Average FPS** | **1.18 FPS** |

### FPGA Accelerator Performance

| Metric | Value |
|---|---|
| **Average Latency** | **368 ms** |
| **Average FPS** | **2.7 FPS** |

**Speedup Calculation**

Speedup is calculated as:

$$Speedup = \frac{CPU\ L \qquad\qquad atency}{FPGA\ L \qquad\qquad atency} = \frac{850}{368} \approx 2.31\times$$

The FPGA accelerator provides a **2.31× improvement** in inference performance compared to CPU-only execution.

**Software Implementation**

The host application performs:

- Loading of .xclbin FPGA binary
- Buffer allocation using OpenCL
- Data transfer between host and device
- Kernel execution
- Latency measurement
- Output retrieval

The system uses Xilinx Runtime (XRT) for FPGA communication.

**Comparison Summary**

| **Metric** | **CPU** | **FPGA** |
|---|---|---|
| **Latency** | 850 ms | 368 ms |
| **FPS** | 1.18 | 2.7 |
| **Speedup** | 1× | **2.31×** |

**Key Contributions**

- Implementation of FPGA-based CNN accelerator
- Integration using Vitis kernel flow
- Performance benchmarking against CPU baseline
- Efficient FPGA resource utilization
- Demonstrated measurable performance improvement

**Advantages**

- Reduced inference latency
- Improved throughput
- Hardware-based parallel computation
- Optimized use of FPGA DSP and BRAM resources

**Limitations**

- Performance constrained by memory bandwidth
- Partial acceleration of YOLO pipeline
- Further optimization possible via deeper pipelining and quantization refinement

**Future Work**

- Increase parallelism for higher FPS
- Optimize DDR bandwidth usage
- Implement deeper pipeline stages
- Explore INT4 or mixed-precision quantization
- Integrate full YOLO backbone acceleration

## Conclusion

This project successfully demonstrates FPGA acceleration of YOLOv4-Tiny on the AMD Kria KV260 platform. The accelerator achieves a **2.31× performance improvement** over the CPU baseline, reducing inference latency from 850 ms to 368 ms and increasing throughput from 1.18 FPS to 2.7 FPS.

The results confirm the effectiveness of FPGA-based hardware acceleration for embedded AI applications.