Research Article

Design And Implementation Of An XNOR-Based Variable-Precision Object Detection Accelerator For Real-Time Edge AI Applications

Dr. Javeed MD[1], Dr. Srinivasa Reddy Dumpa[2], Kammampati Saisri[3], Varda Manasa[4]

[1]Associate Professor, Dept. of ECE, Brilliant Grammar School Educational Society's Group of Institutions - Integrated Campus, Hyderabad, India.Javeed.rahmanee@gmail.com
[2]Associate Professor, Dept. of ECE, Brilliant Grammar School Educational Society's Group of Institutions - Integrated Campus, [3]Hyderabad, India. dr.dsreddi@gmail.com
Lecturer, Dept. of ECE, Brilliant Grammar School Educational Society's Group of Institutions - Integrated Campus, Hyderabad, India. kammampatisaisri472@gmail.com
[4]Dept. of ECE, Brilliant Grammar School Educational Society's Group of Institutions - Integrated Campus, Hyderabad, India, manasagoudgoud257@gmail.com

Abstract

This paper presents a hardware-efficient object detection accelerator based on XNOR-driven variable-precision computation for real-time edge artificial intelligence. The proposed network combines DenseToRes and transition layers to preserve feature information under aggressive quantization. Binary convolution is executed through XNOR and population-count operations, replacing most multiplier-based multiply-accumulate units. To maintain detection accuracy, the architecture supports 1-bit, 2-bit, and 8-bit modes so that sensitive layers can use higher precision while deeper layers operate at reduced precision. A parallel array of 64 processing elements performs multiple output-channel computations concurrently using an output-stationary dataflow. The accelerator integrates on-chip feature and weight memories, data-fetch units, batch normalization, RPReLU activation, quantization, pooling, and lightweight control logic. AXI-based interfacing enables integration with an embedded processing system and external memory. The resulting architecture reduces arithmetic complexity, memory bandwidth, and power consumption while supporting scalable real-time object detection on FPGA-based edge platforms.

Keywords— object detection, XNOR, binarized neural network, variable precision, FPGA accelerator, edge AI, DenseToRes, processing element, AXI.

1. Introduction

Object detection combines classification and localization and is therefore more computationally demanding than image classification. Modern one-stage and two-stage detectors use deep convolutional backbones, multi-scale feature aggregation, and prediction heads to identify objects of different sizes.[1]-[7].

Although GPUs provide high throughput, their power consumption and memory requirements make them unsuitable for many embedded systems.[8] FPGA accelerators offer configurable parallelism and improved energy efficiency, but conventional implementations still rely heavily on multipliers and external-memory traffic.

Binarized neural networks reduce arithmetic cost by constraining weights and activations to one bit. Binary multiplication can then be implemented using XNOR followed by population count. This drastically reduces logic and storage requirements, but full binarization may reduce detection accuracy.[9]-[14]

The proposed work addresses this limitation using variable precision. Critical layers operate at 8-bit or 2-bit precision, whereas computation-tolerant layers use 1-bit processing. A unified XNOR-based datapath supports all modes through bit slicing, bit counting, shifting, and accumulation.[15]-[18]

The architecture also employs DenseToRes blocks to improve information flow under quantization, 64 parallel processing elements for throughput, on-chip buffering for data reuse, and AXI interfaces for system integration.[19]

Fig. 1. Comparison of one-stage and two-stage object detection architectures.

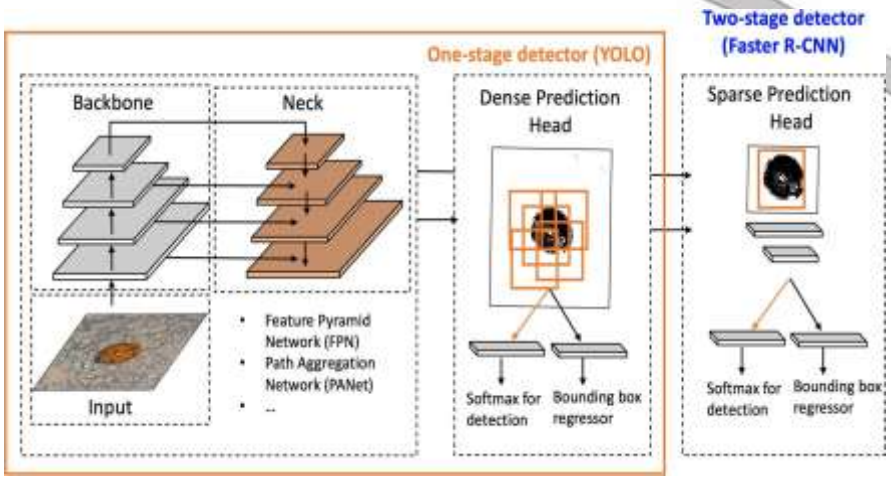
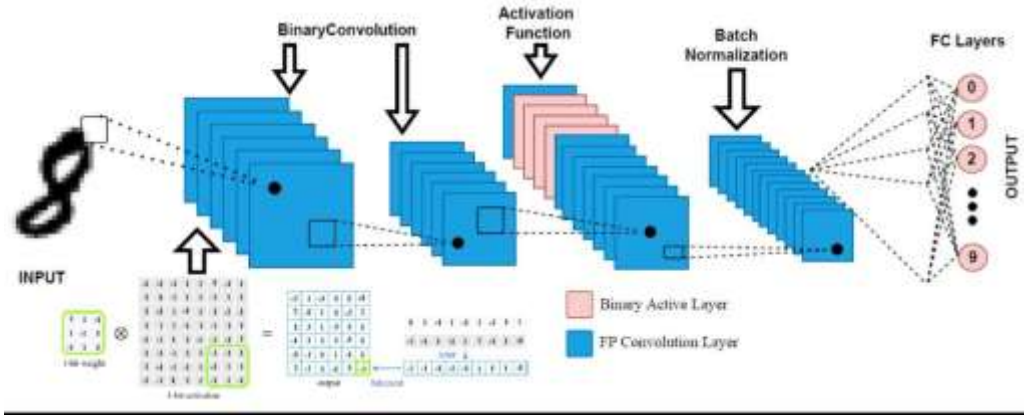


Fig. 2. Binarized neural network with binary convolution, activation, normalization, and classification.



2. Related Work

2.1 Evolution of Object Detection

Object detection evolved from handcrafted Haar, HOG, and SIFT features to region-based CNNs and one-stage detectors such as YOLO and SSD. Two-stage models generally provide strong accuracy but require proposal generation, whereas one-stage models are better suited to low-latency deployment.

2.2 Quantized and Binarized Neural Networks

Quantization reduces storage, bandwidth, and arithmetic complexity. Binarized neural networks and XNOR-Net replace multiplication with logical comparison and bit counting. Bi-Real Net, BinaryDenseNet, and ReactNet improve binary-network accuracy through residual paths, dense connections, and enhanced activations.

2.3 FPGA CNN Accelerators

FPGA accelerators exploit custom dataflow, parallel processing, and on-chip memory. Architectures such as Eyeriss, DianNao, and FINN demonstrate the importance of local data reuse and streaming execution. However, object detection introduces additional memory and multi-scale processing requirements.

2.4 Mixed and Variable Precision

Fixed low precision does not account for layer sensitivity. Mixed-precision methods retain higher precision in the input, output, and other accuracy-critical layers while using aggressive quantization elsewhere. Bit-sliced hardware can reuse the same binary datapath for several precision modes.

2.5 Research Gap

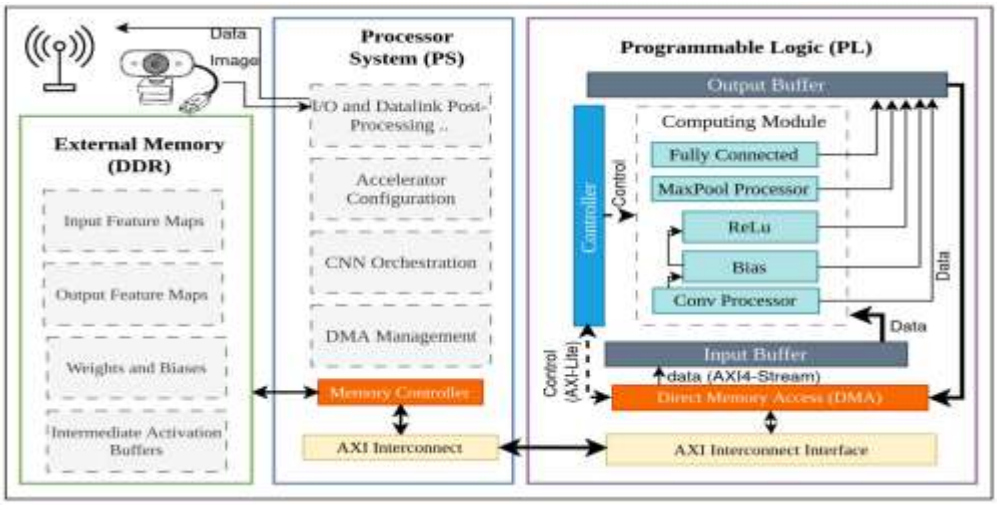
Existing accelerators often use fixed precision, generic CNN dataflows, or separate hardware for binary and multibit computation. A unified object-detection accelerator that combines DenseToRes quantized networks, variable precision, XNOR-popcount arithmetic, 64-way parallelism, and AXI-based deployment is therefore required.

3. Methodology

3.1 Quantized Network Architecture

The proposed detector contains an initial feature extraction stage, repeated DenseToRes blocks, transition layers, and a final detection or classification stage. DenseToRes blocks expand and reuse features, while transition layers reduce spatial dimensions and channel count. Precision is assigned per layer according to sensitivity.

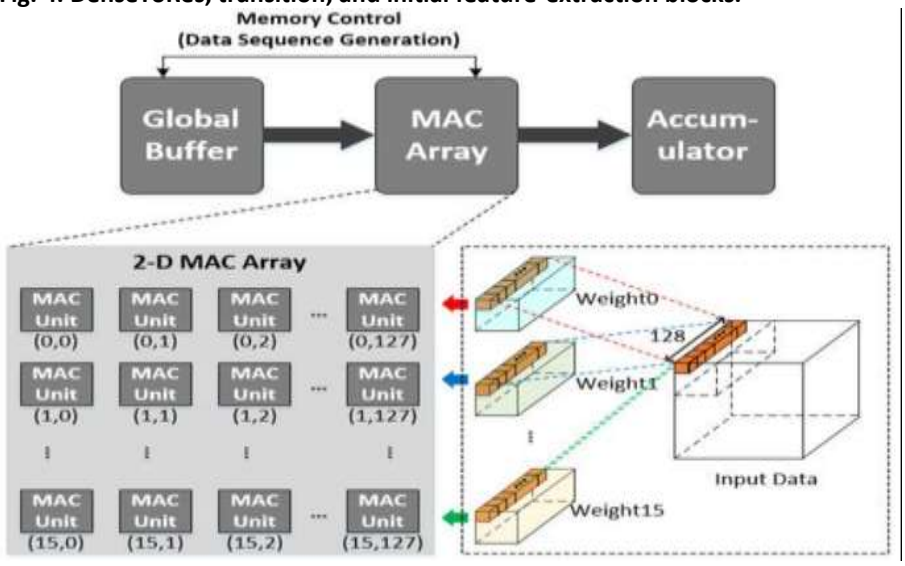
Fig. 3. Overall quantized network architecture with DenseToRes and transition layers.



3.2 DenseToRes and Transition Blocks

The DenseToRes block combines dense concatenation with residual addition. A binarized convolution generates new features, which are concatenated with the original input and projected using a 1×1 convolution. A residual path restores information lost during quantization. Transition layers perform convolution, normalization, residual refinement, and max pooling.

Fig. 4. DenseToRes, transition, and initial feature-extraction blocks.



3.3 Binary XNOR-Popcount Computation

For binary values represented as ±1, multiplication is equivalent to an XNOR comparison in the encoded bit domain. For N binary pairs, the dot product is

$$y = 2\ \text{popcount}\ (\text{XNOR}(\mathbf{a}, \mathbf{w})) - N$$

The popcount result indicates the number of matching bits. The offset N converts the match count into the signed inner-product value. Wide XNOR vectors allow many binary multiplications to be completed in one clock cycle.

3.4 Variable-Precision Arithmetic

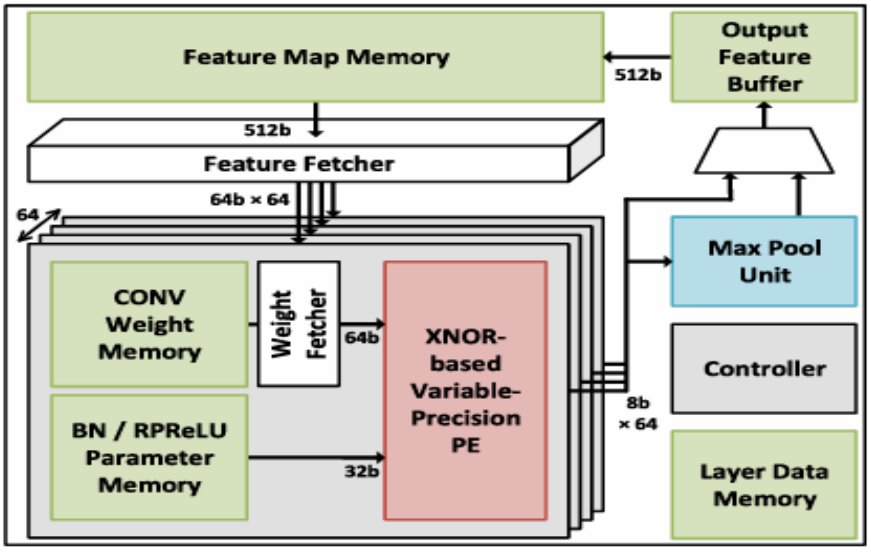
A B-bit activation and a C-bit weight are decomposed into weighted binary planes:

$$a = \sum_{i=0}^{B-1} 2^i a_i, w = \sum_{j=0}^{C-1} 2^j w_j$$

$$a\,w = \sum_{i=0}^{B-1} \sum_{j=0}^{C-1} 2^{i+j} (a_i w_j)$$

Each binary plane product is computed by the XNOR-popcount datapath and reconstructed using shift-add accumulation. The supported modes are 1×1 bit, 2×1 bit, and 8×8 bit. Mode selection is controlled by the layer configuration.

Fig. 5. Variable-precision MAC supporting binary, mixed, and full-precision operation.



3.5 Processing Element Architecture

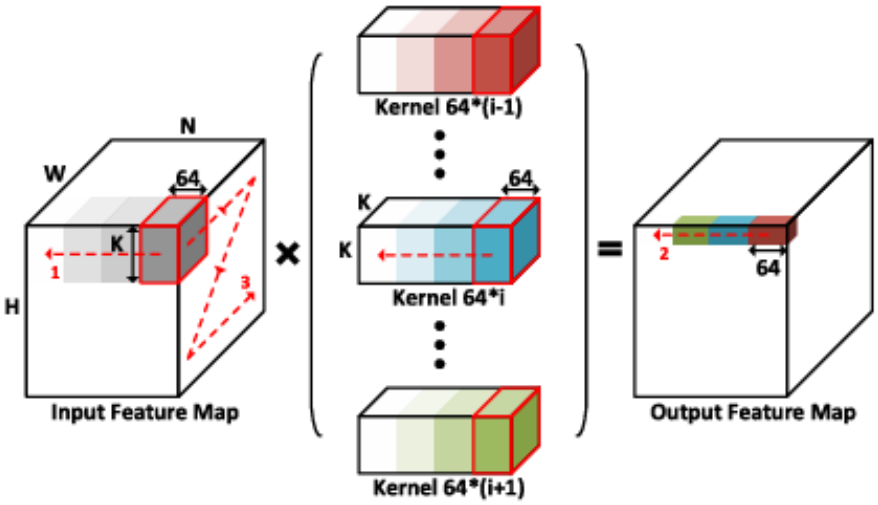
Each processing element contains a variable-precision compute unit, accumulator, batch-normalization stage, RPReLU activation, and quantizer. The output-stationary dataflow retains partial sums locally until one output value is complete, thereby reducing repeated memory accesses.

$$z = \sum_{c} \sum_{k_x} \sum_{k_y} x(c, k_x, k_y)\, w(c, k_x, k_y)$$

$$\text{BN}(z) = \gamma \frac{z - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta$$

RPReLU preserves positive responses and applies a learnable transformation to negative values. The final result is quantized to the storage precision required by the next layer.

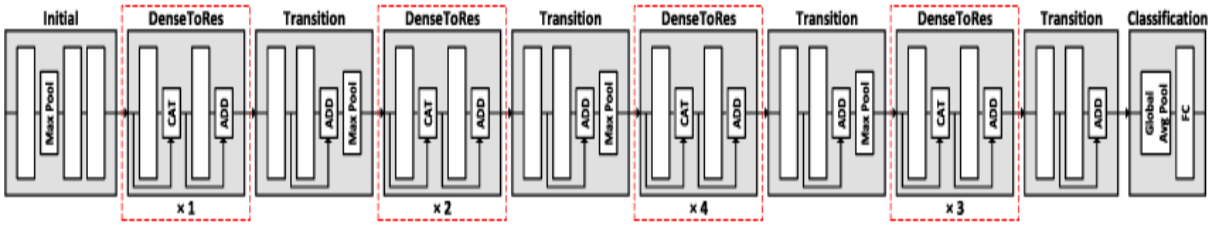
Fig. 6. Processing element integrating variable-precision MAC, normalization, activation, and quantization.



3.6 Parallel Processor and Memory Organization

The processor contains 64 parallel processing elements. Feature-map memory and weight memory use wide words so that all PEs can receive data concurrently. Dedicated fetch units extract the correct bit slices and align them according to the active precision mode. Output buffering decouples computation from external transfer.

Fig. 7. Hardware architecture of the proposed XNOR-based variable-precision processor.



3.7 System Integration

The accelerator is integrated into a heterogeneous platform containing a processing system and programmable logic. The processor performs image preprocessing, accelerator configuration, and output post-processing. AXI-Lite carries configuration registers, while AXI-Stream and DMA transfer high-bandwidth feature data. On-chip buffers minimize access to external DDR memory.

3.8 Control and Execution Flow

1. Load layer parameters, precision mode, dimensions, and addresses through AXI-Lite.
2. Transfer input feature maps and weights into on-chip memories.
3. Fetch activation and weight slices for the selected precision mode.
4. Execute XNOR-popcount or multibit shift-add computation across 64 PEs.
5. Apply batch normalization, RPReLU, quantization, and optional max pooling.
6. Store output feature maps and notify the processing system after layer completion.
7.

3.9 Evaluation Metrics

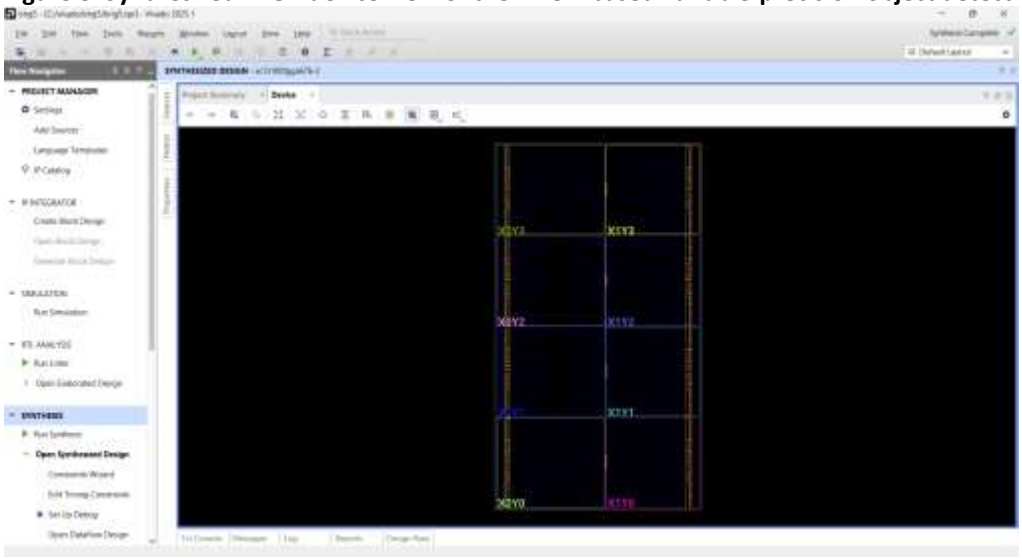
The design should be evaluated using mean average precision, frame rate, latency, throughput, LUT and flip-flop utilization, DSP and BRAM usage, external-memory bandwidth, power, and energy per frame. Accuracy and hardware results should be reported separately for 1-bit, 2-bit, 8-bit, and mixed-precision configurations.

4. Results and Discussion

4.1. FPGA Device Mapping

The synthesized device view confirms successful mapping of the design to the Xilinx Spartan-7 XC7S100-FGGA676-2 device. The clock regions X0Y0 through X1Y3 are visible in the floorplan. The accelerator occupies only a small portion of the available programmable fabric, leaving substantial resources for additional processing elements, image buffering, communication interfaces, or multiple detector instances. The sparse mapping is consistent with the use of XNOR-based arithmetic, which replaces conventional multiplication with low-complexity bitwise operations.

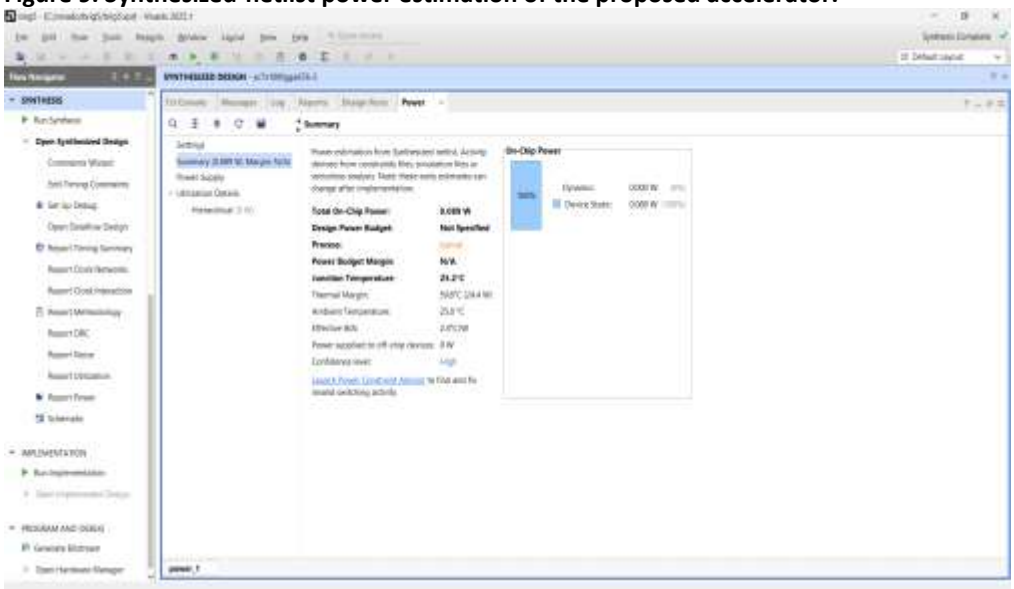
Figure 8. Synthesized FPGA device view of the XNOR-based variable-precision object detection accelerator.



4.2. Power Analysis

The Vivado power report estimates the total on-chip power as 0.089 W. The dynamic power is displayed as 0.000 W, while the complete estimate is assigned to device static power. The junction temperature is 25.2 °C at an ambient temperature of 25.0 °C. A thermal margin of 59.8 °C and an effective junction-to-ambient thermal resistance of 2.4 °C/W are reported, indicating safe thermal operation under the assumed conditions.

Figure 9. Synthesized-netlist power estimation of the proposed accelerator.



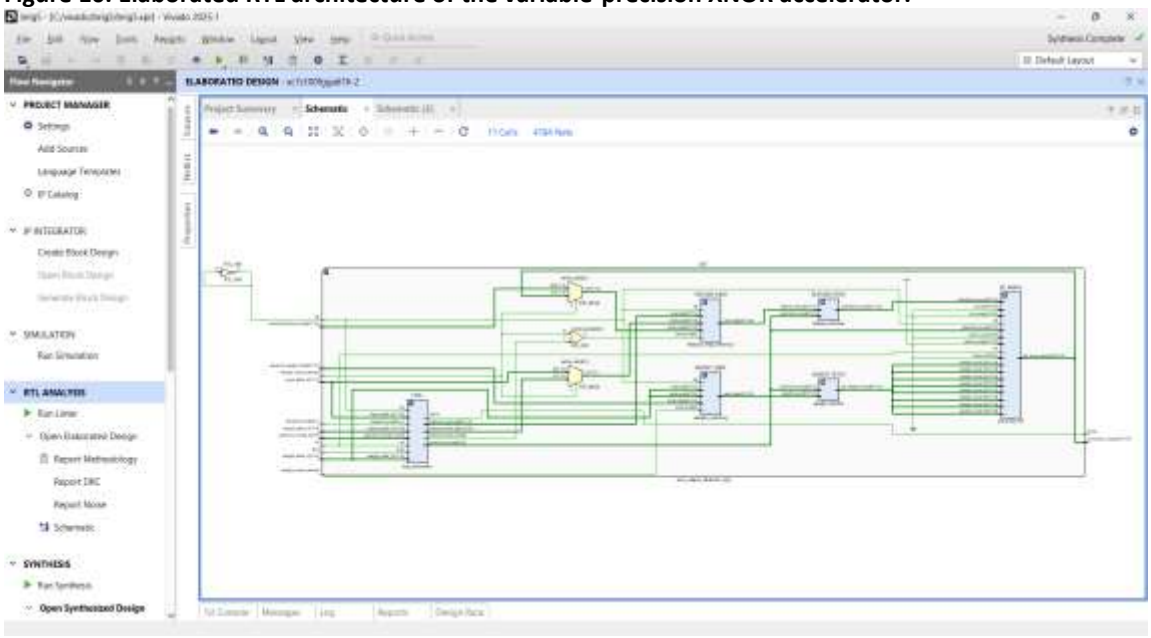
The reported zero dynamic power should not be interpreted as zero energy during active object-detection processing. It indicates that the synthesized power report did not include representative clock and data-transition activity. A post-

implementation power report using SAIF or VCD switching information is recommended for a more accurate estimate of active inference power.

4.3. Elaborated Accelerator Architecture

The elaborated schematic reports 11 major cells and 4184 nets, confirming a hierarchical and highly interconnected processing architecture. The top-level design contains the controller, feature-map memory, weight memory, feature fetch unit, weight fetch unit, and a 64-element processing-element array. The controller coordinates addressing, memory writes, precision selection, start control, and completion signaling. Separate feature and weight memories allow deterministic data delivery to the processing array, while the fetch units align feature activations and weights for parallel computation.

Figure 10. Elaborated RTL architecture of the variable-precision XNOR accelerator.



4.3.1 Processing and Precision Control

The PE array contains 64 parallel processing channels. Each PE receives activation and weight words together with precision-mode and control inputs. In binary or low-precision operation, multiplication is replaced by XNOR comparison followed by bit counting or accumulation. This considerably reduces the logic complexity relative to full-precision multiplication. The precision-mode input enables the same architecture to trade numerical accuracy for throughput, area efficiency, and energy consumption according to the requirements of the edge-AI workload.

For binary operands a and w, the XNOR product can be represented as:

p = XNOR(a, w)

For a vector of K binary elements, the dot-product equivalent is obtained from the number of matching bits:

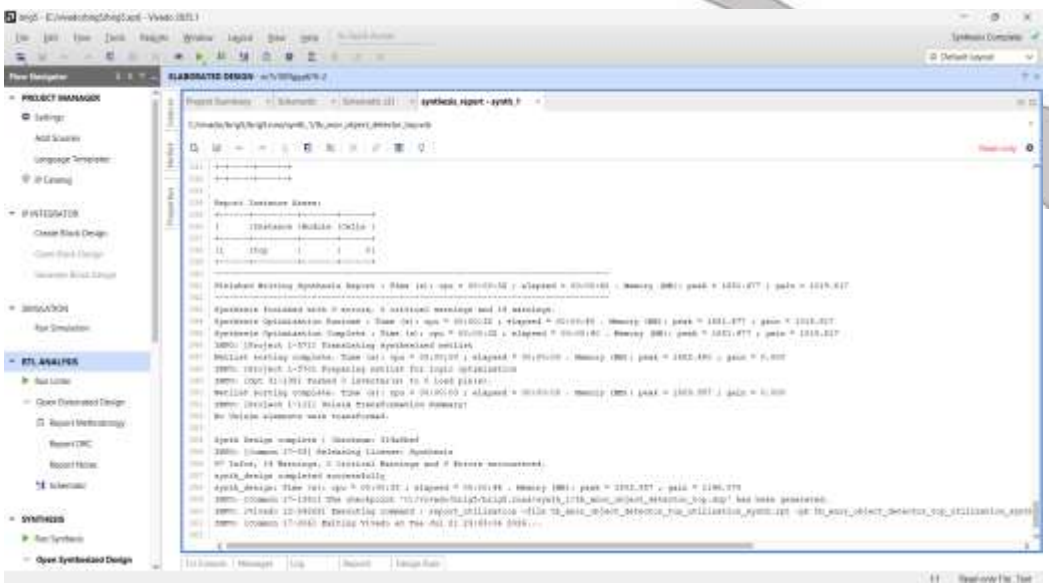
$$y = 2 \cdot \mathrm{popcount}(\mathrm{XNOR}(a, w)) - K$$

This formulation replaces K multipliers with XNOR gates, a population counter, and an accumulator, making the design suitable for real-time edge inference. Variable precision can be implemented by processing multiple bit planes or wider activation and weight words when higher accuracy is required.

4.4. Synthesis Completion and Design Integrity

The synthesis report confirms successful completion with 0 errors and 0 critical warnings. Fourteen ordinary warnings were reported. The report-writing stage required approximately 40 seconds, and the overall synthesis design stage completed in approximately 44 seconds. Peak memory consumption was approximately 1.80 GB. These results confirm that the HDL description is syntactically correct and synthesizable for the selected FPGA.

Figure 11. Vivado synthesis completion report.

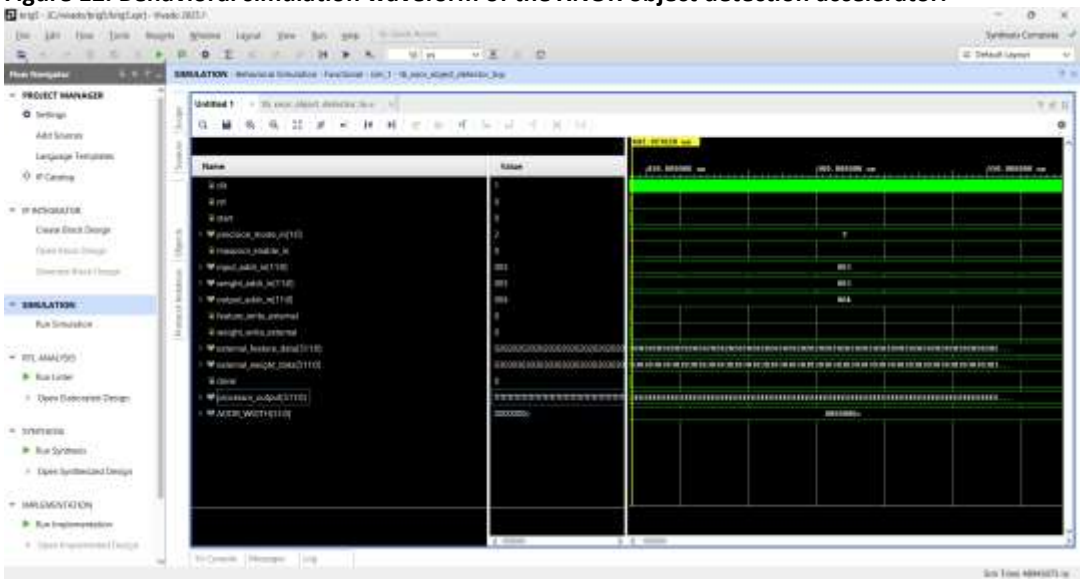


The visible report path contains the name tb_xnor_object_detector_top, and the instance-area table displays zero cells for the shown top entry. This suggests that a testbench or simulation wrapper may have been selected as the synthesis top, or that optimization removed logic not connected to observable outputs. Since the elaborated schematic clearly shows the intended hardware blocks, the final implementation should explicitly select the synthesizable detector module as the top-level source before generating utilization, timing, and bitstream reports. The 14 warnings should also be examined for unconstrained ports, unused signals, width mismatches, or optimized logic.

4.5. Behavioral Simulation Results

The behavioral waveform verifies the principal control and data signals of the accelerator. At the displayed cursor, reset is deasserted, start is low, precision_mode_in is 2, and maxpool_enable_in is 0. The input, weight, and output addresses are 003, 003, and 004, respectively. The external feature data bus contains a repeated 0x02 pattern, while the external weight bus contains a repeated 0x03 pattern. These data patterns verify that wide feature and weight words are correctly supplied to the accelerator.

Figure 12. Behavioral simulation waveform of the XNOR object detection accelerator.



The processor output is shown as a wide all-ones pattern during the observed interval. This may represent an initialized value, a saturated binary result, or a maximum XNOR-match condition for the selected test pattern. The done signal remains low at the

cursor, indicating that the waveform is not positioned on the completion pulse or that a new transaction has not been initiated. For clearer functional evidence, the simulation should be zoomed around the start and done pulses and should include intermediate PE outputs, popcount values, accumulators, and precision-mode transitions.

5. Conclusion

This paper presented an XNOR-based variable-precision object detection accelerator for real-time edge AI. The design combines a quantization-aware DenseToRes network with a unified datapath supporting 1-bit, 2-bit, and 8-bit computation. XNOR-popcount operations replace most multipliers, while shift-add reconstruction supports higher precision when required. A 64-PE output-stationary architecture provides high parallelism, and on-chip feature and weight memories reduce external bandwidth. Batch normalization, RPReLU activation, quantization, pooling, and AXI-based system integration complete the accelerator pipeline. The approach provides a practical balance among detection accuracy, throughput, resource utilization, and power consumption. Future work may include dynamic per-input precision selection, sparsity-aware execution, multi-scale detection heads, and deployment on newer FPGA and ASIC platforms.

References

[1] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, 2012.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[5] R. Girshick, “Fast R-CNN,” in Proceedings of the IEEE International Conference on Computer Vision, 2015.

[6] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” Advances in Neural Information Processing Systems, 2015.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[8] W. Liu et al., “SSD: Single shot multibox detector,” in European Conference on Computer Vision, 2016.

[9] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[10] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” arXiv:1804.02767, 2018.

[11] B. Jacob et al., “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[12] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” Advances in Neural Information Processing Systems, 2016.

[13] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: ImageNet classification using binary convolutional neural networks,” in European Conference on Computer Vision, 2016.

[14] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K. Cheng, “Bi-Real Net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm,” in European Conference on Computer Vision, 2018.

[15] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, “BinaryDenseNet: Developing an architecture for binary neural networks,” in Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019.

[16] Z. Liu, Z. Shen, M. Savvides, and K. Cheng, “ReactNet: Towards precise binary neural network with generalized activation functions,” in European Conference on Computer Vision, 2020.

[17] Y. Chen, T. Krishna, J. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” IEEE Journal of Solid-State Circuits, vol. 52, no. 1, pp. 127–138, 2017.

[18] Y. Umuroglu et al., “FINN: A framework for fast, scalable binarized neural network inference,” in ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017.

[19] P. Judd et al., “Stripes: Bit-serial deep neural network computing,” in Proceedings of the IEEE/ACM International Symposium on Microarchitecture, 2016.