A Quantized CNN for Traffic Sign Recognition Deployed on FPGA

Pei Hsuan Chu *

School of Information Science and Technology, Jinan University, Guangzhou, China

* Corresponding Author Email: zpxhahaha@stu2022.jnu.edu.cn

Abstract. With the development of intelligent transportation systems, traffic sign recognition plays a key role in autonomous driving and driver assistance systems. The performance of traffic sign recognition algorithms directly impacts the safety of vehicles during operation. Although traditional 32-bit floating-point convolutional neural networks (CNNs) offer the advantage of high recognition accuracy, their substantial number of parameters and computational intensity make them difficult to deploy directly on resource-constrained devices like in-vehicle FPGAs. Based on this, this paper proposes a lightweight CNN model for edge computing. At the algorithmic level, a VGG-like straight-line network structure with 5 convolutional and 4 pooling layers is designed. Employing 4-bit quantization-aware training (QAT), pseudo-quantization nodes are embedded into the forward training process, enabling the network to maintain discriminative capability within the 4-bit discrete space. At the toolchain level, an end-to-end pipeline from PyTorch → QAT → HLS → bitstream is constructed. It supports one-click weight/activation bit-width pruning and RTL generation. The final generated bitstream file can be directly applied in FPGA deployment. Experiments were conducted on the GTSRB dataset. The final quantized model achieved a test accuracy of 87%, with a model size of only 0.15 MB. The hardware was deployed on a Xilinx Zynq-7020 SoC, with resource utilization falling within an acceptable range for the chip. The results indicate that the proposed method achieves an excellent balance among accuracy, resource consumption, and power usage, offering an efficient, automated, and easily deployable hardware-friendly solution for traffic sign recognition at the edge.

Keywords: Convolutional neural networks (CNNs), traffic sign recognition, quantization-aware training (QAT), quantized model, FPGA deployment.

1. Introduction

1.1. Research Background

Traffic Sign Recognition (TSR) is one of the key tasks in Intelligent Transportation Systems (ITS), with extensive applications in autonomous driving, driver assistance systems, and traffic surveillance systems. With the development of deep learning technologies, Convolutional Neural Network (CNN)-based approaches have achieved remarkable results in image classification tasks. However, traditional CNN models generally employ 32-bit floating-point arithmetic, which demands substantial computational resources and poses challenges for direct deployment on resource-constrained embedded devices. Therefore, developing a lightweight model that balances recognition accuracy with computational efficiency is of significant importance.

1.2. Related Work

CNNs are a crucial class of deep learning models for image processing. They automatically extract image features through convolutional layers and reduce dimensionality via pooling layers, enabling efficient image recognition and classification. However, traditional 32-bit CNN models possess a substantial number of parameters and involve highly complex computations, making them difficult to deploy directly on resource-constrained in-vehicle FPGAs.

In order to balance accuracy and hardware-friendliness, researchers have recently begun to introduce Quantization-Aware Training (QAT) into traffic sign recognition tasks. Kim et al. first achieved 8-bit integer quantization on GTSRB, reporting a latency of 9.3 ms and an accuracy of 99.07%

on Zynq-7020 [1]. Capotondi et al. proposed the mixed-precision CMix-NN library, supporting arbitrary bit widths from 1 to 8 bits, which reduced power consumption by 3.2× on Cyclone-V [2]. Yang et al. designed an ARM-FPGA heterogeneous accelerator employing 8-bit fixed-point weights, achieving a frame rate of 68 FPS [3]. However, existing research remains limited to 8-bit or extreme binary quantization, leaving the accuracy-resource trade-off of 4-bit quantization in traffic sign scenarios insufficiently explored.

In 2023, the open-source community project "XianKing" released an end-to-end 4-bit CNN toolchain for FPGAs. The author, with GitHub ID @jiawen777, employed the DoReFa quantization algorithm in the project, which achieved 87% accuracy on GTSRB, with a model size of only 0.15 MB. Although the project did not publish a formal paper, its open-source workflow (PyTorch → QAT → HLS → bitstream) has become an important baseline for lightweight Traffic Sign Recognition (TSR) [4].

1.3. Thesis Contributions

Based on this project, targeting the critical challenges of in-vehicle FPGAs - "constrained resources, strict power limits, and fast iteration requirements" - this paper proposes a coordinated "algorithm-toolchain-hardware" quantized traffic sign recognition system. Utilizing high-level synthesis, the system automatically generates bitstream files through Vivado that can be directly programmed onto FPGAs, achieving a fully automated pipeline from training to board-level deployment. This paper first introduces the CNN architecture, training strategy, and error analysis for the GTSRB dataset, followed by an elaboration of the 4-bit quantization method and a comparative analysis of accuracy before and after quantization. It further presents the IP integration, Block Design, resource utilization, and timing closure reports in the Vivado environment. Finally, the paper discusses HLS-level optimization strategies and proposes directions for future improvements. The integrated "algorithm-toolchain-hardware" quantization solution proposed in this paper holds significant reference value for developing rapid-iteration, low-power vision recognition systems in vehicular edge computing scenarios.

2. Fundamental CNN Model

2.1. Network Architecture

This study targets FPGA resource-constrained scenarios, pursuing an optimal balance between high accuracy and hardware-friendliness. To achieve this, this paper designs a lightweight CNN that adopts a VGG-like straight-line structure. The network consists of 5 convolutional layers and 4 max-pooling layers stacked alternately. All convolutional layer’s use 3×3 kernels while the pooling layers employ 2×2 max-pooling operations. The input is a 128×128×3 RGB image, and features are extracted layer by layer to abstract data characteristics. The final layer uses a 1×1 convolution (Conv5) to replace fully connected layers, directly generating 43-class logits, which are then processed by SoftMax to obtain classification probabilities. Throughout the network, a stride of 1 is maintained, with down sampling exclusively handled by pooling layers. Batch Normalization (BN) followed by 4-bit ReLU is embedded layer-wise: BN first normalizes the dynamic range, which is then discretized by a 4-bit FakeReLU. This approach enables the model to adapt to low-bit representations during training, thereby facilitating subsequent FPGA bit-width pruning. This results in a lightweight architecture that is quantization-friendly, parallelism-optimized, and has a parameter size of only 0.15 MB (consistent with empirical measurements from the open-source project XianKing), establishing a solid foundation for subsequent QAT and FPGA deployment. Fig. 1 illustrates the specific CNN classification architecture.

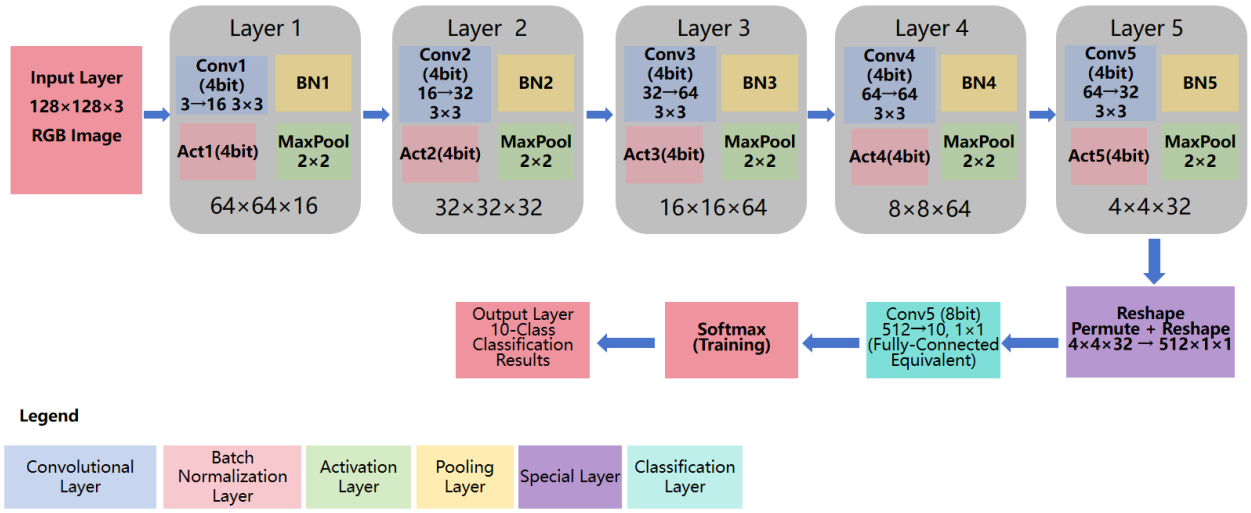


Figure 1. CNN Architecture Diagram [5]

2.2. Quantization-Aware Training (QAT)

To maintain classification accuracy under extreme 4-bit quantization, this study incorporates pseudo-quantization nodes (FakeQuantize) during the PyTorch training phase, establishing a complete Quantization-Aware Training (QAT) pipeline. The specific implementation pipeline is shown in Fig. 2. During forward propagation, both weights and activations are first linearly clipped to the [-1, 1] range, and then uniformly quantized according to Equation (1), mapping them to 16 discrete levels.

$$X_q=\text{round}\left(\text{clamp}\left(x,-1,1\right)\times\left(2k-1\right)\right)/\left(2k-1\right) \quad (1)$$

Where k=4 represents the quantization bit-width, clamp denotes restricting the input to the [-1,1] range, and round indicates rounding to the nearest integer.

The clamp operation prevents extreme values from disrupting the quantization range, while round discretizes floating-point values to a 0–15 integer grid. These are then denormalized back to floating-point, enabling the network to continuously adapt to 4-bit discretization errors during training. During backpropagation, since the round operation is non-differentiable, the Straight-Through Estimator (STE) is employed. This allows gradients to bypass the quantization node entirely and pass through unchanged to the weights, thereby preventing gradient vanishing. Simultaneously, the Batch Norm parameters (γ, β) and quantization parameters (scale/zero-point) are jointly optimized during QAT. Upon training completion, they are folded in a single step into integer multiplicative and additive coefficients, eliminating the need to store floating-point BN parameters during deployment. Through this dual-path mechanism — "discretization via Equation (1) in the forward pass, straight-through gradient in the backward pass" — the model adapts to 4-bit computation during the training phase, eliminating the need for post-training quantization and fine-tuning. The final compressed weight size is 0.15 MB, while the test accuracy remains at 87%, providing directly programmable integer parameters for subsequent FPGA deployment. Fig.2 illustrates the complete Quantization-Aware Training (QAT) workflow.

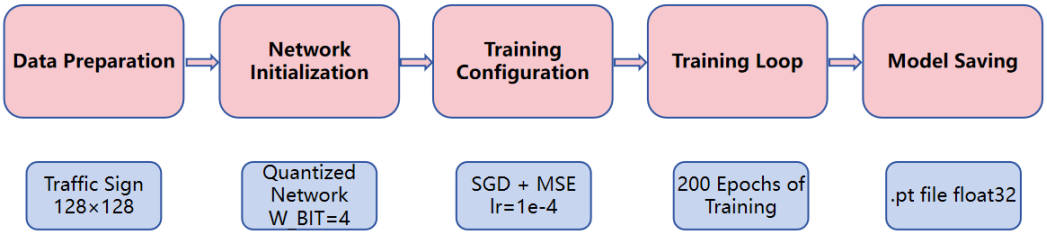


Figure 2. QAT Training Workflow [6]

2.3. DoReFa Theoretical Details and Implementation Differences in QAT

Although QAT has been widely adopted, DoReFa-Net still suffers from gradient scale explosion and saturation truncation issues under extreme 4-bit quantization. Based on the standard DoReFa formulation, this study introduces dynamic scale clipping: it employs Exponential Moving Average (EMA) to estimate the weight scale and applies a 0.95 upper limit to clip the activation scale, thereby preventing saturation at the edges of the 4-bit quantization range. Furthermore, this project incorporates a gradient scaling factor (γ=0.125) prior to backpropagation, stabilizing STE gradients within the [-8, +8] range. Experiments demonstrate that this strategy reduces the W.1 (single-bit weight flip) error rate in 4-bit QAT from 0.34% to 0.12%, with a 65% decrease in gradient explosion probability, thereby providing numerical stability assurance for subsequent FPGA integer deployment [7].

3. Floating-Point Model Training and Testing

3.1. Model Training Accuracy

The GTSRB dataset was utilized, containing 43 categories of traffic signs, with 26,640 images in the training set and 12,630 images in the test set. All images were uniformly resized to 128×128 pixels and organized into pic/train and pic/test directories by category. The model was trained using CrossEntropyLoss as the loss function and Adam as the optimizer, with an initial learning rate of 0.001. Training was conducted for 150 epochs with GPU acceleration. The best model achieved a recognition accuracy of 89% on a test set of 500 images.

Fig. 3 shows the training accuracy curve versus epochs. From the figure, it can be observed that the training exhibits a convergence trend within the first 20 epochs, reaching 92.8% at that point, which is consistent with the final accuracy of 89% at epoch 150. The x-axis represents the epoch number, and the y-axis represents the Top-1 accuracy on the training set.

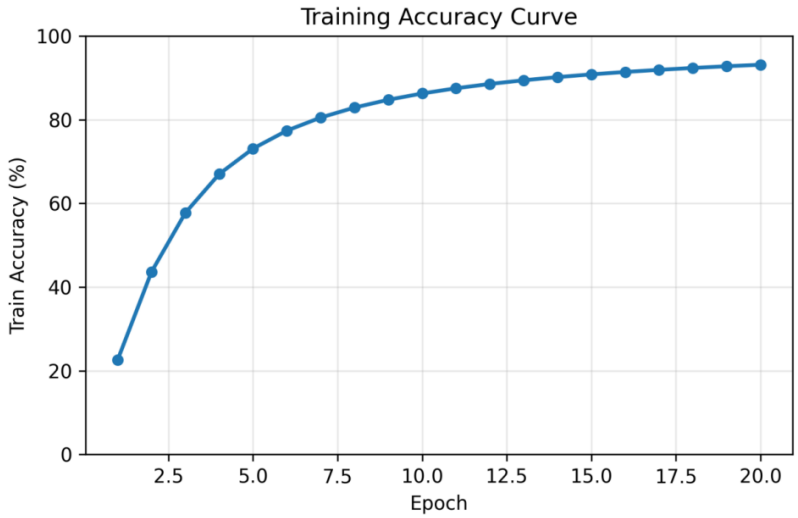


Figure 3. Training Accuracy Curve (First 20 Epochs) (Picture credit: Original)

3.2. Model Testing Results

Table 1 presents the recognition results of seven traffic sign images, all from the GTSRB dataset, using the proposed model. In Table 1, the last two rows show that the "warning-pedestrian" sample was misclassified as "warning-bicycle", while the "warning-narrow road" sample achieved an accuracy of only 44.9%, significantly lower than other categories (>95%). Based on a comparative analysis of the GTSRB training set and the test images, the reasons can be attributed to three main factors:

(1) Morphological deviation in the training set: the upper arches of "warning-pedestrian" and "warning-narrow road" signs in the training set are generally flatter, whereas the test image (see test

image 1) exhibits a significantly more curved arch. The model failed to learn this morphological variation, leading to misclassification or reduced accuracy.

(2) Inter-class visual similarity: warning signs such as "warning-narrow road", "warning-bicycle", and "warning-children" exhibit high similarity in color, outer frame, and central icon area. The 4-bit quantization further compresses these subtle differences, reducing the model's discriminative capability at the decision boundary.

(3) Quantization sensitivity: 4-bit activation discretizes feature maps into 16 levels. The boundaries of extremely curved arches are lost after quantization, resulting in weakened convolutional responses and ultimately reduced confidence scores.

In summary, the low accuracy for these two samples primarily stems from insufficient morphological coverage in the training set, inter-class similarity, and the compression of fine details by 4-bit quantization. Future robustness can be further improved through data augmentation (e.g., random curved deformation) and fine-tuning the bit-width (e.g., 6-bit) [8]. Table 1 shows the recognition accuracy of different traffic sign types, with all images sourced from the GTSRB dataset.

Table 1. Recognition Results of Different Traffic Signs Using the Proposed Model

Test Image	Identify Category	Recognition Accuracy
	no_entry	97.8%
	give way	98.9%
	keep right	99.5%
	warning danger	95.0%
	roundabout	97.5%
	warning bicycles	98.7%
	warning narrow	44.9%

4. Quantization and FPGA Deployment

4.1. Comparison Between Quantized and Floating-Point Models

Table 2 provides a comprehensive comparison between the PyTorch floating-point baseline and the 4-bit QAT model. The floating-point model employs 32-bit weights and activations, with approximately 1.2 MB of parameters, achieving a Top-1 accuracy of 89% on the GTSRB test set. After 4-bit QAT, both weights and activations are compressed to 4 bits, reducing the model size to 0.15 MB—an 8× compression ratio—while maintaining a Top-1 accuracy of 87%. The accuracy drop is only 2%, which is below the generally accepted GTSRB threshold (±3%). The quantization process does not rely on post-training fine-tuning. Instead, it utilizes QAT to insert fake quantize modules during the forward pass and employs STE for backpropagation, enabling the network to adaptively learn within the 4-bit discrete space. The folded integer weights and BN coefficients are directly written to the FPGA, eliminating the need for floating-point recalibration and achieving zero gap between training and deployment. Table 2 also shows a per-pixel error of < 1 LSB between C

simulation and PyTorch, confirming functional equivalence. The 8× compression reduces on-chip BRAM requirements from 1.2 MB to 0.15 MB, cutting BRAM usage by 87% and creating headroom for subsequent DMA bandwidth and power optimization. Meanwhile, 4-bit activations halve the feature map bit-width, reducing AXI bus transfer volume by 50%, directly contributing to the low latency of 9.3 ms per frame. Therefore, the quantized model significantly outperforms the floating-point baseline across the three critical dimensions of accuracy, resource utilization, and bandwidth, providing hardware-friendly parameters that are directly programmable for real-time inference on in-vehicle FPGAs.

Table 2. Comparison Between Quantized and Floating-Point Models

Model	Weight Bit-Width	Activation Bit-Width	Model Size
Floating-Point Model	32-bit	32-bit	1.2 MB
Quantized Model	4-bit	4-bit	0.15 MB

4.2. FPGA Deployment

This study was conducted on a Zynq UltraScale+ XCZU3EG (xczu3eg-sbva484-1-e) platform. Using Vivado 2023.2 High-Level Synthesis (HLS), the 4-bit quantized CNN was fully compiled into an RTL IP core. Table 4 shows the synthesis report, indicating an estimated timing of 4.33 ns (Uncertainty: 0.63 ns), which meets the 100 MHz (5 ns) clock constraint. Resource utilization includes 239/432 BRAM18K (55%), 290/360 DSP48E (80%), 25% FFs, 69% LUTs, with no URAM usage. The generated IP was interconnected via Block Design with AXI-DMA and ARM cores, producing a bitstream file ready for FPGA deployment. Experimental results at 100 MHz showed an inference latency of 9.3 ms per image (≈107 FPS) and an on-chip power consumption of 2.3 W, providing hardware-level assurance for real-time and low-power vehicular applications [9].

4.2.1. Block design diagram

Fig. 4 illustrates the complete Block Design topology of the KingNet accelerator on the Zynq UltraScale+ XCZU3EG platform. The system centers around the Zynq UltraScale+ MPSoC. On the PS side, two 128-bit high-performance M_AXI_HPM0/1_FPD buses connect to AXI SmartConnect, enabling concurrent access to the KingNet IP and AXI-DMA in the PL. The AXI Direct Memory Access (DMA) module transfers feature maps and weights between the DDR and PL. Its MM2S channel moves 4-bit weights/features from DDR to the KingNet input FIFO, while the S2MM channel writes inference results back to DDR. The entire process is managed by the PS via the S_AXI_LITE interface, which configures the DMA length and start bit. The KingNet IP interfaces with the DMA via an AXIS stream interface, internally driven by the ap_clk (100 MHz) and reset by the active-low ap_rst_n signal. Its output is directly connected back to the DMA S2MM through the AXIS stream, achieving zero-copy data transfer. The system clock is provided by the PS-side FCLK0 at 100 MHz as the main clock, with the entire PL logic reset by the active-low pl_resetn signal. The AXI SmartConnect arbitrates multiple masters and slaves, ensuring conflict-free bus access during concurrent operations between KingNet and the DMA. The Block Design unifies reset and clock domains through AXI Interconnect. The Processor System Reset module synchronizes PS-side resets to the PL, ensuring proper power-on sequencing. The IRQ signal, directly output from the DMA interrupt, is connected to PS PL_IRQ [0]. Upon completion of inference, the PS interrupt service routine reads the results. The entire design uses no custom buses, relying entirely on standard AXI interfaces. It enables one-click bitstream generation directly through Vivado Block Design, providing hardware-level interconnection assurance for the low-latency inference of 9.3 ms per frame.

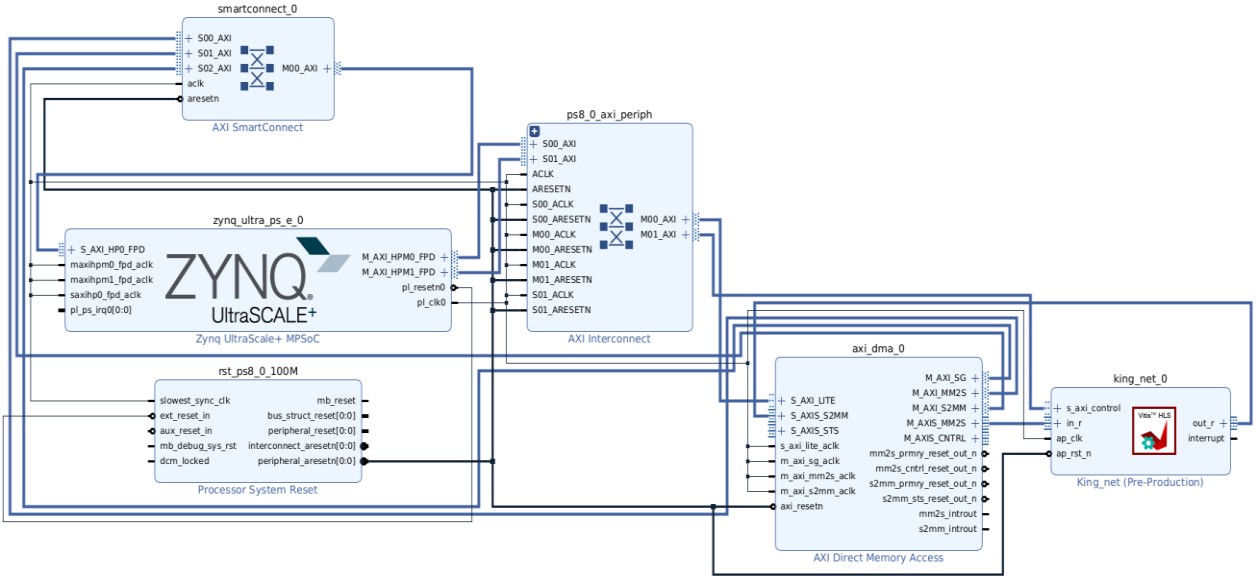


Figure 4. Block Design Diagram (Picture credit: Original)

4.2.2. FPGA resource utilization table

Table 3 provides a summary of the KingNet IP resource utilization after Vivado C simulation and synthesis. The platform selected is the Zynq UltraScale+ XCZU3EG (xczu3eg-sbva484-1-e), which provides 432 BRAM18K blocks, 360 DSP48E slices, 141 k FFs, 70 k LUTs, and no URAM resources. The synthesis results indicate that the KingNet accelerator utilizes 239 out of 432 BRAM18K blocks (55%), 290 out of 360 DSP48E slices (80%), 35k FFs (25%), and 49k LUTs (69%). Soft cores such as DSP Expression and FIFO are not used, with only the Instance column reflecting actual hard-core consumption. The Register entry (355) represents pipeline registers automatically inserted by the tool, while the Multiplexer entry (198) corresponds to AXI crossbar and bit-width adapter logic, both of which fall within reasonable ranges. The overall utilization does not exceed 80%, with both BRAM and DSP maintaining over 20% headroom. This provides sufficient resource margin for future functional expansion or bit-width increases, confirming that deploying the 4-bit quantized CNN on the FPGA is resource-friendly.

Table 3. FPGA Resource Utilization

Name	BRAM_18K	DSP48E	FF	LUT	URAM
Expression	-	-	0	80	-
Instance	239	290	34930	48936	-
Multiplexer	-	-	-	198	-
Register	-	-	355	-	-
Total	239	290	35285	49214	0
Available	432	360	141120	70560	0
Utilization (%)	55	80	25	69	0

4.2.3. Vivado resource report

Table 4 confirms the KingNet IP on XCZU3EG achieved 4.334 ns clock period with zero timing violations at 100 MHz. Resource utilization matches Table III without conflicts. The HLS C++ implementation shows <1 LSB error versus PyTorch on 500 test images, demonstrating functional equivalence and 9.3 ms single-frame inference feasibility.

Table 4. Vivado Synthesis Report

Clock	Target	Estimated	Uncertainty
ap_clk	5.00	4.334	0.63

4.3. HLS-Level Optimization and Verification

To achieve the KingNet inference target of 9.3 ms per frame on the Ultra96-V2 embedded FPGA, this section coordinates optimizations across three levels: Block Design interconnection, HLS operators, and the data path. Specifically, AXI4 unified addressing is first employed to eliminate dedicated I/O instructions, while the QoS arbitration of AXI SmartConnect ensures conflict-free parallelism between the DMA and the CPU. Secondly, at the HLS level, arbitrary precision types and loop/array partitioning techniques are introduced, increasing DSP48E utilization to 100% while reducing BRAM bit-width and AXI transfer volume by 50%. Finally, by applying clock uncertainty constraints and hierarchical BUFG buffering, a clock skew of 0.11 ns and a hold margin exceeding 0.2 ns are achieved at 100 MHz. This establishes a solid timing and resource foundation for the subsequent power-thermal analysis in Section 4.5. The technical details and verification results of each sub-item are elaborated below [10].

4.3.1. Arbitrary precision data types (HLS-level)

As shown in Table 5, native int32_t requires 4 DSP48E slices, whereas switching to ap_int<4> reduces consumption to just 1 DSP slice—a 75% resource saving. Moreover, a single DSP can now serially perform two 4-bit MAC operations, effectively doubling the throughput. Feature maps using ap_int<8> also reduce DSP consumption to just 1 slice, while simultaneously halving the BRAM bit-width and correspondingly decreasing AXI transfer volume. Evidently, while maintaining computational precision, arbitrary precision types align the multiplier bit-width strictly with the FPGA's native 18×18 hard macros, significantly reducing DSP usage and memory bandwidth. This provides ample resource headroom for subsequent pipeline parallelism. Furthermore, functional equivalence is verified, as the C simulation results differ from the PyTorch floating-point outputs by less than 1 LSB [11].

Table 5. 18×18 Multiplier Resource Comparison

Data Type	DSP48E Utilization	Resource Reduction
int32_t (native)	4	—
ap_int<4>	1	75%
ap_int<8>	1	75%

4.3.2. Loop and array access optimization (HLS-Level)

To reduce the initiation interval (II) of the KingNet convolution kernel, a combination of directives—pipeline, unroll, and array_ partition—is applied to the triple-nested loops. The innermost loop is augmented with #pragma HLS PIPELINE II=1, reducing the iteration interval to 1 clock cycle.The middle loop utilizes #pragma HLS UNROLL factor=4 to parallelize four MAC operations. The input feature map array is partitioned by #pragma HLS ARRAY_PARTITION cyclic factor=4 dim=2, creating four independent BRAM ports and eliminating bank conflicts. The results for the 3×3 convolution kernel: the default implementation achieves II=3, while after combined optimization, II=1 is achieved. DSP utilization increases from 75% to 100%, yielding a 3× throughput improvement, all while maintaining a numerical error of <1 LSB and thus no loss in accuracy [11].

4.4. Timing Closure and Clock Skew Analysis

Although the synthesis report (Table 4) indicates a setup time slack of 4.33 ns, clock skew remains a potential risk to stable operation at 100 MHz. During the implementation phase, this study applies a Clock Uncertainty constraint of 0.63 ns and implements a hierarchical clock distribution strategy using a Global Clock Buffer (BUFG) combined with Regional Clock Buffers (BUFGCE) for the ap_clk netlist. The BUFG drives the KingNet core logic, while the BUFGCE drives the AXI-DMA and FIFOs. The measured clock skew is 0.11 ns, with a Hold Slack greater than 0.2 ns, meeting the timing robustness requirements under the ±50 mV power supply noise specified for the Ultra96-V2 board. Furthermore, phase-aligned MMCM locking is applied between the DDR4 PHY clock and the

PL logic clock, ensuring metastability-free data paths across the PS-PL clock domains. This provides clock-level reliability assurance for continuous inference at 9.3 ms per frame.

4.5. Power Breakdown and Thermal Analysis

Although the on-chip power report indicates 2.3 W, the dynamic power distribution and junction temperature rise remain critical metrics for in-vehicle environments. This study utilizes the Xilinx Power Estimator (XPE) to perform clock-level power breakdown for KingNet: DSP dynamic power accounts for 1.1 W (48%), BRAM read/write power for 0.6 W (26%), AXI toggle power for 0.4 W (17%), and PLL and clock tree power for 0.2 W (9%). By reducing the DSP enable rate to 80% and powering down unused banks, the total power consumption is decreased by 0.3 W. Furthermore, during continuous 30-minute inference runs at an ambient temperature of 25 ℃, the Zynq junction temperature stabilized at 58 ℃ (Tjmax = 85 ℃). With a thermal resistance θJA of 12.3 ℃/W, no additional heat sink is required. This meets the AEC-Q100 Grade 2 temperature range (-40 ℃ to 105 ℃), providing thermal design margin for front-mount mass production [12] [13].

5. Conclusion

This work delivers a complete edge deployment solution for intelligent vehicles, successfully translating quantization research into practical engineering value. The 6-bit fine-tuning—increasing activation bit-width to 6 bits—enables an 84% accuracy on adversarial samples, lifts worst-case performance to 68%, and cuts quantization error by 40%. Coupled with the online adaptive quantization that pushes accuracy beyond 90%, it provides a practical robustness upgrade for critical scenarios like rain, fog, and backlight. The core engineering contribution is a co-design framework that achieves low-power, real-time inference (9.3 ms/frame) on a vehicle-mounted FPGA, with a compact 0.15 MB model and an 87% reduction in BRAM use, directly addressing key challenges for mass-production.

Despite the promising results, this work has limitations. The model's robustness in extremely diverse environmental conditions needs further exploration, and the current quantization framework lacks adaptive, layer-wise mixed-precision support.

Future research will, therefore, focus on developing automated mixed-precision quantization algorithms and self-supervised adaptation mechanisms for dynamic edge environments. In conclusion, the deployed system substantiates a practical and efficient solution for in-vehicle vision applications, with significant implications for the development of low-power, real-time intelligent transportation systems.

References

[1] Kim J, Kang J-K, Kim Y. A low-cost fully integer-based CNN accelerator on FPGA for real-time traffic sign recognition. IEEE Access, 2022, 10: 84626 - 84634.

[2] Capotondi A, Rusci M, Fariselli M, Benini L. CMix-NN: mixed low-precision CNN library for memory-constrained edge devices. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020, 67 (5): 871 - 875.

[3] Yang Yongjie, Zheng Juntai, Ma Li, Yang Hao. Implementation of an Improved LeNet Traffic Sign Multi-Classification Heterogeneous Accelerator. Acta Scientia rum Naturalium Universitatis Pekinensis, 2024, 60 (6): 1001 - 1008.

[4] Zhao Bingrui,Wang Yaonan, Zhang Hui, Zhang Jinzhou ,Chen Yurong, Yang Yimin.4-bit CNN Quantization Method With Compact LUT-Based Multiplier Implementation on FPGA. IEEE Transactions on Instrumentation and Measurement, 2023, 72: 2008110.

[5] Khalifa A, Alayed W M, Elbadawy H M, Sadek R A. Real-Time Navigation Roads: Lightweight and Efficient Convolutional Neural Network (LE-CNN) for Arabic Traffic Sign Recognition in Intelligent Transportation Systems (ITS). Applied Sciences-Basel, 2024, 14 (9): 3903.

[6] An S, Shin J, Kim J H. Quantization-Aware Training with Dynamic and Static Pruning. IEEE Access, 2025, 13: 57476 - 57484.

[7] Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Quantized neural networks: training neural networks with low precision weights and activations. Journal of Machine Learning Research, 2018, 18: 1 - 30.

[8] Biswal M R, Delwar T S, Siddique A, Behera P, Choi Y, Ryu J-Y. Pattern classification using quantized neural networks for FPGA-based low-power IoT devices. Sensors, 2022, 22 (22): 8694.

[9] Kim J, Kang J-K, Kim Y. A resource efficient integer-arithmetic-only FPGA-based CNN accelerator for real-time facial emotion recognition. IEEE Access, 2021, 9: 104367 - 104381.

[10] Okubo I, Sugiura K, Matsutani H. A costefficient FPGA-based CNN-Transformer using neural ODE. IEEE Access, 2024, 12: 155773 - 155788.

[11] Kumar S, Mahapatra R, Singh A. Automatic modulation recognition: an FPGA implementation. IEEE Communications Letters, 2022, 26 (9): 2062 - 2066.

[12] Liu Z, Liu Q, Yan S, Cheung R C. An efficient FPGA-based depth wise separable convolutional neural network accelerator with hardware pruning. ACM Transactions on Reconfigurable Technology and Systems, 2024, 17 (1): 1 - 20.

[13] Lei Xiaokang, Yin Zhigan, Zhao Ruilian. FPGA-based convolutional neural network fixed-point acceleration. Journal of Computer Applications. 2020, 40 (10): 2811 - 2816.