Real-Time Semantic Segmentation on FPGA for Autonomous Vehicles Using LMIINet with the CGRA4ML Framework

Amir Mohammad Khadem Hosseini and Sattar Mirzakuchaki

Abstract—Semantic segmentation has emerged as a fundamental problem in computer vision, gaining particular importance in real-time applications such as autonomous driving. The main challenge is achieving high accuracy while operating under computational and hardware constraints. In this research, we present an FPGA-based implementation of real-time semantic segmentation leveraging the lightweight LMIINet architecture and the Coarse-Grained Reconfigurable Array for Machine Learning (CGRA4ML) hardware framework. The model was trained using Quantization-Aware Training (QAT) with 8-bit precision on the Cityscapes dataset, reducing memory footprint by a factor of four while enabling efficient fixed-point computations. Necessary modifications were applied to adapt the model to CGRA4ML constraints, including simplifying skip connections, employing hardware-friendly operations such as depthwise-separable and 1A-1 convolutions, and redesigning parts of the Flatten Transformer. Our implementation achieves approximately 90% pixel accuracy and 45% mean Intersection-over-Union (mIoU), operating in real-time at $20\,\mathrm{frames\ per\ second}$ (FPS) with $50.1\,\mathrm{ms}$ latency on the ZCU104 FPGA board. The results demonstrate the potential of CGRA4ML, with its flexibility in mapping modern layers and off-chip memory utilization for skip connections, provides a path for implementing advanced semantic segmentation networks on FPGA for real-time applications to outperform traditional GPU solutions in terms of power efficiency while maintaining competitive accuracy. The code for this project is publicly available at https://github.com/STAmirr/cgra4ml_semantic_segmentation.

Index Terms—FPGA, LMIINet, semantic segmentation, real-time inference, CGRA4ML, autonomous vehicles, hardware acceleration, quantization-aware training (QAT)

I. INTRODUCTION

RELIABLE autonomous driving demands dense scene understanding at video rates under strict power, thermal, and cost constraints. Among perception tasks, *semantic segmentation* provides pixel-wise interpretation of a scene—separating roadway, sidewalk, vehicles, pedestrians, and signage—and directly supports planning and control. Over the last decade, accuracy has improved rapidly with deeper backbones and attention/Transformer modules; however, typical GPU deployments remain power-hungry and introduce non-deterministic latency due to driver scheduling, memory traffic, and batching requirements. These properties are problematic for embedded electronic control units (ECUs) that must meet hard end-to-end deadlines and functional-safety constraints in automotive environments.

A. Why FPGA for real-time semantic segmentation.

Field-programmable gate arrays (FPGAs) provide fine-grained spatial parallelism and deterministic, clocked pipelines with tight control over the memory hierarchy. Compared to general-purpose GPUs, the accelerator datapath can be specialized to the exact operations in the network (e.g., depthwise separable convolutions, pointwise convolutions, attention projections), enabling:

- **Deterministic latency:** deep pipelines with static initiation intervals (II) and bounded queues, avoiding OS and driver jitter. End-to-end latency scales with pipeline depth rather than input batch size; real-time inference at batch $b{=}1$ is natural rather than penalized.
- **Reduced data movement:** line buffers and on-chip SRAM tile the feature maps to minimize off-chip DRAM traffic—a dominant source of latency and energy on GPUs. Streaming between layers eliminates intermediate host-device transfers.
- **Quantization-first arithmetic:** fixed-point INT8/INT4 MAC arrays match the needs of quantization-aware training (QAT), lowering energy per operation while preserving accuracy under careful calibration.
- **Operator fusion and dataflow execution:** custom pipelines fuse convolution, normalization, and activation, collapsing kernel boundaries and removing launch overheads.
- **Safety and lifecycle:** reconfigurability enables over-the-air updates and field patches while preserving a verified timing budget—useful for ISO 26262 processes.

B. From FPGA to ASIC: The Migration Path and Cost Rationale

A hardware-software co-design validated on FPGA establishes the micro-architecture (compute arrays, buffer sizes, dataflow, and quantization) that can later be hardened as an ASIC. While ASICs incur non-recurring engineering (NRE) costs, at production scale their unit cost and energy per inference are substantially lower than GPU modules or large FPGAs. Crucially, the design techniques we employ—tiling sizes, reuse factors, loop unrolling, operand bit-widths, and on-chip buffer topology—transfer directly to an ASIC netlist. Thus, the same design that yields deterministic $\leq\,50\,\mathrm{ms}$

Corresponding author: Amir Mohammad Khadem Hosseini.

Amir Mohammad Khadem Hosseini and Sattar Mirzakuchaki are with the Department of Electrical Engineering, Iran University of Science and Technology (IUST), Tehran 16846-13114, Iran (e-mails: am_khadem@cmps2.iust.ac.ir, m_kuchaki@iust.ac.ir).

latency and real-time throughput on FPGA provides a blueprint for a cost-optimized ASIC with even lower latency, higher TOPS/W, and reduced bill-of-materials at volume.

C. Problem setting and gap.

Lightweight convolutional networks such as ENet and ERFNet [5], [6] and more recent hybrid CNN-Transformer models (e.g., SegFormer and LMIINet [1], [7]) deliver compelling accuracy-speed trade-offs on GPUs. FPGA implementations to date have typically focused on very compact CNNs or aggressive quantization (e.g., ENetHQ) to reach real-time, often sacrificing accuracy on benchmarks like Cityscapes [9]. Conversely, Transformer-augmented models improve global context reasoning but are rarely mapped efficiently to reconfigurable logic due to memory access patterns and attention bottlenecks. This work bridges that gap via a co-design that preserves accuracy-critical components of LMIINet while restructuring the network and its schedule for a spatial dataflow backend.

D. Our approach in brief.

We re-architect *LMIINet* for FPGA deployment and map it with the *CGRA4ML* framework. The network integrates a lightweight encoder-decoder with a reduced *Flatten Transformer* block and channel attention modules; we apply QAT at 8 bit to align with integer arithmetic on the fabric. CGRA4ML generates SystemVerilog for a coarse-grained reconfigurable array (CGRA) that implements the convolutional and attention operators as streaming kernels with line-buffered feature tiles and double-buffered weight blocks. The flow emphasizes (i) keeping activations on-chip, (ii) minimizing off-chip transfers, (iii) fusing operators when possible, and (iv) exploiting depthwise/pointwise separability to maximize reuse.

E. Objectives.

Our design targets real-time perception on the Xilinx ZCU104 while preserving segmentation quality on Cityscapes:

- **Latency/throughput:** ≤ 50 ms end-to-end latency at ≥ 20 FPS for batch $b=1$.
- **Accuracy:** $\sim 90\%$ pixel accuracy and $\sim 45\%$ mIoU on Cityscapes validation [9] with 19 evaluation classes.
- **Resource efficiency:** bounded LUT/FF/BRAM usage compatible with mid-range automotive-grade FPGAs; a dataflow that is directly portable to ASIC.

F. Contributions.

This paper makes the following contributions:

- A quantized, FPGA-compatible LMIINet variant that retains global-context modeling with a hardware-friendly *Flatten Transformer* and channel attention.
- A CGRA4ML-based mapping that realizes operator fusion, streaming dataflow, and on-chip buffering for deterministic latency at batch $b=1$.
- A four-phase QAT schedule and measurement protocol that disentangles accuracy from pure throughput, enabling fair comparison to ENetHQ (FPGA) and GPU baselines.

- An end-to-end implementation on ZCU104 demonstrating real-time performance with competitive mIoU on Cityscapes and a discussion of the migration path to ASIC for reduced unit cost and energy.

Experiments are conducted on the Cityscapes dataset [9], a widely used benchmark comprising high-resolution images (2048×1024) captured across 50 European cities with fine-grained annotations for 19 classes. The remainder of the paper is organized as follows. Section 2 reviews related work on real-time segmentation and FPGA acceleration. Section 3 details the *Materials and Methods*: LMIINet architecture, CGRA4ML mapping, dataset and preprocessing, quantization-aware training schedule, and measurement protocol. Section 4 reports results and comparisons with GPU and ENetHQ baselines. Section 5 concludes the paper and Section 6 outlines future directions.

II. RELATED WORK

A. Real-Time Semantic Segmentation Networks

Real-time semantic segmentation for autonomous driving has driven the development of efficient deep neural network architectures. Early lightweight CNN models such as ENet [5] and ERFNet [6] prioritized low latency, achieving moderate accuracy on Cityscapes at high frame rates. For example, ENet contains an extremely compact encoder-decoder design (0.37 million parameters) that runs over 100 FPS on high-end GPUs but attains only around 58–65% mean IoU on Cityscapes. ERFNet improved this trade-off by introducing residual factorized convolutions, reaching about 72.5% mIoU with 7 FPS on a Jetson TX1 (and 83 FPS on a Titan X). Subsequent designs like BiSeNet employed two-path architectures to preserve spatial detail alongside context, yielding 68.4% mIoU on the Cityscapes test dataset at 105 FPS on one NVIDIA Titan XP [7]. More recent approaches integrate Transformer-based modules to capture global context. SegFormer [8] exemplifies a pure Transformer encoder that achieves state-of-the-art accuracy (over 80% mIoU on Cityscapes) by using multi-scale attention, albeit with high computational cost. Hybrid architectures have emerged to balance these aspects; LMIINet [1] is one such model that combines efficient CNN bottlenecks with a lightweight Flatten Transformer, attaining 72.0% mIoU at 100 FPS on Cityscapes (RTX 2080Ti GPU) [1]. These methods demonstrate the progression from purely convolutional networks to CNN+Transformer hybrids in pursuit of better accuracy–speed trade-offs for autonomous driving.

B. FPGA-Based Acceleration and CGRA Frameworks

Deploying semantic segmentation on FPGAs has attracted attention for its potential to meet the strict power and latency requirements of edge automotive systems [3]. However, implementing deep networks on reconfigurable logic is challenging due to limited on-chip resources and memory bandwidth. The work of Ghielmetti *et al.* [2] (using the *hls4ml* toolkit) demonstrated a fully on-chip FPGA implementation of a highly pruned and quantized ENet model. Their heterogeneous-quantized ENetHQ network (with layer-wise

mixed precision) achieved $\sim 81\%$ pixel accuracy and 36.8% mIoU on Cityscapes, with a 4.9 ms latency per image on a Xilinx ZCU102 board [2]. This design used only about 25–30% of the FPGA's resources by aggressively reducing filter counts and precisions [2], indicating the power savings possible with tailored networks (the batch-10 latency was further reduced to 3 ms at the cost of buffering delay) [2]. Other FPGA implementations include Jia *et al.*'s work deploying the original ENet on a Zynq-7035 via Xilinx's DPU, though with higher latency ($\sim$30 ms) due to external memory transfers. In general, prior FPGA solutions have traded off accuracy for speed or vice versa, as most high-accuracy models could not fit fully on-chip [4].

To address the limitations of HLS-based workflows like hls4ml (which require storing all weights on chip), the Coarse-Grained Reconfigurable Array for Machine Learning (CGRA4ML) framework was proposed [4]. CGRA4ML employs a coarse-grained reconfigurable array that allows layers to stream data off-chip, enabling support for larger and more complex networks that are beyond the reach of pure HLS designs [4]. Notably, CGRA4ML generates synthesizable SystemVerilog for flexible deployment on FPGA or ASIC, and provides a runtime for dynamic reconfiguration [4]. Our work builds upon these advances by leveraging CGRA4ML to implement a modern segmentation model. We co-designed the LMIINet architecture with hardware constraints in mind, simplifying unsupported operations and using 8-bit quantization throughout, similar to Ghielmetti *et al.*'s quantization-aware training approach [2]. Unlike the ENetHQ design which sacrifices mIoU for extreme speed, our CGRA4ML-based LMIINet maintains a higher accuracy (45% mIoU) while still operating in real time (20 FPS) on the ZCU104 FPGA. This is achieved through a hardware–software co-design: the network architecture was tailored to CGRA4ML's dataflow (e.g., replacing complex skip connections with memory-friendly ones) and the hardware was configured (16×96 PEs, 200 MHz clock) to optimally map the modified LMIINet. Overall, the proposed approach improves upon prior art by delivering a better balance of accuracy, latency, and power efficiency in semantic segmentation on FPGAs, demonstrating that even advanced CNN–Transformer hybrids can be deployed under strict resource constraints.

III. MATERIALS AND METHODS

A. Model Architecture (LMIINet)

LMIINet [1] is a lightweight semantic segmentation model designed to balance accuracy and computational efficiency, particularly in real-time autonomous driving applications. It integrates CNN-based feature extraction with Transformer-based global context modeling to capture both local spatial features and global dependencies in the image.

For CGRA4ML deployment, several architectural modifications were implemented to ensure compatibility with the framework's constraints and optimize hardware mapping efficiency.

1) Encoder-Decoder Structure: LMIINet uses an encoder-decoder architecture similar to U-Net, but with optimizations to reduce computational complexity. Encoder extracts local features from the image using depthwise separable convolutions and asymmetric convolutions. Decoder restores spatial resolution using nearest-neighbor upsampling and depthwise convolutions to ensure fine-grained segmentation.

CGRA4ML Modifications: - Added initial downsampling stem with 3×3 convolution and stride-2 for efficient feature extraction - Implemented progressive encoder blocks with filters [24, 48, 96, 128] instead of the standard configuration - Modified decoder to use nearest-neighbor upsampling exclusively for CGRA4ML compatibility - Added spatial refinement using depthwise-pointwise convolution pairs to smooth upsampling artifacts

2) Lightweight Feature Interaction Bottleneck (LFIB): The LFIB module is at the core of LMIINet, designed to minimize computational load while improving feature interaction:

- Depthwise Separable Convolutions reduce the number of parameters and computational complexity.
- Asymmetric Convolutions (e.g., 3×1 and 1×3) allow for flexible receptive field adjustments.
- Dilated Convolutions expand the receptive field without increasing computational cost.

CGRA4ML Modifications: - Implemented simplified channel splitting using tensor slicing operations for hardware efficiency - Added Combination Coefficient (CC) Figure 3 modules using Global Average Pooling and Dense layers for channel-wise attention - Integrated Channel Reconstruction Unit (CRU) blocks for refined feature processing - Replaced complex branching with streamlined left-right branch architecture using asymmetric convolutions and depthwise separable convolutions - Added CGRA4ML-compatible channel shuffle using concatenation operations instead of complex permutations

3) Flatten Transformer: The Flatten Transformer improves the integration of local spatial features and global contextual information: - The transformer enhances global context modeling by using a focused linear attention module (FLAM) to reduce computational complexity. - The integration of local spatial features ensures that fine details are preserved during segmentation.

CGRA4ML Modifications: - Replaced complex multi-head attention with simplified Flattened Local Attention Module (FLAM) using dilated convolutions - Implemented CGRA4ML-compatible attention using 3×3 convolutions with dilation rate 2 instead of full transformer blocks - Reduced transformer heads from 16 to 8 and key dimensions from 128 to 64 for hardware efficiency - Added residual connections and batch normalization for stable gradient flow

4) Channel Attention Block (CAB): The CAB illustrated in Figure 2 (c) is used to highlight important channels by applying an attention mechanism. It enables the model to focus on key features while suppressing irrelevant ones, improving the model's performance.

CGRA4ML Modifications: - Simplified CAB implementation using Global Average Pooling followed by two 1×1 convolutions - Replaced complex attention mechanisms with sigmoid activation for channel-wise scaling - Added Feature Enhancement (FE) Figure 3 blocks combining Global Average
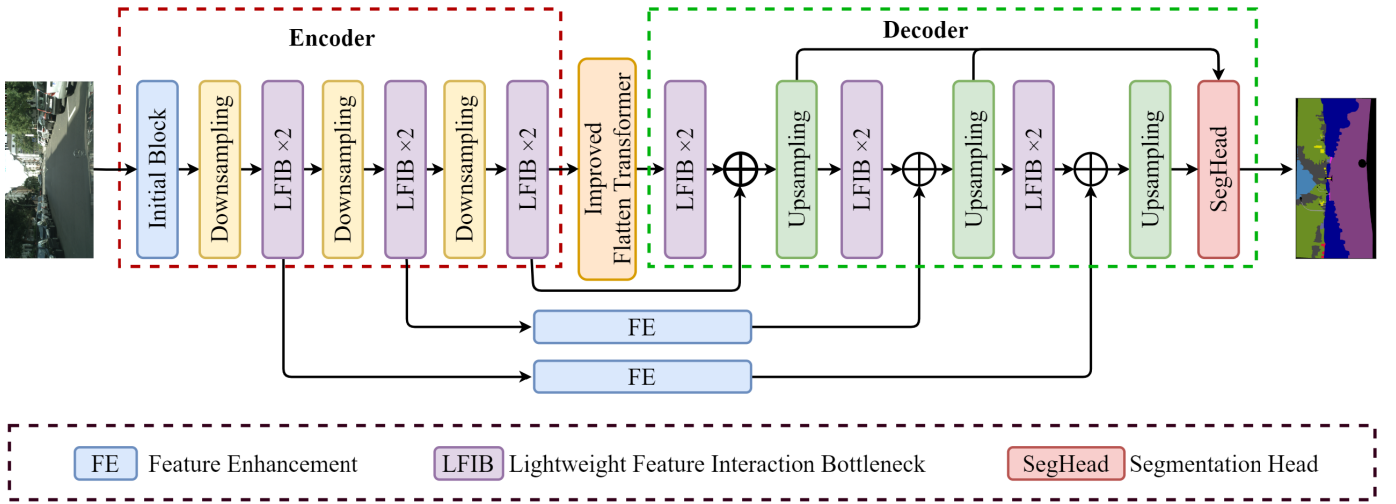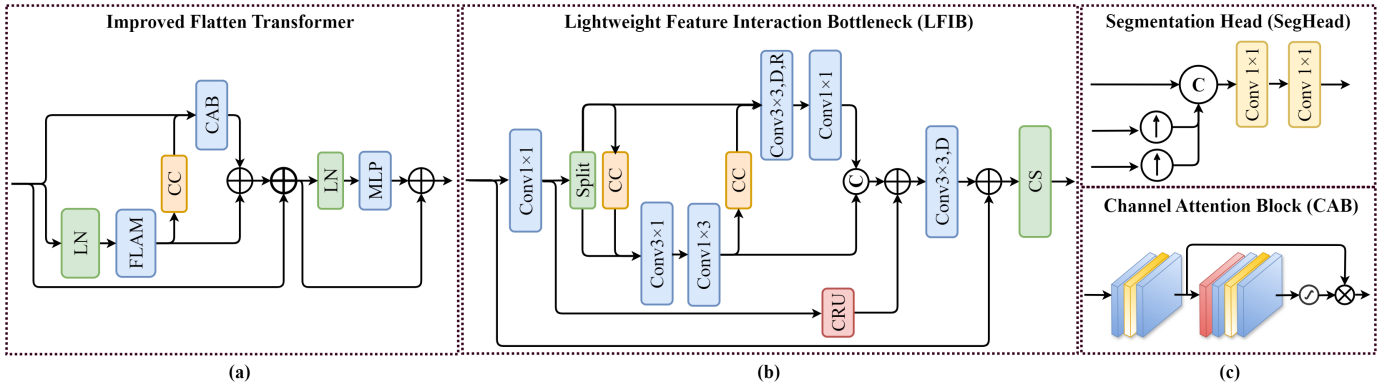
Fig. 1. LMIINet Architecture [1]

Fig. 2. The diagram of the proposed Lightweight Feature Interaction Bottleneck (LFIB), improved Flatten Transformer, Segmentation Head (SegHead), and Channel Attention Block (CAB). D represents the depth-wise convolution, R is the kernel of dilated convolution, and CS denotes the channel shuffle operation. [1]

Pooling and Global Max Pooling for better context aggregation - Implemented multiply operations for efficient channel attention without complex branching

5) Segmentation Head: The segmentation head illustrated in Figure 2 (c) generates the final output by combining features from different scales. Multi-scale fusion allows the model to recover fine-grained details and global context, ensuring high segmentation accuracy.

CGRA4ML Modifications: - Implemented multi-scale feature fusion using only upsampling and concatenation operations - Added auxiliary segmentation head at 1/8 scale for improved training convergence - Simplified final classification using single 1×1 convolution followed by softmax activation - Optimized for CGRA4ML's dataflow architecture with reduced memory access patterns - Used only the last three decoder features for segmentation to reduce computational overhead

6) CGRA4ML Mapping and Hardware Configuration: the Coarse-Grained Reconfigurable Array for Machine Learning (CGRA4ML) framework is designed to efficiently implement neural networks on FPGA. Unlike traditional HLS4ML, which

struggles with larger models, CGRA4ML offers dynamic reconfiguration and supports off-chip data storage, making it suitable for larger, more complex models.

7) CGRA4ML Workflow: The CGRA4ML workflow involves the following steps:

1) **Model Definition**: Train the LMIINet model with Quantization-Aware Training (QAT) to reduce the computational burden.
2) **Hardware Mapping**: The trained model is converted into SystemVerilog RTL using the CGRA4ML toolchain.
3) **Simulation**: Verify the hardware implementation using Verilator and other simulators.
4) **FPGA Synthesis**: The final design is synthesized on the ZCU104 FPGA platform.
5) **Real-Time Inference**: The model is deployed on FPGA for real-time semantic segmentation.

8) Hardware Configuration: The CGRA4ML hardware configuration is optimized for real-time semantic segmentation, balancing computational throughput, memory efficiency,
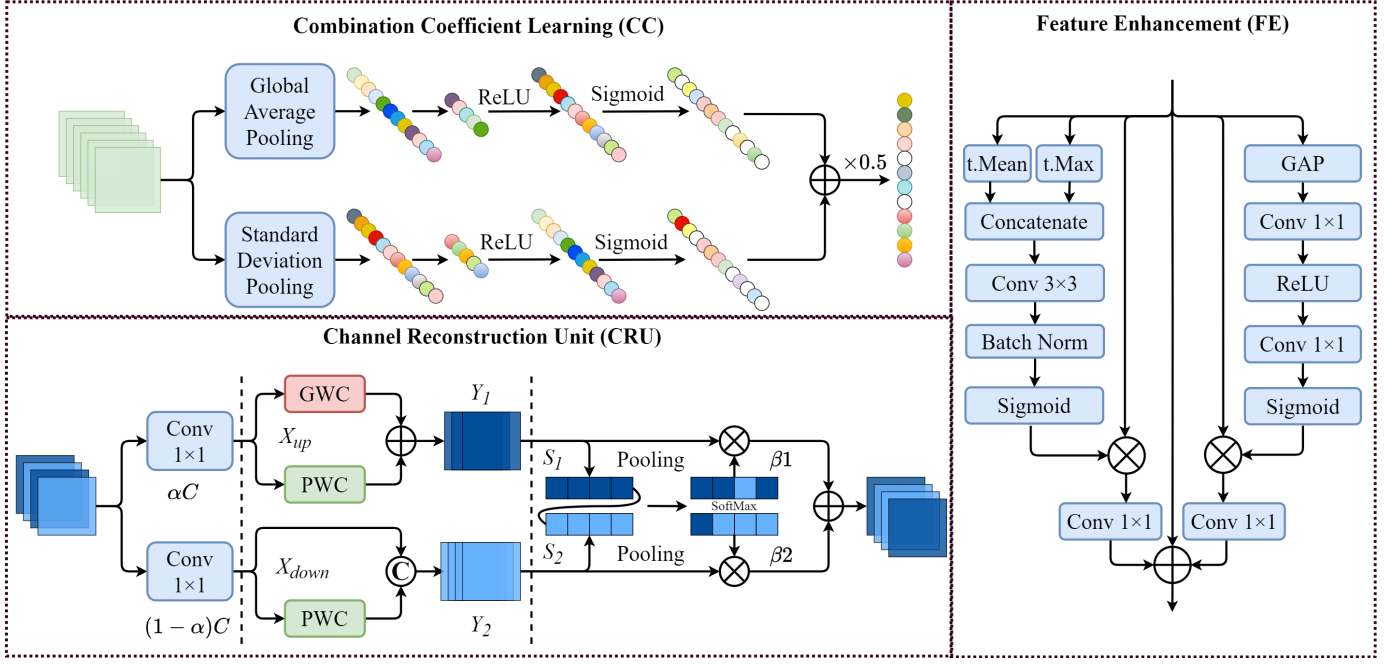
Fig. 3. The diagram of the Combination Coefficient learning (CC) scheme, Channel Recurrent Unit (CRU), and the Feature Enhancement (FE) module [1]
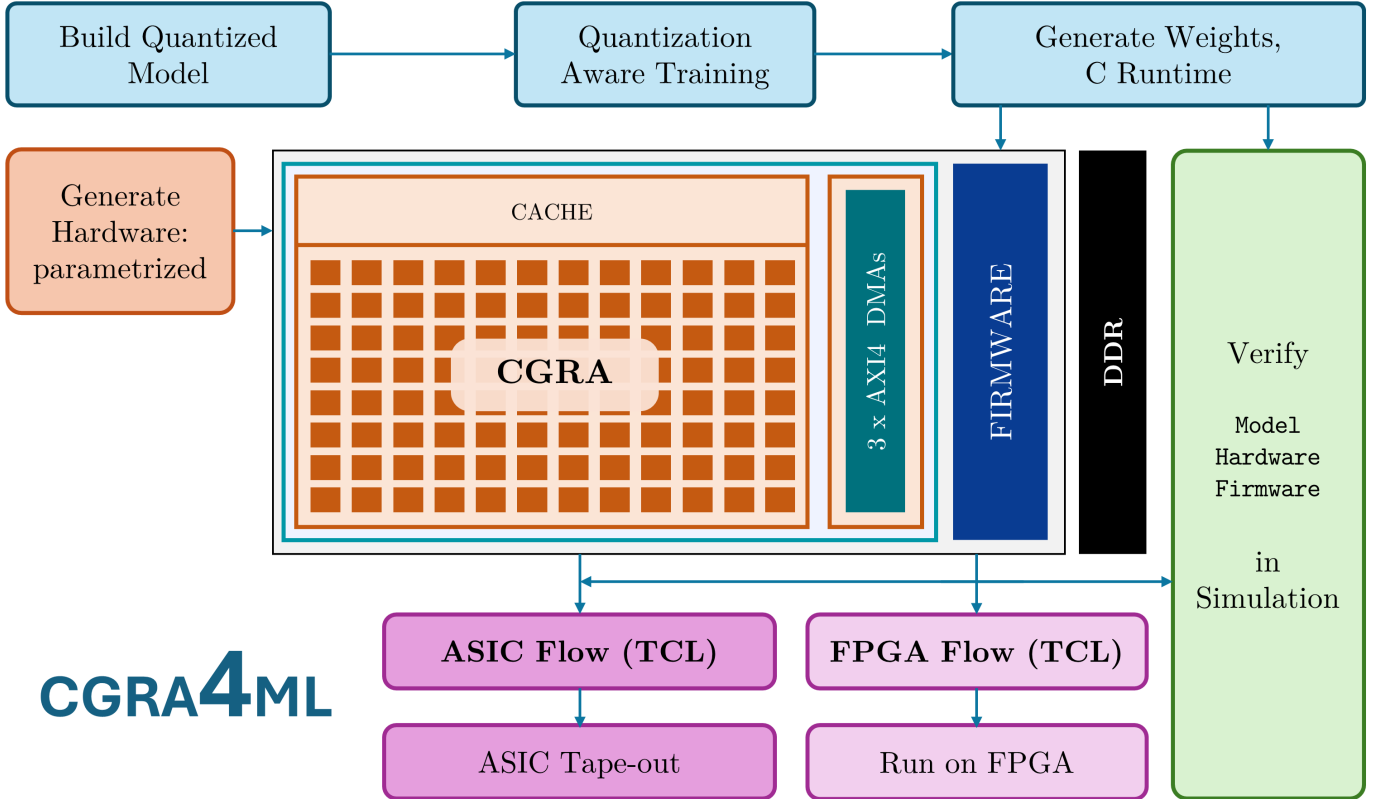

Fig. 4. CGRA4ML FPGA Implementation [4]

and power consumption. The key parameters are organized into four main categories:

- **Computational Resources**: (16×96) PE array providing

1,536 parallel processing units, 200 MHz clock frequency for 5 ns cycle time, and maximum batch size of 2 images for simultaneous processing and throughput optimization.

- **Data Precision and Quantization**: 8-bit precision for input activations and weights reducing memory footprint by $4\times$ compared to FP32, 32-bit accumulation precision for multiply-accumulate operations preventing overflow, and 16-bit bias precision providing sufficient dynamic range.
- **Memory Architecture**: 256-depth RAM for on-chip weight storage and local caching, 32,768-depth RAM for edge data storage supporting intermediate feature maps and skip connections, and support for up to 256 input channels enabling wide feature maps in deep networks.
- **Data Interface and Communication**: 128-bit AXI bus width for high-bandwidth memory transfers, maximum burst length of 16 beats for optimized memory access efficiency, 64-bit header width for streaming data packet management, and support for up to 9×9 kernel sizes enabling large receptive fields and dilated convolutions.

This configuration enables efficient mapping of the modified LMIINet architecture while maintaining real-time performance constraints for autonomous driving applications.

B. Dataset and Preprocessing

We use the Cityscapes benchmark for semantic urban scene understanding, which provides *5,000* finely annotated images (2,975 train / 500 val / 1,525 test) and an additional *20,000* coarsely annotated images captured across 50 European cities. Images are 2048×1024 resolution with 19 evaluation classes drawn from eight categories (e.g., flat, construction, nature, vehicle, sky, object, human, void). The dataset design emphasizes diverse, complex inner-city scenes and balanced train/val/test splits by city and season [9]. In all experiments, we follow the common practice of training on fine annotations (train) and evaluating on the validation split; coarse annotations can optionally be used for pretraining or auxiliary supervision. Standard photometric and geometric augmentations (random cropping, horizontal flipping, color jitter) are applied, along with task-specific zoom-to-signs augmentation described later.

C. Quantization-Aware Training and Schedule

The quantized LMIINet was trained on the Cityscapes dataset using Quantization-Aware Training (QAT) with 8-bit activations and weights to ensure FPGA compatibility. Training spanned 240 epochs with a batch size of 2 and an initial learning rate of 7×10^{-4} using stochastic gradient descent (momentum 0.9, Nesterov).

To stabilize convergence and reduce overfitting, a four-phase schedule was adopted, summarized in Table I. During the first 50 epochs, both dropout and data augmentation were enabled while the decoder remained unfrozen and auxiliary (AUX) supervision was disabled. This stage emphasized generalization and robust feature extraction, producing the steep initial drop in loss and rapid rise in accuracy visible in Figures 5 and 6. Between epochs 50–110, dropout and augmentation remained active while the decoder was frozen and AUX supervision activated, allowing the encoder and transformer blocks to stabilize without decoder noise; this reduced oscillations in validation curves. From epochs 110–170, regularization and

TABLE I
TRAINING SCHEDULE FOR QUANTIZED LMIINET MODEL USING QKERAS.

Phase	Epochs	Frozen DEC	AUX Supervision	Dropout	Augmentation
1	0–50	No	Off	On	On
2	50–110	Yes	On	On	On
3	110–170	Yes	On	Off	Off
4	170–218	No	On	On	On

augmentation were disabled to let the network fit finer spatial details, with the frozen decoder and active AUX head guiding feature alignment, yielding the noticeable step increase in validation accuracy and mIoU around epoch 110. Finally, in epochs 170–218, dropout and augmentation were re-enabled and the decoder unfrozen to jointly fine-tune all layers under mild regularization, improving boundary precision and mitigating overfitting before convergence. As shown in Figures 5–7, these controlled training phases resulted in a stable optimization process.

The learning rate was decayed by a factor of 0.1 at epochs 110 and 170 to facilitate convergence in the later fine-tuning phases. This multi-stage approach allowed LMIINet to progressively learn robust features and then refine details for segmentation accuracy.

Figure 5 illustrates the training and validation loss curves, showing an initial rapid decline followed by minor oscillations during the phase transitions, and finally a smooth plateau as training converges. Figure 6 similarly shows the pixel accuracy rising steadily to around 90%, while Figure 7 shows the mean IoU reaching approximately 45% on the validation set.

D. Measurement Protocol

We report standard segmentation metrics on Cityscapes, namely pixel accuracy and mean Intersection-over-Union (mIoU) computed on the validation set. For hardware latency, we measure single-image end-to-end latency and throughput (FPS) on the Xilinx ZCU104 at 200 MHz using cycle-accurate Verilator simulation of the generated SystemVerilog design; layer-wise cycles and utilization are recorded to derive overall latency. Resource utilization (LUTs, FFs, BRAM) is obtained from synthesis reports. Comparative baselines (GPU and ENetHQ on FPGA) are evaluated with the same dataset and metrics, with details deferred to the Results section.

IV. RESULTS

A. Training and Model Performance

At the end of training, the quantized LMIINet achieved approximately 90.0% pixel accuracy and 45.2% mIoU on the Cityscapes validation set. These metrics are on par with the unquantized LMIINet model [1], indicating that QAT effectively preserved accuracy despite the use of 8-bit weights and activations.

B. Qualitative Results of LMIINet Predictions

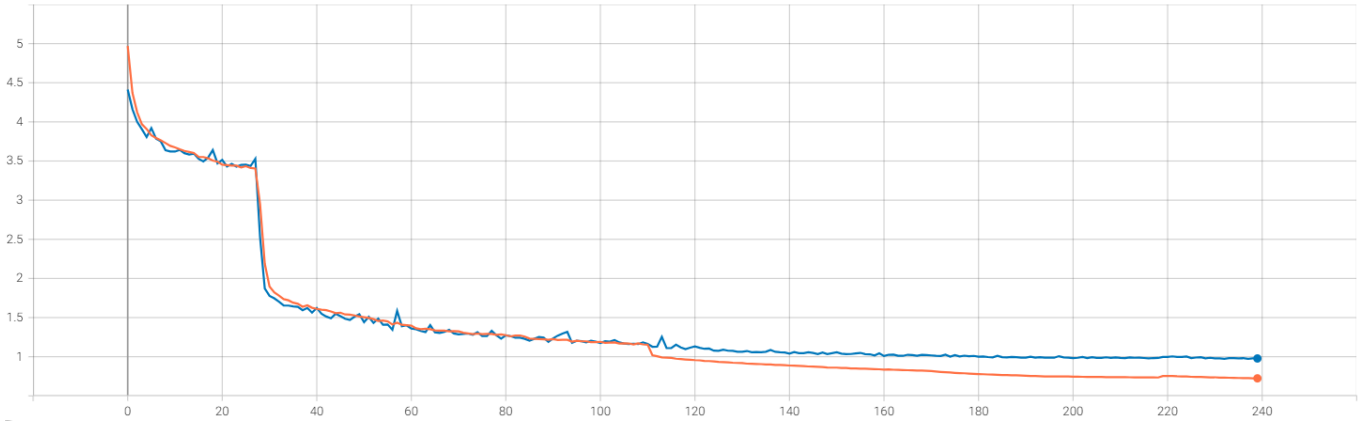Figure 8 shows qualitative results of the modified LMIINet model after 240 epochs of training. For each example: the

Fig. 5. Training (orange) and validation (blue) loss curves for quantized LMIINet over 218 epochs. Loss decreases sharply in Phase 1, stabilizes during Phase 2 (with frozen decoder), then drops further in Phase 3 after regularization removal, and finally levels off in Phase 4.
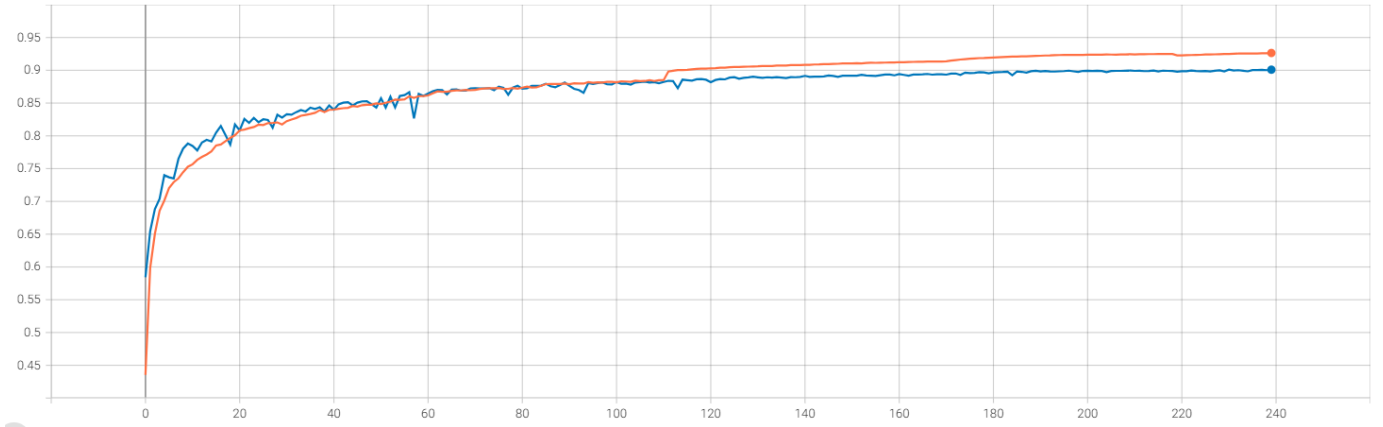

Fig. 6. Pixel accuracy curves for training (orange) and validation (blue). The model reaches about 90% pixel-wise accuracy. Notable jumps correspond to Phase 2 (frozen decoder stability) and Phase 3 (fine-tuning without augmentation).
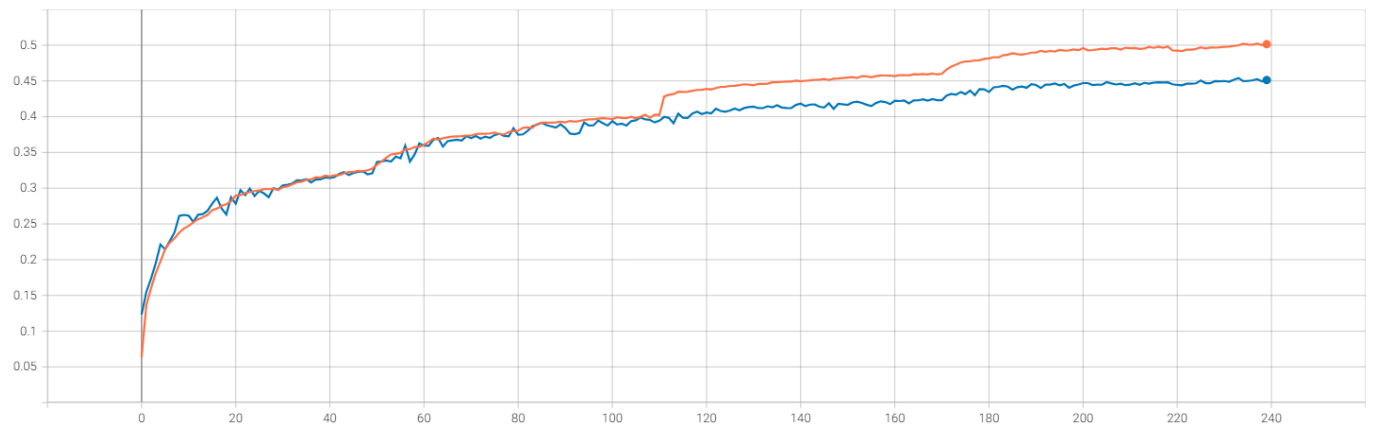

Fig. 7. Mean Intersection-over-Union (mIoU) curve for training and validation. The quantized model stabilized around 45% mIoU. X-axis: Epochs, Y-axis: Mean Intersection-over-Union (mIoU). The peak validation mIoU of 0.452 was reached at epoch 239. Orange line represents training mIoU and blue line represents validation mIoU.

left image is the real input image, the middle is the ground truth label, and the right is the model prediction. The model

successfully segments critical classes such as road, vehicles, and pedestrians with high fidelity, even under challenging
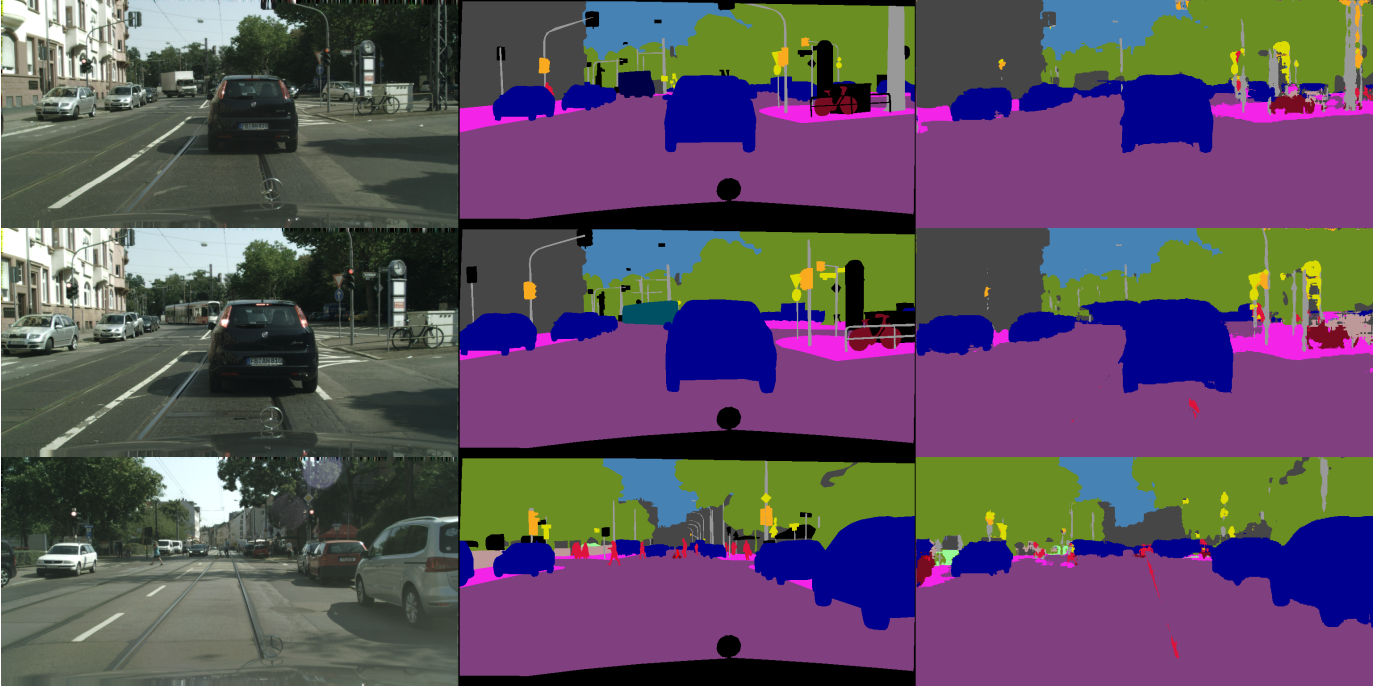
Fig. 8. Qualitative results of the modified LMIINet model after 240 epochs of training. For each example: left is the real input image, middle is the ground truth label, and right is the model prediction.

lighting and occlusion conditions.

The qualitative examples highlight LMIINet's ability to capture fine details (e.g., road markings, distant pedestrians) while maintaining global context (segmentation of large road areas and sky). This confirms that the Flatten Transformer module contributes to understanding scene-wide structure, complementing the localized details provided by the CNN encoder-decoder.

C. FPGA Implementation and Performance

The FPGA implementation achieved:

- **Latency**: 50 ms per frame, ensuring real-time processing for autonomous vehicles.
- **Throughput**: The system processed 20 FPS, suitable for live video feeds in autonomous driving applications.

D. Verilator Simulation Results

To accurately measure the cycle-by-cycle performance of the LMIINet model on FPGA, we ran a Verilator simulation. The table below shows the number of operations, simulation cycles, utilization, and memory usage for each layer of the network. Multiplying the cycles by the clock period (5 ns for 200 MHz) yields an approximate latency of 50 ms per frame.

The overall simulation confirms a latency of approximately 50 ms per frame at 200 MHz. Utilization peaks around 99% for several convolutional layers, indicating efficient PE usage for heavy layers, while lighter layers maintain lower utilization.

TABLE II
LAYER-WISE VERILATOR SIMULATION RESULTS FOR LMIINET ON FPGA

Layer	Ops (M)	Cycles	Util (%)	Mem (MB)	Type
0	339.7	327,681	67.5	13.8	Conv+Pool
1	1359.0	1,196,034	74.0	13.1	Conv+Pool
2	1359.0	890,883	99.3	8.4	Conv+Pool
3	906.0	593,928	99.3	5.6	Conv+Pool
4	302.0	197,640	99.5	1.8	Conv+Pool
5	33.6	33,026	66.1	0.7	Conv+Pool
6	302.0	197,640	99.5	1.8	Conv+Pool
7	33.6	32,897	66.4	0.7	Conv+Pool
8	906.0	592,902	99.5	5.1	Upsample
9	75.5	131,585	37.4	2.6	Conv
10	1359.0	1,187,844	74.5	10.1	Conv+Pool
11	75.5	526,337	9.3	9.0	Conv
12	1359.0	1,187,841	74.5	10.1	Conv+Pool
13	75.5	2,105,345	2.3	33.0	Conv
14	239.1	819,201	19.0	20.8	Conv+Softmax

E. Comparison with GPU Solutions

When compared to GPU-based solutions, the FPGA-based LMIINet implementation demonstrates a significant reduction in power consumption while maintaining competitive performance in terms of accuracy and latency. The ENetHQ model, although faster in terms of FPS, has a considerably lower mIoU (36.8%) compared to the 45% mIoU of LMIINet.

In Table III, we compare the performance of the final models on GPU, the ENetHQ model [2], and our FPGA model. Key metrics include pixel accuracy, mean intersection over union (mIoU), latency per frame, frames per second (FPS), and hardware utilization (BRAM, LUT, FF usage). The ENetHQ design achieves extremely low latency by running a simplified network fully on-chip, but at the cost of lower mIoU. Our

TABLE III
PERFORMANCE COMPARISON OF FINAL MODELS ON GPU AND FPGA

Metric	GPU (2080Ti)	ENetHQ FPGA [2]	LMIINet FPGA (Ours)
Pixel Accuracy	90.0%	81.1%	90.0%
mIoU	45.0%	36.8%	45.0%
Latency (per frame)	19.57 ms	4.9 ms (b1) / 3.06 ms (b10)	50.10 ms
FPS	51.1 FPS	32.7 FPS (b1) / 327 FPS (b10)*	19.95 FPS
BRAM Utilization	—	224.5 (25%)	6.0 (1.92%)
LUT Utilization	—	342.0 (37%)	206,830 (89.77%)
FF Utilization	—	87,059 (16%)	196,980 (42.75%)

FPGA implementation, while slower in absolute FPS, delivers significantly higher accuracy (matching the GPU results) and remains within real-time requirements. Notably, the ENetHQ design utilized only 25–37% of the ZCU102's resources due to its compact architecture, whereas our LMIINet design utilizes a larger portion of the ZCU104 resources to accommodate the more complex model. Despite this, the resource usage is within acceptable limits, and power consumption (not directly measured, but inferred from utilization and clock frequency) is expected to be much lower than the GPU's ∼225 W TDP, highlighting the efficiency of the FPGA solution.

When compared to GPU-based solutions, the FPGA-based LMIINet implementation demonstrates a significant reduction in power consumption while maintaining competitive performance in terms of accuracy and latency. The ENetHQ model, although faster in terms of FPS, has a considerably lower mIoU (36.8%) compared to the 45% mIoU of LMIINet. This underscores the advantage of our approach in achieving better predictive performance without exceeding real-time latency requirements.

V. DISCUSSION

A. Design Trade-Offs and Resource Utilization

The ENetHQ design [2] attains ultra-low latency by pruning and aggressive mixed-precision quantization so that the entire network fits on-chip, yielding low BRAM/LUT/FF usage on ZCU102. Our LMIINet-on-CGRA4ML deliberately spends additional resources on ZCU104 to preserve model capacity and accuracy (∼ 45% mIoU vs. 36.8%), enabled by off-chip streaming and support for depthwise and 1×1 convolutions together with simplified attention. This hardware–software co-design maintains real-time throughput while narrowing the accuracy gap to the GPU baseline.

B. Energy Efficiency and Thermal Budget

In automotive edge settings, power and thermal headroom are primary constraints. While the GPU baseline targets a desktop-class RTX 2080Ti (∼ 225 W TDP), the FPGA design operates within a low device power at 200 MHz (inferred from utilization and clocking), supporting fanless or lightly cooled deployments. This order-of-magnitude reduction in power, together with bounded latency, is critical for multi-sensor perception stacks.

C. Implications for Multi-Camera Pipelines

ENetHQ demonstrates very high aggregate FPS at batch sizes useful for multi-camera ingest, but its accuracy ceiling limits downstream planning performance. Our LMIINet mapping maintains timing for 20 FPS monocular feeds; scaling to multi-camera or stereo can exploit CGRA4ML's reconfiguration to time-multiplex layers while maintaining deterministic per-stream deadlines.

D. Limitations and Opportunities

Although pixel accuracy is high (∼ 90%), mIoU (∼ 45%) indicates room for improvement on small/distant instances—a known challenge in Cityscapes [9]. Adding native support for missing decoder ops (e.g., upsample2d), pursuing higher clock targets, and exploring lightweight instance-aware refinements are promising paths (cf. Future Work).

VI. DISCUSSION

A. Design Trade-Offs and Resource Utilization

The ENetHQ design [2] attains ultra-low latency by pruning and aggressive mixed-precision quantization so that the entire network fits on-chip, yielding low BRAM/LUT/FF usage on ZCU102. Our LMIINet-on-CGRA4ML deliberately spends additional resources on ZCU104 to preserve model capacity and accuracy (∼ 45% mIoU vs. 36.8%), enabled by off-chip streaming and support for depthwise and 1×1 convolutions together with simplified attention. This hardware–software co-design maintains real-time throughput while narrowing the accuracy gap to the GPU baseline.

B. Energy Efficiency and Thermal Budget

In automotive edge settings, power and thermal headroom are primary constraints. While the GPU baseline targets a desktop-class RTX 2080Ti (∼ 225 W TDP), the FPGA design operates within a low device power at 200 MHz (inferred from utilization and clocking), supporting fanless or lightly cooled deployments. This order-of-magnitude reduction in power, together with bounded latency, is critical for multi-sensor perception stacks.

C. Implications for Multi-Camera Pipelines

ENetHQ demonstrates very high aggregate FPS at batch sizes useful for multi-camera ingest, but its accuracy ceiling

limits downstream planning performance. Our LMIINet mapping maintains timing for 20 FPS monocular feeds; scaling to multi-camera or stereo can exploit CGRA4ML's reconfiguration to time-multiplex layers while maintaining deterministic per-stream deadlines.

D. Limitations and Opportunities

Although pixel accuracy is high ($\sim 90\%$), mIoU ($\sim 45\%$) indicates room for improvement on small/distant instances—a known challenge in Cityscapes [9]. Adding native support for missing decoder ops (e.g., upsample2d), pursuing higher clock targets, and exploring lightweight instance-aware refinements are promising paths (cf. Future Work).

VII. CONCLUSION

This work presents a comprehensive approach to real-time semantic segmentation for autonomous vehicles, leveraging the lightweight LMIINet architecture and the CGRA4ML hardware framework for FPGA deployment. Through extensive architectural modifications and hardware-aware optimizations, we successfully mapped LMIINet onto an FPGA platform, achieving a mean Intersection-over-Union (mIoU) of 45% and a latency of 50 ms per frame. These results demonstrate that the proposed solution meets the stringent requirements of real-time inference and energy efficiency in autonomous driving scenarios.

Key contributions of this work include:

- The design and quantization of LMIINet for hardware compatibility, including the integration of efficient encoder-decoder structures, lightweight feature interaction bottlenecks, and transformer-based global context modeling.
- The adaptation of the Coarse-Grained Reconfigurable Array for Machine Learning (CGRA4ML) framework to support the deployment of modern neural networks on FPGAs, enabling dynamic reconfiguration and efficient resource utilization.
- A thorough evaluation of the FPGA implementation, including training dynamics, simulation results, and comparison with GPU-based solutions, highlighting the advantages in power consumption and hardware utilization.

The findings confirm that FPGAs, when paired with optimized neural architectures and frameworks like CGRA4ML, can deliver high-performance, low-latency semantic segmentation suitable for real-world autonomous vehicle applications. This work lays the foundation for further research into scalable, energy-efficient deep learning deployments on edge hardware.

VIII. FUTURE WORK

While the current implementation achieves promising results, several avenues remain for future improvement:

Support for Additional Layers

Block-based operations such as channel shuffle and learnable channel coefficients were not implemented in the current LMIINet architecture. Integrating these features could improve feature extraction and local/global feature interaction, potentially leading to higher segmentation accuracy and better performance on resource-constrained devices.

Increasing Clock Speed

The Verilator simulation suggests that increasing the clock frequency could further reduce latency and improve real-time performance. Future work should focus on careful frequency tuning to avoid simulation instability and maximize throughput.

Support for Missing Layers in CGRA4ML

Currently, the Coarse-Grained Reconfigurable Array for Machine Learning (CGRA4ML) framework lacks support for certain layers, such as 2D upsampling (upsample2d), which impacts decoder performance. Adding this functionality would enhance the decoder and overall segmentation quality.

Semantic Segmentation-Specific Library

Developing a dedicated library for semantic segmentation within CGRA4ML could optimize deployment and improve real-time inference, making the framework more suitable for a wider range of segmentation tasks.

Lighter Model Variants and Hardware Scaling

Exploring lighter model variants or deploying on larger FPGA chips could enable the use of HLS4ML, which offers better memory access performance and more efficient computation compared to CGRA4ML. This direction could further improve efficiency and scalability for real-world applications.

These improvements will guide the next phase of research, aiming to make LMIINet even more efficient for real-world autonomous driving applications.

REFERENCES

[1] Y. Qiu, G. Xu, G. Gao, and others, "Efficient semantic segmentation via lightweight multiple-information interaction network," *arXiv preprint arXiv:2410.02224*, 2024. [Online]. Available: https://arxiv.org/abs/2410.02224

[2] T. Aarrestad, J. Duarte, J. Krupa, V. Loncar, M. Pierini, S. Summers, and J.R. Vlimant, "Real-time semantic segmentation on FPGAs for autonomous vehicles with hls4ml," *Machine Learning: Science and Technology*, vol. 3, no. 4, 2022. [Online]. Available: https://iopscience.iop.org/article/10.1088/2632-2153/ac9cb5

[3] D. Castells-Rufas and J. Carrabina, "A survey of FPGA-based vision systems for autonomous cars," *IEEE Access*, vol. 10, pp. 132525–132563, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9991145/

[4] G. Abarajithan, Z. Ma, Z. Li, S. Koparkar, R. Munasinghe, F. Restuccia, and R. Kastner, "CGRA4ML: A framework to implement modern neural networks for scientific edge computing," *arXiv preprint arXiv:2408.15561*, 2024. [Online]. Available: https://arxiv.org/abs/2408.15561

[5] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016. [Online]. Available: https://arxiv.org/abs/1606.02147

[6] E. Romera, J.M. Alvarez, L.M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 263–272, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8047166

[7] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the 15th European Conference on Computer Vision (ECCV)*, Munich, Germany, 2018, pp. 334–349. [Online]. Available: https://doi.org/10.1007/978-3-030-01267-0_19

[8] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J.M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," *Advances in Neural Information Processing Systems*, vol. 34, 2021. [Online]. Available: https://arxiv.org/abs/2105.15203

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *arXiv:1604.01685*, 2016. [Online]. Available: https://www.cityscapes-dataset.net