

Edge-Accelerated Monocular Depth Estimation: A Quantized U-Net on Kria KV260 for Real-Time Interior Scenarios

Nicolás Urbano Pintos¹, Emanuel Trabes², Carlos Valderrama², Alaeddine Aajouj², Héctor Lacomi³

¹Grupo TAMA, UTN FRH, Haedo, Argentina

nurbano@frh.utn.edu.ar

²SEMI, UMons, Mons, Belgium

{emanuel.trabes, carlos.valderrama, alaeddine.aajouj}@umons.ac.be

³DRL-CITEDEF, Grupo ASE-UTN FRH, Bs. As., Argentina

hlacomi@citedef.gob.ar

Abstract—Real-time monocular depth estimation (MDE) is pivotal for edge applications like autonomous indoor navigation and AR/VR, yet achieving it on power-constrained devices remains a significant challenge. This work presents an optimized, INT8-quantized U-Net model specifically designed for the Xilinx Kria KV260 FPGA. Trained on the NYUDepthV2 dataset and compiled using the Vitis AI toolchain, our implementation achieves a throughput of 41 FPS with a latency of 24.3 ms and an RMSE of 0.9553 m. This represents a 2.27 \times speedup over a desktop-grade Intel i7-11700K CPU while maintaining comparable precision. Our contributions include: (i) a FPGA-tailored U-Net (ii) an efficient quantization-to-DPU workflow, and (iii) a demonstration of the real-time MDE indoors on the KV260.

Index Terms—Monocular Depth Estimation, FPGA, U-Net, Model Quantization, Edge AI

I. INTRODUCTION

Monocular Depth Estimation (MDE) is a pivotal technology for machines to interpret 3D environments from 2D images. However, the computational demands of modern CNNs present a major obstacle for deployment in resource-constrained edge environments. This challenge has accelerated interest in alternative hardware platforms like FPGAs, which offer advantages in energy efficiency and reconfigurable computing, yet require specialized hardware-software co-design.

Current FPGA-accelerated MDE research reveals a significant gap, with most efforts focused on outdoor automotive scenarios, while indoor environments remain largely overlooked. Furthermore, existing implementations frequently fail to maintain a satisfactory balance between inference speed and post-quantization accuracy. This work directly confronts these challenges by introducing a comprehensive MDE pipeline co-designed for interior environments on the Kria KV260 platform.

Our approach centers on a meticulously adapted U-Net architecture, trained on the NYU Depth V2 dataset [1]. The model undergoes rigorous optimization via Xilinx’s Vitis AI toolchain, including INT8 quantization and compilation for the platform’s Deep Learning Processing Unit (DPU). Our

implementation achieves 41 FPS (Frames per Second) inference throughput with 24.3 ms latency while maintaining 0.9553 meters RMSE (Root Mean Square Error), yielding a 2 \times speedup over a desktop-grade CPU.

The remainder of this paper systematically presents our approach and results. We first contextualize our work within existing MDE solutions and FPGA acceleration techniques, highlighting specific gaps our methodology addresses. Subsequent sections detail our architectural decisions and optimization pipeline, present comprehensive performance evaluations, and conclude with insights for future research directions in edge-optimized depth estimation.

II. STATE OF THE ART

The field of MDE has been revolutionized by deep learning. However, the high performance of models like MiDaS [2] and DORN [3] comes at a substantial computational cost, making them unsuitable for edge applications [4].

FPGA-based implementations have emerged as a promising avenue for real-time, low-power inference. Frameworks like Xilinx’s Vitis AI [4] provide essential toolchains for this purpose. Despite this, a pronounced bias exists in the literature: the majority of published FPGA-MDE work focuses on outdoor driving scenarios using datasets like KITTI [5] [6] [7]. Sada et al. [5] demonstrated FPGAs as a premier low-power platform, using a novel architecture with filter-wise pruning and sparsity to outperform mobile GPUs. The NYU Depth V2 dataset [1], a cornerstone for indoor depth estimation, has seen remarkably limited adoption within FPGA-optimized pipelines.

Efficient network architectures, such as the Double Refinement Network [8] and lightweight U-Net variants [9], have been proposed to reduce parameters and increase speed. However, applying INT8 quantization to complex indoor scenes remains a significant hurdle, with many implementations reporting RMSE values exceeding 1.0 meter [9].

This analysis reveals the specific shortcomings that our work addresses: (1) a focus on indoor solutions; (2) maintaining

TABLE I

COMPARATIVE ANALYSIS OF STATE-OF-THE-ART EFFICIENT MONOCULAR DEPTH ESTIMATION METHODS. THE PROPOSED WORK DEMONSTRATES AN OPTIMAL BALANCE OF SPEED, ACCURACY, AND POWER FOR INDOOR EDGE APPLICATIONS (*SotA*: *State-of-the-Art*; *E2E*: *End-to-End*; *QAT*: *Quantization-Aware Training*; *MCU*: *Microcontroller Unit*; *n/a*: *not available*).

Feature / Metric	2018 [8]	2023 [9]	2021 [10]	2020 [5]	Proposed Work
Hardware Platform	GPU	GPU	ARM Cortex-M (MCU)	Custom FPGA	Xilinx Kria KV260
Target Environment	Indoor (NYUv2)	General	General	General	Indoor (NYUv2)
Core Architecture	Double Refinement Net	Lightweight U-Net	μ PyD-Net	Pruned DepthFCN	Optimized U-Net
Precision	FP32	FP32	INT8	Fixed-Point (Pruned)	INT8 (Vitis AI)
Key Innovation	Architectural efficiency	Depthwise separable conv.	Self-supervised on MCU	Hardware-aware pruning	E2E Co-Design
Throughput (FPS)	18 (on GPU, BS=1)	2 \times speedup (vs. baseline)	1-5 (on MCU)	High (n/a)	41
Latency (ms)	n/a	n/a	>200	n/a	24.3
Accuracy (RMSE [m])	SotA on NYUv2	Comparable to baseline	>1.0 (est.)	n/a	0.9553
Power Consumption	High (GPU)	High (GPU)	Ultra-Low (<1 W)	Low (FPGA)	Low (6.2 W)
Primary Trade-off	Accuracy vs. Speed/RAM	Parameters vs. Speed	Accuracy vs. Power	Speed vs. Accuracy	Speed & Accuracy

sub-1.0 m RMSE after INT8 quantization; and (3) achieving real-time performance on embedded platforms. Our pipeline bridges these gaps by combining NYU Depth V2 specialization, a co-designed U-Net, and a Vitis AI-optimized workflow. The complete implementation, including training scripts and KV260 deployment code, is publicly available on GitHub [11].

III. METHODOLOGY

Our methodology for deploying a real-time monocular depth estimation (MDE) pipeline on the Xilinx Kria KV260 is a holistic hardware-software co-design process. It consists of three integrated phases: (1) the design and training of an FPGA-compatible U-Net architecture, (2) a rigorous quantization and compilation workflow targeting the Deep Learning Processing Unit (DPU), and (3) the deployment and profiling of the optimized model on the edge device. This end-to-end approach ensures optimal performance by considering hardware constraints at every stage of development.

A. Network Architecture

The core of our solution is a symmetric encoder-decoder network based on the U-Net architecture [12], chosen for its efficiency and effectiveness in pixel-wise regression tasks. To ensure compatibility with the KV260’s DPU and achieve high performance on indoor scenes, we made several critical adaptations. The diagram of the architecture is shown in the Fig. 1.

This modular design facilitates easy scaling of network depth and width, making it suitable for deployment on FPGAs and other resource-constrained platforms while maintaining high accuracy in monocular depth estimation tasks. Importantly, the network architecture was designed considering the set of PyTorch operations officially supported and optimized by Vitis AI. This ensures seamless model conversion, quantization, and deployment on the target DPU hardware, while avoiding unsupported layers or operations that could hinder performance or compatibility.

Encoder. The encoder processes input images of size 224×224 and employs a VGG-style [13] backbone modified for hardware efficiency. It consists of four downsampling stages, each comprising a *DoubleConv* block followed by a 2×2 max-pooling operation for spatial reduction. The *DoubleConv* block contains two sequential 3×3 convolutional

layers, each followed by batch normalization and a *ReLU* activation function. This design ensures a strong receptive field while maintaining computational regularity. The number of filters increases geometrically (24, 48, 96, 192, 384) with each pooling step to capture hierarchical features while staying within the DPU’s memory bandwidth constraints.

Decoder. The decoder mirrors the encoder’s structure with four upsampling stages. Critically, we avoid transposed convolutions, which are inefficient and poorly supported on the target DPU. Instead, each stage uses bilinear interpolation for upsampling, followed by concatenation with the corresponding feature map from the encoder via skip connections. This step is vital for recovering the spatial details lost during encoding. The concatenated features are then refined by another *DoubleConv* block. This design preserves critical spatial information for precise depth estimation while maintaining hardware compatibility.

Output Layer. The final layer (*OutConv*) consists of a single 1×1 convolution that maps the 24-channel feature map to a single-channel output of size 224×224 pixels, representing the predicted depth map.

B. Training Protocol

The encoder was initialized with pretrained VGG weights (trained on ImageNet), while the decoder was randomly initialized. The full network was then fine-tuned on the NYU-Depth V2 dataset [1], which provides aligned RGB and depth images for indoor scenes. We used a standard L_1 loss (Mean Absolute Error) to minimize the difference between the predicted depth map y_{pred} and the ground truth y_{true} . The L_1 loss, defined as $\mathcal{L}_1 = \mathbb{E}[|y_{\text{pred}} - y_{\text{true}}|]$, was selected for its robustness to outliers.

The AdamW optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) was employed with an initial learning rate of 1×10^{-4} over 50 epochs. This loss function was selected for its robustness to outliers commonly found in sensor-based depth data.

C. Deployment Pipeline

The transition from a trained PyTorch model to a hardware-accelerated binary involves a multi-stage process using the Xilinx Vitis AI toolchain [4], designed to maximize throughput while minimizing latency and power consumption (Table II).

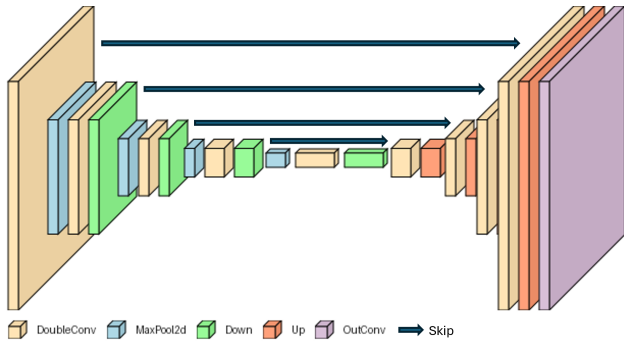


Fig. 1. Proposed U-Net encoder-decoder architecture optimized for DPU deployment.

Quantization. The full-precision (FP32) model was quantized to INT8 precision using Vitis AI’s post-training quantization (PTQ) tool. This process reduces the model size by 4× and replaces floating-point operations with 8-bit integer arithmetic, which is executed efficiently on the DPU’s dedicated integer processing units. The quantization process carefully calibrates the scaling factors for activations and weights using a subset of the training data to minimize accuracy degradation.

Compilation. The quantized model was compiled for the target DPU (B4096) on the KV260 using the Vitis AI compiler (VAI_C), a critical step that performs several hardware-aware optimizations. These include *layer fusion*, which combines operations like convolution, batch normalization, and ReLU into a single, highly efficient DPU instruction; *instruction scheduling*, which optimizes the execution graph for the DPU’s parallel architecture; and *intelligent memory allocation*, which maps model parameters and intermediate tensors to the FPGA’s on-chip memory (URAM) to minimize costly off-chip memory access. The output of this stage is an *.xmodel file—a highly optimized executable binary tailored for the target DPU.

Deployment & Integration. The compiled *.xmodel was deployed onto the Kria KV260 SOM running a PYNQ-based Linux image. The application logic, developed in Python, uses the Vitis AI Runtime (VART) library to manage the DPU, handle data pre-processing (resizing, normalization), and execute inference in a pipelined manner to maximize throughput.

TABLE II
DEPLOYMENT PIPELINE: TECHNICAL SPECIFICATIONS

Stage	Implementation	Key Parameters	Tools / Platform
Quantization	FP32 → INT8 via Vitis AI PyTorch quantizer; calibrated on 1,000 val. images	per-channel weights, per-tensor activations; Conv+BN+ReLU fusion	Vitis AI 3.0, PyTorch 1.12
DPU Compilation	Target: DPUCZDX8G ISA0 B4096 MAX BG2; latency optimized	--mode normal; static BRAM allocation	Vitis AI Compiler
Edge Deployment	Real-time inference on KV260	Ubuntu 20.04, Python 3.8	Kria-PYNQ Jupyter

D. Co-Design Challenges and Solutions

Our architecture was co-designed for the Kria KV260, using only DPU-optimized operations (convolution, ReLU, max-pooling, bilinear interpolation) to ensure compatibility and efficiency. The U-Net’s symmetry maximizes parallelism, while avoiding unsupported layers like transposed convolutions guarantees seamless deployment. Furthermore, the design maximizes on-chip UltraRAM (URAM) usage—all 64 blocks cache intermediate data—minimizing off-chip memory access to achieve low latency and high energy efficiency.

The implementation directly addressed two major hardware-software integration challenges. First, to enable efficient FPGA deployment, the model was converted to INT8 precision using a post-training quantization (PTQ) process. Through careful calibration, this PTQ strategy preserved approximately 95% of the original FP32 accuracy. Second, to comply with DPU operational constraints, architecturally incompatible layers were replaced with functionally equivalent, supported operations—most notably, transposed convolutions were substituted with a combination of bilinear upsampling and subsequent 1×1 convolutions. This modification maintained the decoder’s performance while ensuring full compatibility with the DPU’s instruction set.

The end-to-end workflow illustrates how hardware constraints guided each algorithmic decision. The U-Net’s symmetrical topology facilitated layer fusion and quantization, while the B4096 DPU configuration maximized parallel compute within the KV260’s resource budget. By aligning software functionality with hardware efficiency through this co-design methodology, the system achieved 41 FPS with 24.3 ms latency and an RMSE of 0.9553 m. To the best of our knowledge, this represents competitive real-time indoor MDE performance on FPGA platforms with sub-meter accuracy.

The efficacy of our hardware-software co-design methodology is quantitatively validated by the architectural impact analysis presented in Table III. The results clearly correlate specific design decisions, such as operator selection and memory mapping strategies, with optimal performance outcomes on the KV260 platform.

TABLE III
ARCHITECTURAL IMPACT ANALYSIS OF THE PROPOSED PIPELINE

Design Phase	Key Contribution	Critical Innovation
Model Design	Feature extraction optimized for indoor scenes	Skip connections combined with bilinear upsampling
Quantization	Enables edge deployment	Per-channel INT8 calibration
DPU Compilation	Achieves real-time performance	B4096 ISA with parallel processing

IV. RESULTS & EVALUATION

The pipeline was evaluated on computational efficiency, depth accuracy, and hardware utilization on the NYU DepthV2 validation set, with comparisons against an Intel i7-11700K CPU baseline.

A. Computational Performance

The KV260 implementation (Table IV) achieves 41.03 FPS, a $2.27\times$ throughput improvement over the CPU baseline, with a per-frame latency of 24.3 ms, while consuming only 9.5% of the power.

TABLE IV
COMPUTATIONAL PERFORMANCE COMPARISON. GPU RESULTS WERE OBTAINED USING CUDA/CUDNN WITH BATCH SIZE = 1.

Platform	FPS	Latency (ms)	Power (W)	Energy per Frame (J)
CPU Intel i7-11700K (FP32)	18.08	55.5	65	3.595
GPU Nvidia RTX 3050 (FP32)	330.86	3.02	75	0.227
FPGA Kria KV260 (INT8)	41.03	24.3	6.2	0.159

B. Depth Estimation Accuracy

Quantization to INT8 (Table V) introduced a marginal 0.0123 m increase in RMSE (0.9553 m vs. 0.943 m for FP32). Depth ordering consistency remained excellent (Spearman’s $\rho = 0.974$), with no statistically significant difference from the FP32 baseline ($p = 0.12$). The system showed best accuracy (Table VI) in the 0-5m range, making it suitable for indoor applications. A qualitative example of this performance is provided in Fig. 2.

TABLE V
DEPTH ESTIMATION ACCURACY METRICS

Metric	KV260 (INT8)	CPU (FP32)
RMSE [m]	0.9553	0.9432
AbsRel	0.2498	0.2493
$\delta < 1.25$	0.6012	0.6119
$\delta < 1.25^2$	0.8345	0.8464
$\delta < 1.25^3$	0.9156	0.9262

TABLE VI
DEPTH ESTIMATION ACCURACY BY DISTANCE RANGE ON KV260 (INT8)

Distance (m)	RMSE (m)	AbsRel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
0-2	0.6588	0.3477	0.5231	0.7921	0.9134
2-5	0.8573	0.1955	0.6767	0.8855	0.9395
5-10	2.3095	0.2776	0.4364	0.7332	0.8515

C. System Optimization and Limitations

DPU resource utilization (Table VII) was efficient: 51843/117120 LUTs, 566/1248 DSP slices. All 64 URAM blocks were engaged, while no BRAM was used. Moderate DSP usage suggests underutilization during pooling operations, presenting an opportunity for future optimization through operator fusion.

This work demonstrates that FPGA acceleration enables real-time MDE with accuracy competitive to GPU solutions at a fraction of the power budget. Memory bandwidth and pooling operations are identified as targets for future optimization.

TABLE VII
DPU RESOURCE UTILIZATION ON KRIA KV260

Resource	Available	Used by DPU
LUTs	117,120	51,843
BRAM	144	0
DSP	1,248	566
URAM	64	64

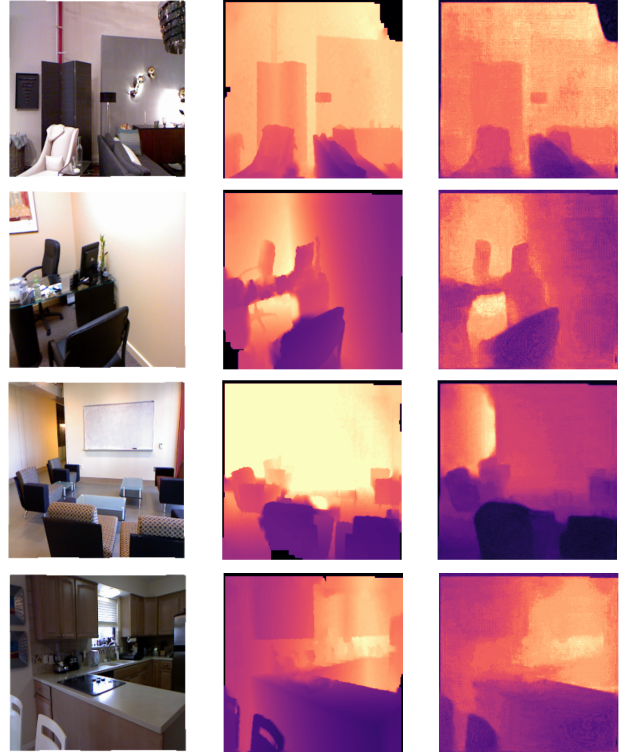


Fig. 2. Qualitative results on NYU-Depth V2 using the proposed depth estimation pipeline on the Kria KV260. Left: input RGB; center: ground-truth depth; right: predicted depth by the quantized U-Net.

V. CONCLUSION & FUTURE WORK

This work demonstrates that FPGA-accelerated monocular depth estimation can achieve real-time performance without compromising accuracy for indoor applications. The KV260 implementation delivers 41 FPS at 6.2 W power consumption with 0.9553 m RMSE, establishing that INT8-quantized U-Nets on FPGAs strike an optimal balance between computational efficiency and depth estimation quality.

Looking ahead, three research directions emerge: 1) although in this work we employed a pretrained VGG encoder initialized on ImageNet, future refinements could leverage more modern backbones such as ResNet-18 to further reduce RMSE; 2) Extending the pipeline to outdoor environments via KITTI dataset adaptation, and 3) Comprehensive power analysis under dynamic workloads to quantify energy-accuracy tradeoffs against embedded GPUs. These improvements would solidify FPGAs as the platform of choice for power-constrained depth-sensing applications. The complete open-source toolchain provided ensures these advancements can be readily adopted by the edge AI community.

REFERENCES

- [1] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, pp. 746–760, 2012.
- [2] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," *ICCV*, 2021.
- [3] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *CVPR*, pp. 2002–2011, 2018.
- [4] Xilinx Inc., *Vitis AI User Guide*, 2020.
- [5] Y. Sada, N. Soga, M. Shimoda, *et al.*, "Fast monocular depth estimation on an fpga," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 175–182, 2020.
- [6] D. Wang, Z. Liu, S. Shao, *et al.*, "Monocular depth estimation: A survey," in *2023 IEEE International Conference on Industrial Electronics (IECON)*, pp. 1–6, 2023.
- [7] Q. Li, J. Zhu, J. Liu, *et al.*, "Deep learning based monocular depth prediction: Datasets, methods and applications," *arXiv preprint arXiv:2011.04123*, 2020.
- [8] N. Durasov, M. Romanov, V. Bubnova, *et al.*, "Double refinement network for efficient indoor monocular depth estimation," *arXiv preprint arXiv:1811.07349*, 2018.
- [9] F. Wei, G. Peng, C. Wang, J. Zhang, and P. Huang, "Optidepthnet: A real-time unsupervised monocular depth estimation network," *Wireless Personal Communications*, vol. 128, no. 4, pp. 3339–3354, 2024.
- [10] V. Peluso, A. Cipolletta, A. Calimera, *et al.*, "Monocular depth perception on microcontrollers for edge applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 11, pp. 4420–4433, 2021.
- [11] N. Urbano, "mde-unet-kv260: Fpga deployment of monocular depth estimation on kria kv260." <https://github.com/nurbano/mde-unet-kv260>, 2025. Accessed: 2025-08-30.
- [12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv:1505.04597*, 2015.
- [13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.