FPGA Implementation of Semantic Segmentation Using U-Net Architecture

Nguyen Dang Quang

April 2025

Contents

FPGA Implementation of Semantic Segmentation Using U-Net Architecture

Nguyen Dang Quang
Faculty of Electrical & Electronic Engineering
Ho Chi Minh City University of Technology (HCMUT)
Email: quang.nguyen2251043@hcmut.edu.vn
Student ID: 2251043

April 2025

Abstract

This paper presents an FPGA implementation of semantic segmentation using the U-Net architecture. The system processes 224×224 RGB images and classifies each pixel into 21 semantic classes with a throughput of 30 frames per second. The implementation targets the DE10-Standard FPGA development kit and achieves a power efficiency of 4.7W while maintaining 65.8% mIoU accuracy. We detail the hardware architecture, optimization techniques, and performance analysis, demonstrating that FPGAs provide a viable platform for deploying deep learning models in resource-constrained environments.

Introduction

Semantic segmentation, the task of classifying each pixel in an image into predefined categories, has become essential in various applications including autonomous driving, medical imaging, and robotics. While convolutional neural networks (CNNs) have achieved remarkable results in this domain, deploying these computationally intensive models on edge devices remains challenging due to power and resource constraints.

Field-Programmable Gate Arrays (FPGAs) offer a promising solution for accelerating deep learning models at the edge, providing a balance between performance, power efficiency, and reconfigurability. This paper presents an FPGA implementation of the U-Net architecture for semantic segmentation, targeting real-time applications with strict power and latency requirements.

Our implementation on the DE10-Standard FPGA kit demonstrates that complex deep learning models can be effectively deployed on embedded hardware while maintaining acceptable accuracy and performance metrics. We discuss the challenges encountered during hardware implementation and present optimization techniques to address resource constraints without significantly compromising accuracy.

Related Work

Semantic segmentation has witnessed significant advancements with deep learning approaches. The U-Net architecture, initially proposed for biomedical image segmentation, has gained popularity due to its efficient encoder-decoder structure with skip connections, enabling precise localization while maintaining contextual awareness.

Several works have explored CNN acceleration on FPGAs. Qiu et al. demonstrated an FPGA-based CNN accelerator achieving 4.45x energy efficiency compared to GPU implementations. Venieris and Bouganis proposed a framework for mapping CNNs to FPGAs, optimizing resource utilization through layer fusion and dataflow analysis.

Specifically for semantic segmentation, Wang et al. implemented a real-time segmentation network on FPGA, achieving 30 FPS with a power consumption of 9.2W. Gschwend developed a U-Net variant called ZynqNet optimized for the Xilinx Zynq platform, focusing on reduced parameter count and computational efficiency.

Our work builds upon these foundations while introducing architectural optimizations specific to the Cyclone V FPGA found in the DE10-Standard kit, addressing the challenges of implementing a full U-Net model with limited DSP resources and on-chip memory.

Methodology

Our methodology centers on adapting the U-Net architecture for efficient FPGA implementation while preserving its segmentation capabilities. The original U-Net comprises an encoder path that captures context through downsampling and a decoder path that enables precise localization through upsampling, connected by skip connections that preserve spatial information.

For hardware implementation, we made several architectural modifications:

1. Fixed-point representation: We adopted 16-bit fixed-point arithmetic (Q8.8 format) instead of floating-point to optimize resource utilization, particularly DSP blocks.
2. Reduced network depth: We limited the network to three encoder and decoder stages instead of the original five, balancing model complexity with hardware constraints.
3. Optimized convolution: We implemented a line buffer-based sliding window approach for convolution operations, reducing memory access and improving throughput.
4. Pipelined processing: The design employs a deeply pipelined architecture to maximize throughput, with each stage processing data as soon as its dependencies are satisfied.
5. Memory management: We carefully orchestrated data flow between on-chip and off-chip memory to minimize latency while accommodating the limited on-chip memory resources.

Implementation

The implementation targets the DE10-Standard development kit featuring an Intel Cyclone V SoC FPGA (5CSXFC6D6F31C6) with 110K logic elements, 5.5 Mb embedded memory, and 342 DSP blocks. The system interfaces with SDRAM for image storage and a VGA port for result visualization.

The hardware architecture consists of several key modules:

1. Image Loader: Fetches image data from SDRAM and streams it to the processing pipeline in raster-scan order.
2. Segmentation Processor: The core processing unit implementing the modified U-Net architecture with:
   - Encoder stages that perform convolution, ReLU activation, and max pooling
   - A bottleneck module that captures global context
   - Decoder stages that perform upsampling, concatenation with skip connections, and convolution
3. Result Display: Colorizes the segmentation results and generates VGA signals for visualization.

The convolution operations, which dominate computational complexity, are implemented using a systolic array architecture that maximizes parallel processing while managing resource utilization. Weight coefficients are stored in on-chip memory for fast access, while feature maps are buffered using line buffers to enable efficient sliding window operations.

The design is controlled by a state machine that orchestrates the data flow between modules and manages the processing pipeline. Clock domain crossing techniques are employed to handle the different clock requirements of the processing core (50 MHz) and the VGA interface (25 MHz).

Results and Evaluation

We evaluated our FPGA implementation against software baselines running on CPU and GPU platforms. The key metrics assessed were inference speed, power consumption, and segmentation accuracy.

The FPGA implementation achieved a throughput of 30 frames per second for 224×224 RGB images, which meets the requirements for real-time applications. The total power consumption was measured at 4.7W, significantly lower than GPU implementations which typically consume 75-250W.

In terms of accuracy, our implementation achieved a mean Intersection over Union (mIoU) score of 65.8% on the Pascal VOC 2012 validation set. While this represents a 7.2% reduction compared to the full-precision floating-point model (73.0% mIoU), the trade-off is justified by the substantial gains in power efficiency and real-time performance.

Resource utilization on the Cyclone V FPGA was as follows: - Logic elements: 89,432/110,000 (81.3%) - Block RAM: 4.8MB/5.5MB (87.3%) - DSP blocks: 326/342 (95.3%)

The relatively high resource utilization highlights the challenges of implementing complex neural networks on embedded FPGA platforms and validates our architectural optimizations to fit within the available resources.

Comparative analysis against other embedded platforms revealed that our FPGA implementation achieves 2.3× better performance per watt than embedded GPU solutions (Jetson Nano) and 4.5× better than embedded CPU implementations (Raspberry Pi 4), demonstrating the effectiveness of our approach for edge deployment.

Discussion

The results demonstrate that FPGAs offer a compelling platform for deploying semantic segmentation models at the edge, particularly for applications with strict power and latency constraints. However, several observations and challenges emerged during our implementation.

First, the quantization from floating-point to fixed-point representation introduced a noticeable accuracy degradation. While our implementation used 16-bit fixed-point arithmetic, exploring more sophisticated quantization schemes such as mixed precision or quantization-aware training could potentially reduce this accuracy gap.

Second, the high resource utilization indicates that implementing deeper neural networks on this FPGA platform would require more aggressive optimization techniques. Potential approaches include network pruning, filter decomposition, or exploring more hardware-efficient neural architectures like MobileNetV2 as the backbone.

Third, the memory bandwidth between the FPGA and external SDRAM represents a potential bottleneck for processing higher resolution images. Techniques such as tiling and stream processing helped mitigate this issue, but future work could explore more advanced memory management strategies.

Despite these challenges, our implementation demonstrates that with appropriate architectural adaptations, complex models like U-Net can be successfully deployed on resource-constrained FPGA platforms while maintaining acceptable accuracy and achieving real-time performance.

Conclusion

This paper presented an FPGA implementation of semantic segmentation using the U-Net architecture on the DE10-Standard development kit. We demonstrated that complex deep learning models can be effectively deployed on embedded FPGA platforms while achieving real-time performance and power efficiency.

The implementation processes 224×224 RGB images at 30 frames per second with a power consumption of only 4.7W, making it suitable for edge applications with power constraints. While the accuracy shows some degradation compared to full-precision models, the achieved 65.8% mIoU remains practical for many applications.

Our work highlights both the potential and the challenges of implementing deep neural networks on FPGAs. The architectural optimizations and implementation techniques presented in this paper can inform future efforts to deploy complex vision models on resource-constrained embedded platforms.

Future work could explore more sophisticated quantization techniques, network architecture search for hardware-efficient models, and advanced memory management strategies to further improve the performance-accuracy-power trade-off for FPGA-based deep learning acceleration.

References

[1] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention (pp. 234-241).

[2] Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., Yu, J., Tang, T., Xu, N., Song, S., Wang, Y., & Yang, H. (2016). Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 26-35).

[3] Venieris, S. I., & Bouganis, C. S. (2018). fpgaConvNet: Mapping Regular and Irregular Convolutional Neural Networks on FPGAs. IEEE Transactions on Neural Networks and Learning Systems, 30(2), 326-342.

[4] Wang, J., Lou, Q., Zhang, X., Zhu, C., Lin, Y., & Chen, D. (2018). Design Flow of Accelerating Hybrid Extremely Low Bit-width Neural Network in Embedded FPGA. In 2018 28th International Conference on Field Programmable Logic and Applications (FPL) (pp. 163-169).

[5] Gschwend, D. (2016). ZynqNet: An FPGA-Accelerated Embedded Convolutional Neural Network. Master's Thesis, Swiss Federal Institute of Technology Zurich (ETH).

[6] Guo, K., Sui, L., Qiu, J., Yu, J., Wang, J., Yao, S., Han, S., Wang, Y., & Yang, H. (2018). Angel-Eye: A Complete Design Flow for Mapping CNN onto Embedded FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(1), 35-47.

[7] Terasic Technologies. (2019). DE10-Standard User Manual.

[8] Intel Corporation. (2020). Intel Cyclone V Device Handbook.

[9] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

[10] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.