

Cross-Platform Performance Analysis of GNN-Based SAR-ATR on FPGAs

Frederico Meletti Rappa¹, Alfredo Goldman¹

¹Institute of Mathematics and Statistics
University of São Paulo

{frederico.rappa,gold}@ime.usp.br

Abstract. *Graph Neural Networks have been proposed as a lightweight alternative to CNNs for SAR Automatic Target Recognition (SAR-ATR), representing images as sparse graphs to reduce computational cost. This work extends an existing GNN-based SAR-ATR FPGA accelerator with feature aggregation parameterization and cross-platform evaluation on Xilinx Alveo U280 and U55C boards, exploring the impact of SLR mapping, memory configuration, and pipeline parallelism. Results show that compared to the original design single-SLR placement reduces latency and that fewer scatter-gather pipelines result in better performance.*

1. Introduction

Synthetic Aperture Radar Automatic Target Recognition (SAR-ATR) is a remote sensing application that involves classifying images obtained from radars. These images are usually captured from moving, resource-constrained platforms, such as satellites and airplanes, and therefore require low latency and power consumption.

Convolutional Neural Networks (CNNs) are the traditional approach for SAR-ATR [Wickramasinghe et al. 2025], but are often computationally and memory intensive, and perform poorly on such devices. Graph Neural Networks (GNNs), on the other hand, offer an alternative, since the representation of images as graphs can employ sparse matrix operations common in GNNs and better capture the spatial relationships in the image [Zhang et al. 2022].

In this work, we extend the GNN accelerator proposed by Zhang et al. [Zhang et al. 2023] for SAR-ATR on FPGAs, exploring how architectural and memory decisions affect performance across different hardware platforms.

2. Background

2.1. Graph Neural Networks

Graph Neural Networks (GNNs) are a set of neural networks which operate directly on graph-structured data [Scarselli et al. 2008a, Scarselli et al. 2008b]. GNNs typically rely on *message passing*: at each layer, a node aggregates features from its neighbors into a “message” and combines this message with its own feature vector [Gilmer et al. 2017]. In this mechanism, the aggregation step is memory-bound, since neighboring nodes are often scattered non-contiguously in memory, while the combination step is compute-bound, requiring feature updates for every node.

3. GNN Accelerator Design

Our design expands on the work of [Zhang et al. 2023], based on SAR-ATR in the MS-TAR datasets. In this work, SAR images are converted into graphs, where each pixel becomes a node and edges are created between adjacent pixels. Although effective at accelerating GNN inference, Zhang et al. evaluate their design only on a single FPGA platform, Xilinx Alveo U280, with no design exploration. The model consists of GraphSAGE, attention, and pooling layers to reduce the graph’s dimensions, and Multi-Layer Perceptron layers to classify the features. As in [Zhang et al. 2023], the model and accelerator are designed to classify SAR images in real time, and are therefore limited to the image dimension and output format.

The accelerator adopts a design that divides the GNN inference into three modules: Feature Aggregation (FA), Feature Update (FU), and Multi-Layer Perceptron (MLP). Data is passed between modules through buffers in HBM, allowing for parallel execution of modules for different layers. As described by [Zhang et al. 2023], each set of modules makes a Processing Element, responsible for fully executing inference on a single graph. The FA module performs the aggregation and pooling steps using a scatter-gather approach: edges are read from memory and pipelined to parallel compute units. Each unit computes a partial update, either by accumulating the source vertex feature, or computing the pooling operation. After processing all edges, the partial updates are written back to memory. The FU module performs the combination step via a 2D systolic array, by multiplying the denser feature and weight matrices. The MLP module classifies features via an adder tree.

4. Evaluation

We evaluate the design on the MSTAR dataset, measuring latency, throughput, and energy consumption. The dataset contains 2,747 128×128 images across 10 military target classes, with images converted to graphs. We define latency as the average time to process a graph, throughput as the number of graphs per second, and energy consumption as the average energy per image, measured using the Xilinx Power Estimator (XPE) tool.

The accelerator designed is originally evaluated on a Xilinx Alveo U280, with both DDR5 and HBM2 memory, due to the need to handle parallel reads and writes and costly weight storage. DDR generally offers larger capacity at lower bandwidth, while HBM has higher bandwidth through memory dies located closer to the FPGA. In addition to the hardware in the authors’ work, our experiments employ a Xilinx Alveo U55C board. As seen in Table 1, although both devices contain similar resources, their memory hierarchies and topologies present important differences that affect accelerator design. A Super Logic Region (SLR) is a die slice of an FPGA, with adjacent SLRs connected by slower long-distance routing. The authors split the kernels across the three SLRs of the U280 to avoid performance bottlenecks introduced in the routing in a single SLR.

4.1. Memory Configuration and SLR Mapping

Three configurations were evaluated on the U280: the original baseline architecture, HBM-only memory usage, and a merged single-SLR design with no kernel splitting. The U55C replicated the HBM-only configuration for cross-platform comparison, as the board lacks DDR memory. The model was evaluated on an Intel i9-13900K CPU and an NVIDIA GTX 1070 GPU as baselines using PyTorch Geometric.

Tabela 1. Comparison between Xilinx Alveo U280 and U55C

	U280 (2019)	U55C (2023)
HBM2 total	8 GB	16 GB
HBM2 bandwidth	38 GB/s	-
SRAM total	43 MB	43 MB
SRAM bandwidth	35 TB/s	35 TB/s
Maximum power	225 W	150 W
LUTs	1,303,680	1,303,680
FFs	2,607,360	2,607,360
DSP slices	9,024	9,024
SLRs	3	3

Tabela 2. Comparison of performance of FPGA configurations for GNN inference

Board	Configuration	Latency $\pm$ std (ms)	Throughput (img/s)	Power (W)	Energy (J/img)
U280	Original	3.291 ± 0.411	304.0	33.76	0.1111
U280	HBM only	3.282 ± 0.403	304.7	30.27	0.0994
U280	Single-SLR	3.086 ± 0.335	324.1	33.64	0.1041
U55C	HBM only	3.299 ± 0.076	303.1	22.88	0.0755
CPU	i9-13900K	15.534 ± 5.497	64.4	92.85	1.443
GPU	GTX 1070	4.240 ± 4.595	235.8	36.37	0.129

As described, SAR-ATR applications generally require low latency and energy consumption: as seen in Table 2, the deviation of latency is under 10% of the average value, with the single-SLR design achieving the lowest values on average. Compared to the baseline design on the same board, this result indicates a possible performance gain in the design by minimizing communication between SLRs, contrary to the assumptions described in [Zhang et al. 2023]. The distance of the kernels and the proximity to the HBM gates in the SLR0 are likely relevant factors in the results. On the other hand, the U55C implementation presents higher average latency, but considerably lower variation in the data, even when compared with the same design in the U280. Energy consumption has similar values between designs, with the U55C presenting especially low average consumption per graph.

Additional experiments were conducted using PLRAM and external memory and, in both instances, the memory communication resulted in timeouts, likely due to the high cost of external memory access and impossibility of processing the data in PLRAM. Compared to the CPU all FPGA configurations achieve significantly lower latency and energy consumption, while compared to the GPU, the FPGAs maintained lower latency and higher throughput, necessary for SAR-ATR applications.

4.2. Pipeline Parallelism

We parameterized the FA module pipelines and varied the number of scatter-gather pipelines between 1, 2, 4, and 8. The experiment results prove counter-intuitive: fewer scatter-gather pipelines present better performance on both boards. On the U280, latency

increases from 2.583 ms at 1 pipeline to 4.151 ms at 8 pipelines, while throughput decreases from 363.4 to 240.9 images/s. Similarly, in the U55C, throughput is reduced from 372.8 to 242.5 images/s. Average power on the U280 rises with more pipelines, from 31.08 W to 33.86 W, while energy efficiency worsens significantly: the U280 consumes 0.0718 J/image at 1 pipeline and 0.1405 J/image at 8, with similar results in the U55C.

This behavior points to limitations in the design: while the image structure is represented as a graph, node features are stored in an array indexed by pixel position. Each pipeline holds a band of the feature array, and aggregating features in a band's edges requires loading rows from the neighboring band. Since only the aggregation computation runs in parallel across pipelines, while data loading and storing are performed sequentially per pipeline, this operation with neighboring bands grows with the pipeline count, eventually dominating the pipelined execution and the gains from additional parallelism.

4.3. Conclusion and Future Work

We explored memory configuration, SLR mapping, and pipeline parallelism for a GNN-based SAR-ATR FPGA accelerator across Xilinx Alveo U280 and U55C. Expanding on previous work, single-SLR placement and HBM-only memory reduce latency and power, while fewer scatter-gather pipelines result in better performance, indicating the need for future optimizations. The U55C offers the best energy efficiency and latency consistency, with FPGA designs achieving up to $5\times$ lower latency and $19\times$ better energy efficiency over CPU. Future work will extend the design to other GNN models and graph topologies.

This study was supported by the São Paulo Research Foundation (FAPESP), Brazil. Process Number #19/26702-8.

Referências

- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Int. Conf. on Machine Learning*, pages 1263–1272. Pmlr.
- Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008a). Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102.
- Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008b). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.
- Wickramasinghe, S., Lin, Y.-C., Raghavendra, C., and Prasanna, V. (2025). Smart: High-performance sar atr through model-architecture co-design on fpga. In *2025 IEEE 33rd Int. Symp. on Field-Programmable Custom Computing Machines (FCCM)*, pages 75–84.
- Zhang, B., Kannan, R., Prasanna, V., and Busart, C. (2022). Accurate, Low-latency, Efficient SAR Automatic Target Recognition on FPGA. In *2022 32nd Int. Conf. on Field-Programmable Logic and Applications (FPL)*, pages 1–8.
- Zhang, B., Kannan, R., Prasanna, V., and Busart, C. (2023). Accelerating GNN-Based SAR Automatic Target Recognition on HBM-enabled FPGA. In *2023 IEEE High Performance Extreme Computing Conf. (HPEC)*, pages 1–7.