Predicting Hardware Resource Allocation of MVAU Modules in FINN-Generated FPGA Accelerators

**Beatriz Aline Arend, Arthur Ferreira Ely,
Fernanda Lima Kastensmidt and Mariana Recamonde-Mendoza**

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

Abstract. *We propose a method to predict hardware resource allocation of the Matrix Vector Activation Unit (MVAU) in FPGA-based neural network accelerators generated with the FINN framework. A dataset built from synthesis results includes architectural parameters and utilization of LUTs, FFs, RAMBs, and DSPs. Three regression models were evaluated via 10-fold cross-validation. All achieved average* R^2 *above* 0.99*, with XGBoost yielding the lowest MSE and MAE. Feature importance analysis showed that the Number of Processing Elements and RTL implementation dominate predictions, together contributing nearly 69%. The approach enables accurate and practical resource estimation to support efficient FPGA accelerator design.*

1. Introduction

Artificial Intelligence (AI) has become essential across diverse domains, from autonomous vehicles and medical diagnosis to aerial image analysis. Among machine learning approaches, neural networks, particularly deep architectures, stand out for their accuracy, but they demand substantial computational power and energy. To address these requirements, specialized hardware accelerators have gained popularity, with GPUs, DPUs, and FPGAs playing a central role in both cloud and edge deployments.

Field-Programmable Gate Array (FPGA) devices offer a unique balance between performance and energy efficiency through fine-grained reconfigurability, making them ideal for embedded and resource-constrained environments. For BNN workloads, experimental comparisons in [Nurvitadhi et al. 2016] have demonstrated that FPGA accelerators can achieve orders of magnitude better performance per watt than even highly optimized CPU and GPU software. Frameworks such as FINN [Umuroglu et al. 2017] streamline the deployment of quantized neural networks (QNNs) onto FPGAs, allowing designers to configure parallelism and balance throughput against hardware usage. However, in practice, accelerator designs often face strict area constraints, especially when sharing the FPGA fabric with CPUs, memory controllers, and other IP cores in System-on-Chip (SoC) platforms.

A common challenge is that designers must iterate through time-consuming synthesis cycles, sometimes discovering only at the end that the design exceeds available resources or that the chosen network topology is infeasible for the target FPGA. To address this issue, this work proposes a machine learning–based method to predict the FPGA area utilization of the main computational module in FINN accelerators from its configuration parameters. This enables early detection of infeasible designs, saving valuable build time and avoiding unnecessary training of models that cannot be deployed on the intended hardware.

2. Background

FINN [Umuroglu et al. 2017] is a framework that generates FPGA accelerators for QNNs by converting ONNX models into hardware implementations via High-Level Synthesis (HLS). Models are trained in PyTorch with Brevitas [Pappalardo et al. 2024] for weight and activation quantization, then exported to ONNX for FINN, which maps layer operations into specialized harwdware modules and generates configuration parameters. The Matrix-Vector Activation Unit (MVAU) is a key component, performing most computations for convolutional and fully connected layers. Its configuration largely determines performance and resource usage. Parallelism in the MVAU is controlled by folding parameters: Processing Elements (PEs) for parallel output channels and SIMD lanes for parallel multiplications. Adjusting these parameters balances throughput, latency, and FPGA resource consumption, enabling optimized designs for resource-constrained systems.

3. Methodology

Data was collected from ONNX models generated by FINN, which include layer hyperparameters and parallelism settings (i.e., PEs and SIMD per PE). These models provided the input features for our predictive pipelines. Accelerators were synthesized using Vivado to obtain detailed resource utilization per layer (LUTs, FFs, RAMBs, DSPs). Extraction and organization were automated in Python, and outputs were stored in spreadsheets. The dataset comprises 213 builds, yielding 832 MVAU instances with diverse configurations. The builds were generated exhaustively by iteratively increasing the SIMD factor up to the limit of the slowest layer in each round, followed by increasing the number of PEs, ensuring maximum exploration of the design space. Two representative CNN topologies for SAT-6 [Basu et al. 2015] classification (t1: two conv layers, t2: four conv layers) were used to guarantee diversity and realism in the dataset. Tables 1 and 2 summarize the dataset. Table 1 shows the hardware resource metrics targeted for prediction. Table 2 lists the MVAU module parameters used as input features.

The collected raw data was then cleaned and organized for predictive modeling. Datatypes were converted to numeric bitwidths, tensor dimensions were flattened, and constant or negligible features were removed. Categorical features, such as *resType*, were one-hot encoded. Highly correlated features were further filtered by retaining the more important variable. Resource metrics were unified to simplify modeling: Total LUTs sums Logic LUTs, LUTRAMs, and SRLs; RAMB aggregates RAMB18 and RAMB36 (each RAMB18 = 0.5 RAMB36).

3.1. Experimental Setup

We employed a 10-fold cross-validation scheme for multi-target prediction. Following common practice in multi-target learning, results were aggregated by averaging the scores across all outputs, yielding a single performance metric per model. The task was to predict the number of Total LUTs, FFs, RAMBs, and DSP Blocks required by the MVAU module in the target FPGA.

Three models were evaluated: **(i)** a neural network with three layers, ReLU activation, and learning rate 0.001; **(ii)** a Random Forest with maximum depth 30, 200 estimators, and minimum sample split 2; and **(iii)** XGBoost with maximum depth 5, 300 estimators, learning rate 0.1, subsample 0.8, and column sample by tree 1.0. The models' hyperparameters were defined empirically, based on preliminary experimentation.

Table 1. Predicted FPGA Resources

Resource	Description
Total LUTs	Sum of Logic LUTs and LUTRAMs.
Logic LUTs	Used for general-purpose logic.
LUTRAMs	LUTs configured as small RAMs.
SRLs	LUTs as shift registers for pipelining.
FFs	Flip-Flops for sequential storage.
RAMB36	Dedicated 36-kb block RAMs.
RAMB18	Dedicated 18-kb block RAMs.
DSPs	Specialized blocks for arithmetic.

Table 2. MVAU Parameters

Param	Description
Processing Elements (PE)	Computes output channels in parallel.
RTL Implementation	Indicates if the module uses a RTL or a HLS implementation.
Accumulator Data Type	Data type of the internal accumulator, in bits.
Weight Matrix Width (MW)	Width of the weight matrix.
Resource Type: Auto	Automatic selection of hardware resources (LUTs, DSPs).
SIMD Lanes per PE	Number of parallel input lanes per PE.
Input Data Type	Data type of input activations, in bits.
Clock Cycles Estimate	Estimated clock cycles for the operation.
Weight Matrix Height (MH)	Height of the weight matrix.
Output Data Type	Data type of the outputs, in bits.
Weight Data Type	Data type of the weights, in bits.
Input Vectors	Number of input vectors processed.

4. Results

For baseline comparison, we first tested a default linear regression model, which showed reasonable performance, indicating that much of the data follows a linear trend. This suggested that more advanced regression algorithms could achieve even better results. Table 3 confirms this: all three models achieved average R^2 scores above 0.99.

XGBoost achieved the best performance overall, with the lowest MSE and MAE, making very few errors and with minimal deviation. The neural network and Random Forest also performed well, with only slightly lower R^2 values. However, the neural network exhibited a higher MAE, while the Random Forest had a higher MSE, indicating more pronounced individual errors.

Table 3. Performance of all models in multi-target hardware resource prediction

Model	Avg. MSE ($\pm$ std)	Avg. MAE ($\pm$ std)	Avg. R^2 ($\pm$ std)
Linear Regression	522682.02 ± 550975.20	365.23 ± 371.12	0.7836 ± 0.0524
Random Forest	9400.70 ± 9974.42	17.10 ± 16.95	0.9969 ± 0.0011
Neural Network	6052.40 ± 6364.52	36.36 ± 37.45	0.9974 ± 0.0012
XGBoost	$\mathbf{343.23 \pm 343.41}$	$\mathbf{4.04 \pm 4.04}$	$\mathbf{0.9999 \pm 0.0001}$

4.1. Discussion

To understand the high model performance, we analyzed feature importance using the Random Forest model (Figure 1). The analysis reveals that the Number of Processing

Elements (PE) and RTL implementation are predominant, contributing to nearly 69% of the model's predictions. This is expected, as these parameters directly control the parallelism and resource mapping efficiency in FINN accelerators, thereby heavily influencing the final utilization of LUTs, DSPs, and FFs.

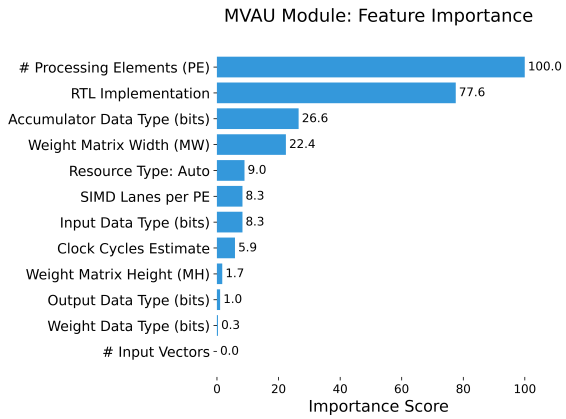

Figure 1. Normalized cumulative feature importance (0–100) of MVAU parameters for predicting hardware resource utilization. PE (38.1%) and isRTL (29.7%) dominate the impact.

5. Conclusion

By enabling prediction of MVAU hardware resource allocation, this work adds a valuable tool to the FPGA design process. A dataset was created from synthesis results of neural network accelerators generated with the FINN framework, enabling accurate multi-target prediction of LUTs, FFs, RAMBs, and DSPs. The proposed models, particularly XGBoost, achieved near-perfect accuracy. Future work includes developing a classification engine integrating specialized models for each accelerator module type, enabling end-to-end prediction of total resource consumption for complete accelerator designs.

References

[Basu et al. 2015] Basu, S., Ganguly, S., Mukhopadhyay, S., DiBiano, R., Karki, M., and Nemani, R. (2015). DeepSat: A learning framework for satellite imagery. In *Proc. 23rd Int. Conf. Advances in Geographic Information Systems*, Seattle, Washington. doi: 10.1145/2820783.2820816.

[Nurvitadhi et al. 2016] Nurvitadhi, E., Sheffield, D., Sim, J., Mishra, A., Venkatesh, G., and Marr, D. (2016). Accelerating binarized neural networks: Comparison of fpga, cpu, gpu, and asic. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 77–84.

[Pappalardo et al. 2024] Pappalardo, A., Franco, G., Colbert, I., et al. (2024). Xilinx/Brevitas.

[Umuroglu et al. 2017] Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., and Vissers, K. (2017). FINN: A framework for fast, scalable binarized neural network inference. In *Proc. 2017 ACM/SIGDA International Symp. on Field-Programmable Gate Arrays*, FPGA '17, page 65–74, New York, NY, USA. ACM. doi: 10.1145/3020078.3021744.