



Machine Learning Assisted Design Space Exploration and Performance Prediction of MAERI DNN Accelerators

Karra Greeshma¹ and K. Jhansi Rani²

¹PG Scholar, ²Associate Professor

Department of Electronics and Communication Engineering
University College of Engineering Kakinada (UCEK)
Jawaharlal Nehru Technological University Kakinada (JNTUK)
Kakinada, Andhra Pradesh, India

Abstract: Deep neural network (DNN) accelerators must balance throughput, power and area, and physically implementing every candidate configuration to explore that trade-off is prohibitively expensive. This work builds an ML-assisted flow that establishes actual, physically measured runtime and energy for two implemented MAERI configurations, MAERI-64 and MAERI-128, and predicts these same metrics for a third, considerably larger MAERI-256 configuration by combining a hardware-trend extrapolation from MAERI-64/128 with independent architectural simulation, avoiding the need for repeated synthesis, physical design or simulation on every new query. MAERI-64 and MAERI-128 were synthesized at 45 nm (Cadence Genus) and physically implemented at 90 nm (Cadence Innovus), giving actual ResNet-50 runtimes of 239.96 ms and 120.54 ms and energies of 40.96 mJ and 41.10 mJ. Cycle-level simulation across five CNN workloads (ResNet-50, ResNet-34, AlexNet, GoogLeNet, MobileNetV1) produced a 355-sample, 26-feature dataset used to train polynomial regression, ElasticNet and gradient boosting models, later extended with XGBoost and ExtraTrees. The extended models reach mean errors as low as 2.52% (runtime) and 5.49% (energy) for ResNet-34, though irregular workloads like GoogLeNet and MobileNetV1 remain harder to fit (14–24% error). For MAERI-256, frequency (0.25 GHz) and power (681.61 mW) were linearly extrapolated from the 64/128 trend and combined with real simulated cycle counts, predicting a ResNet-50 runtime of 63.73 ms and energy of 43.44 mJ. MAERI-128 roughly halves ResNet-50 runtime relative to MAERI-64 at the cost of higher power, and this trend extrapolates consistently to MAERI-256, confirming that the proposed ML-assisted flow can reliably guide design-space exploration for larger, unimplemented MAERI configurations without the time and cost of a full physical-design run.

Index Terms - MAERI, deep neural network (DNN) accelerator, application-specific integrated circuit (ASIC) synthesis, Cadence Genus, Cadence Innovus, design space exploration (DSE), machine learning, polynomial regression, XGBoost, ExtraTrees, MAERI-256 extrapolation, runtime and energy prediction.

I. INTRODUCTION

Deep neural network (DNN) workloads used in computer vision, natural language processing and autonomous systems have grown rapidly in depth and parameter count, placing increasing demands on the compute throughput and memory bandwidth of the underlying hardware. General-purpose processors and even graphics processing units (GPUs) are often power-inefficient for such regular, data-parallel workloads, which has motivated the development of dedicated DNN accelerators such as systolic arrays, Eyeriss-style spatial architectures, and reconfigurable-interconnect accelerators such as MAERI.

MAERI (Multiply-Accumulate Engine with Reconfigurable Interconnects) is an open-source DNN accelerator architecture that replaces the rigid, fixed interconnect of a conventional systolic array with a reconfigurable distribution and reduction network built from small switches between processing elements (PEs). This allows MAERI to adapt its dataflow on a per-layer basis and to sustain high PE utilization across both dense and sparse workloads, at the cost of additional interconnect and control complexity relative to a systolic array.

Evaluating a candidate accelerator configuration in terms of area, power, frequency, runtime and energy conventionally requires (i) cycle-accurate architectural simulation to obtain workload execution cycles and (ii) logic synthesis and physical design using an ASIC flow to obtain silicon-level area, power and timing. Both steps are time-consuming, and physical design in particular can take from several hours to multiple days per configuration, which makes an exhaustive design space exploration (DSE) across PE counts, technology nodes and workloads impractical. This motivates the use of machine learning (ML) models trained on a limited set of physically implemented and simulated configurations to predict the runtime and energy of unseen, typically larger, configurations.

This paper reports a postgraduate project that builds such an ML-based prediction flow specifically for the MAERI accelerator at two physically implemented PE configurations, MAERI-64 and MAERI-128, synthesized in 45 nm technology and physically implemented in 90 nm technology. The RTL of MAERI, originally released in Bluespec System Verilog (BSV), was first compiled to synthesizable Verilog, then carried through synthesis (Cadence Genus, 45 nm) and physical design (Cadence Innovus, 90 nm) to obtain area, power, gate count and timing data for both configurations. In parallel, cycle-level architectural simulation of MAERI-

64 and MAERI-128 was performed for five CNN workloads to obtain per-layer cycle counts. The two datasets were merged and used to train regression and ensemble ML models capable of predicting workload runtime and energy directly from architectural and layer-level features, without repeating synthesis, physical design or simulation for every new query. As a further extension, the trained frequency, power and area characterization is linearly extrapolated to a third, larger, unimplemented configuration, MAERI-256, and combined with actual architectural-simulation cycle counts obtained for that configuration to predict its per-workload runtime and energy without carrying out a physical-design run at this larger PE count.

The remainder of this paper is organized as follows. Section II reviews background concepts and related DSE literature for DNN accelerators. Section III describes the tools used in the RTL generation, physical design and ML flow. Section IV summarizes the existing ML-based DSE approach that motivated this work. Section V describes the proposed MAERI-specific methodology, including its extension to MAERI-256. Section VI presents synthesis, physical design and ML prediction results, a DSE comparison of MAERI-64 and MAERI-128, and the extrapolated MAERI-256 projections. Section VII concludes the paper and outlines future work.

II. BACKGROUND AND RELATED WORK

A. DNN Accelerator Architectures

A DNN accelerator maps the matrix-multiplication and convolution operations that dominate DNN inference and training onto an array of PEs connected through some interconnect topology. Systolic arrays such as those used in the Google TPU offer a compact, rigid interconnect that is area- and power-efficient but suffers from underutilization on irregular or sparse layers [4]. Eyeriss demonstrated a row-stationary dataflow that improves energy efficiency by maximizing data reuse within the PE array [3]. Flexible-interconnect accelerators such as MAERI and SIGMA generalize the interconnect to support many dataflow mappings per layer, improving utilization at the cost of additional routing and control area.

B. Design Space Exploration for Accelerators

Because manually sweeping accelerator configurations through a full physical-design flow is slow, several works have investigated faster, model-based DSE. Esmaeilzadeh et al. proposed learning-based performance prediction for hardware-accelerated ML algorithms using AutoML regression models trained on synthesized 2-D designs [6]. Software-defined DSE approaches have similarly used analytical and learned cost models to search accelerator architecture parameters efficiently [4]. On-chip memory technology DSE for mobile DNN accelerators using 3-D vertical RRAM has also been studied for iso-throughput configurations [5]. More recently, 3DNN-Xplorer trained separate ML models gradient boosting for effective frequency, and polynomial/ElasticNet regression for power, runtime and energy on MAERI, Axiline and systolic-array configurations synthesized and physically implemented across 16 nm and 28 nm technology nodes, and used these models to extrapolate performance to configurations with up to 16,384 PEs with a maximum prediction error of 13.9% [1]. That work also introduced heterogeneous 3-D compute-on-memory integration styles that combine different technology nodes for the compute and memory tiers of a two-tier 3-D IC, showing energy and runtime benefits over homogeneous 3-D designs.

C. MAERI and its Simulation Support

MAERI's reconfigurable distribution and augmented reduction tree network allows a single physical PE array to be time-multiplexed across multiple virtual neurons and dataflow mappings [1]. The STONNE simulator provides cycle-level modelling of MAERI-like flexible accelerators, including support for multiple dataflow mappings, but does not by itself model on-chip memory reuse across DNN layers or full physical-design-level power and area [1]. Related studies on reconfigurable systolic accelerators such as ReDAS have explored dynamically adaptable systolic interconnects to improve utilization relative to fixed systolic designs, while compiler-integrated evaluation frameworks such as Bifrost couple flexible accelerator mapping search with a deep-learning compiler stack, though at the cost of higher tool-chain complexity and without full silicon-level validation.

D. Research Gap

Across this literature, three gaps motivate the present work. First, most existing DSE studies for MAERI-like flexible accelerators rely primarily on architectural simulation and do not combine it with actual post-physical-design area, power and timing data obtained from a commercial ASIC flow. Second, ML-based extrapolation frameworks such as 3DNN-Xplorer have so far been evaluated mainly on 16 nm/28 nm technology nodes and a systolic/MAERI/Axiline design space, leaving open the question of how well similar ML techniques generalize to a 45 nm MAERI-only implementation trained on a comparatively small number of PE configurations (here, only 64 and 128 PEs). Third, flexible interconnects such as MAERI's reconfigurable network introduce complex, workload-dependent utilization behaviour that basic regression models can under-fit, particularly for irregular workloads, motivating the use of tree-based ensemble models such as XGBoost and ExtraTrees in addition to polynomial regression.

III. TOOLS AND METHODOLOGY USED

A. RTL Generation from Bluespec System Verilog

The MAERI accelerator RTL is distributed in Bluespec System Verilog (.bsv). To obtain a synthesizable Verilog netlist-ready description, a Linux/Ubuntu build environment was configured with the Glasgow Haskell Compiler (GHC) and the Cabal package manager, which are required to build the Bluespec toolchain and its Haskell dependencies. The Bluespec Compiler (BSC) was then used to compile the .bsv sources for the 64-PE and 128-PE MAERI configurations into structural Verilog (.v), which was used as the RTL input to the ASIC synthesis flow. Fig. 1 summarizes this generation flow together with the downstream dataset-construction step: a resource description (PE count, distribution- and gather-network bandwidth) and a per-layer description (R, S, C, K, X, Y, stride) are combined by the MAERI RTL generator and BSV compiler with a building-block switch-fabric library to produce both the Verilog RTL carried into Genus/Innovus and the cycle-accurate architectural-simulation outputs used to build the ML training dataset.

MAERI RTL Generation and Dataset-Construction Flow Used in This Work

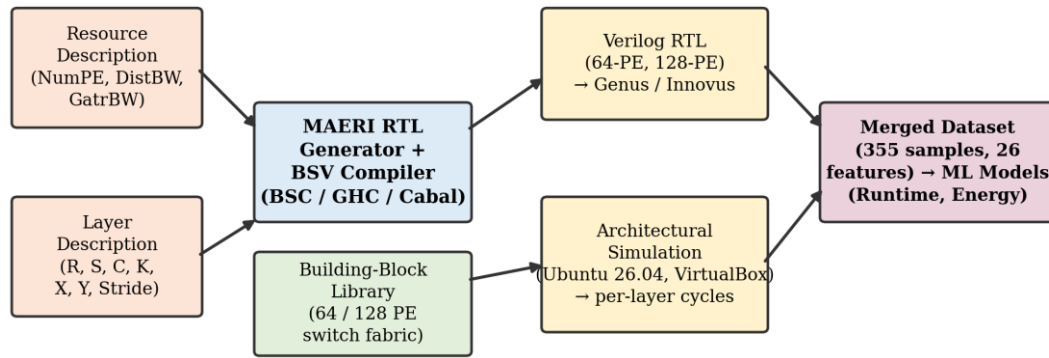


Fig. 1. MAERI RTL generation and dataset-construction flow.

B. Remote VLSI Workflow

MobaXterm, an enhanced Windows terminal with an integrated X11 server and tabbed SSH client, was used to connect to the Linux-based EDA servers running Cadence tools. This allowed the Cadence Genus and Innovus graphical interfaces to be visualized remotely and simplified organization of RTL, constraint (SDC) and TCL script files throughout the flow.

C. Logic Synthesis — Cadence Genus

Cadence Genus was used to synthesize the MAERI-64 and MAERI-128 Verilog RTL into a gate-level netlist using a 45 nm standard-cell technology library. A target clock period of 4 ns was applied through SDC timing constraints, and logic synthesis and technology mapping were performed for both configurations. Post-synthesis reports provided combinational and sequential instance counts, total cell area, internal and switching power, and worst negative slack (WNS). Fig. 2 summarizes the Genus synthesis flow used, from RTL/Liberty/LEF/SDC inputs to the generated gate netlist, timing, area and SDF reports.

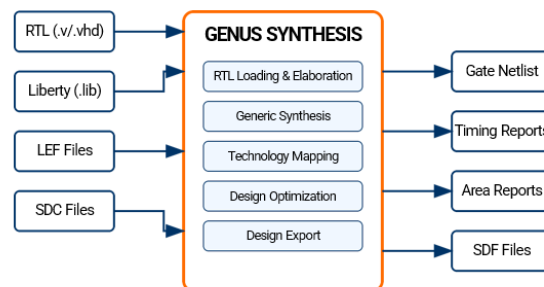


Fig. 2. Cadence Genus logic-synthesis flow used to generate the gate-level netlist, timing, area and SDF reports for MAERI-64 and MAERI-128.

D. Physical Design — Cadence Innovus

Cadence Innovus was used to carry the synthesized netlists through physical design. The flow followed the standard stages of Floor planning (core-area definition and block placement), power planning (VDD/VSS grid construction), standard-cell placement (with timing, congestion and area optimization), clock tree synthesis (CTS, to build and optimize the clock distribution network), and detailed routing followed by design-rule-check (DRC) and post-route timing verification, as summarized in Fig. 3. Representative Innovus commands used in the flow included source Default.globals, init_design, place_opt_design, ccopt_design and routeDesign. Area and power were extracted at each stage (Floor planning, placement, CTS and routing) for both MAERI-64 and MAERI-128.



Fig. 3. Cadence Innovus physical-design flow: Floor planning, placement, clock-tree synthesis, routing and GDSII export.

E. Architectural Simulation Environment

Cycle-level architectural simulation of MAERI-64 and MAERI-128 was performed inside an Oracle VirtualBox virtual machine running Ubuntu 26.04 LTS, configured independently of the physical Linux EDA server used for synthesis and physical design. This environment was used to compile and run the MAERI architectural simulator for five DNN workloads ResNet-50, ResNet-34, AlexNet, GoogLeNet and MobileNetV1 layer by layer, recording per-layer architectural parameters (filter height R, filter width S, input channels C, output channels K, input feature-map dimensions X and Y, and stride) together with the total execution cycles for that layer and configuration.

F. Dataset Construction and ML Environment

The per-layer cycle counts obtained from architectural simulation were combined with the post-physical-design frequency and power results from Cadence Innovus to compute per-layer runtime and energy, following the same general approach of combining

physical design metrics with workload simulation used in prior ML-based DSE work [1]. The combined dataset 355 samples across 26 features spanning architectural, workload and physical-design parameters for MAERI-64 and MAERI-128 was assembled and analysed in Google Colab using Python-based data-science and ML libraries (scikit-learn for polynomial regression, ElasticNet and gradient boosting; XGBoost and scikit-learn's ExtraTrees for the extended ensemble models).

IV. EXISTING WORK

The base study that motivated this project, 3DNN-Xplorer, proposes an ML-based DSE framework for heterogeneous 3-D DNN accelerators combining 28 nm and 16 nm technology nodes for compute and memory tiers [1]. It trains four separate models gradient boosting for effective frequency, and polynomial/ElasticNet regression for chip power, workload runtime and workload energy on MAERI, Axiline and systolic-array configurations with 10 to 256 PEs, and shows that these models extrapolate to configurations of up to 16,384 PEs with a maximum absolute error of 13.9%. Its DSE results identify that a heterogeneous 3-D style with 28 nm compute and 16 nm memory is the most energy-efficient choice for a fixed PE count, offering up to 50% energy savings and 8.8% lower runtime relative to other 3-D integration styles, while a 16 nm-compute/28 nm-memory style is more area-efficient for a fixed on-chip memory capacity.

This existing work, however, targets a heterogeneous multi-technology 3-D design space rather than a single, homogeneous 45 nm 2-D implementation, and its MAERI training set spans 16–64 PEs before extrapolation. It also relies primarily on polynomial regression and gradient boosting; more expressive tree-ensemble methods such as XGBoost and ExtraTrees, which can better capture the non-linear, workload-dependent utilization behaviour of a flexible interconnect such as MAERI's, are not explored. These observations directly motivated the present project's focus on a MAERI-only, 45 nm, two-PE-configuration study extended with ensemble ML models.

V. PROPOSED WORK

A. MAERI Accelerator Configurations Evaluated

Two MAERI configurations were physically implemented and evaluated in this work: MAERI-64 (64 PEs) and MAERI-128 (128 PEs). Both configurations use MAERI's reconfigurable distribution network to broadcast operands to the PE array and its augmented reduction tree network to collect partial sums, allowing the same physical fabric to be reconfigured for different DNN layer shapes and dataflow mappings. Fig. 4 illustrates this arrangement at a conceptual level: a pre-fetch buffer feeds operands into a fan-out distribution network built from small reconfigurable switches, which drives the PE array; an augmented reduction tree network (a fan-in mirror of the distribution network, augmented with additional links for partial-sum forwarding between adjacent branches) then collects and combines the PE outputs. Reconfiguring the switches in these two networks lets the same 64- or 128-PE fabric be time-multiplexed across many virtual neurons and dataflow mappings, which is what allows MAERI to sustain high PE utilization on both the regular convolutional layers and the more irregular, e.g. depth wise-separable, layers found in the five workloads evaluated in this study. A third, larger configuration, MAERI-256 (256 PEs), is additionally evaluated in Section V-E and Section VI-F through extrapolation rather than physical implementation.

MAERI Reconfigurable-Interconnect Accelerator (Conceptual Overview)

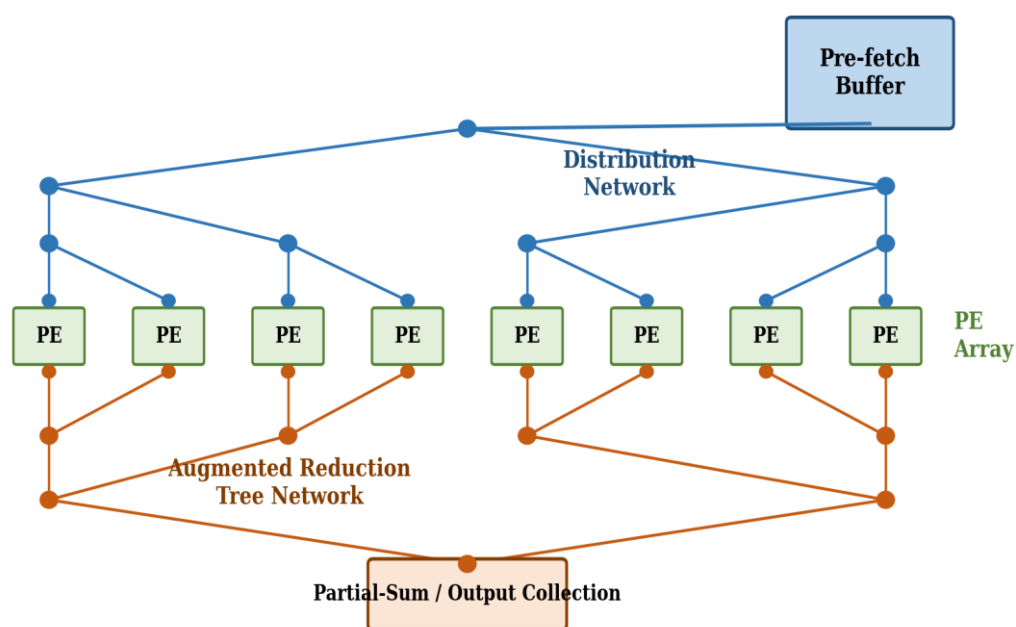


Fig. 4. Conceptual overview of the MAERI reconfigurable-interconnect accelerator: a pre-fetch buffer feeds a fan-out distribution network to the PE array, whose partial sums are collected by an augmented, fan-in reduction tree network.

B. End-to-End Design Flow

The overall flow proceeds in six stages, summarized in Fig. 5. (1) RTL generation: the MAERI BSV sources for both PE configurations are compiled to Verilog using BSC. (2a) Logic synthesis: the Verilog RTL is synthesized in Cadence Genus at a 4 ns clock period using a 45 nm library, producing a gate-level netlist. (2b) Architectural simulation, carried out in parallel: MAERI-64 and MAERI-128 are simulated layer-by-layer in STONNE for ResNet-50, ResNet-34, AlexNet, GoogLeNet and MobileNetV1 inside the Ubuntu 26.04 VirtualBox environment, yielding total execution cycles per layer. (3) Physical design: the synthesized netlist is carried through Floor planning, power planning, placement, CTS and routing in Cadence Innovus using a 90 nm library, yielding the effective frequency, power and area for both configurations. (4) Dataset merge: the physical-design results and the simulation-derived cycle counts are combined for each layer and workload using

$$\text{Runtime} = \text{Total Cycles} / \text{Frequency} \quad (1)$$

$$\text{Energy} = \text{Power} \times \text{Runtime} \quad (2)$$

to obtain the 355-sample, 26-feature training dataset described in Section V-C, in which runtime and energy are true, measured values rather than predictions. (5) ML training: this true-value dataset is used to train the baseline and proposed ML models (Section V-D), which predict per-workload runtime and energy directly from architectural and layer-level features and are validated against these true values (Tables IV, VI), demonstrating that synthesis, physical design and simulation need not be re-run for every new query within the trained design space; these models do not themselves generate the MAERI-64/128 results reported in Section VI. (6) Extrapolation: separately, the actual post-physical-design frequency, power and area of MAERI-64/128 (Stage 3) are linearly extrapolated to the unimplemented MAERI-256 configuration (Section V-E) and combined with actual MAERI-256 simulation cycles via the same Eqs. (1)–(2) to obtain its predicted per-workload runtime and energy the only genuinely model-based runtime/energy predictions reported in this paper.

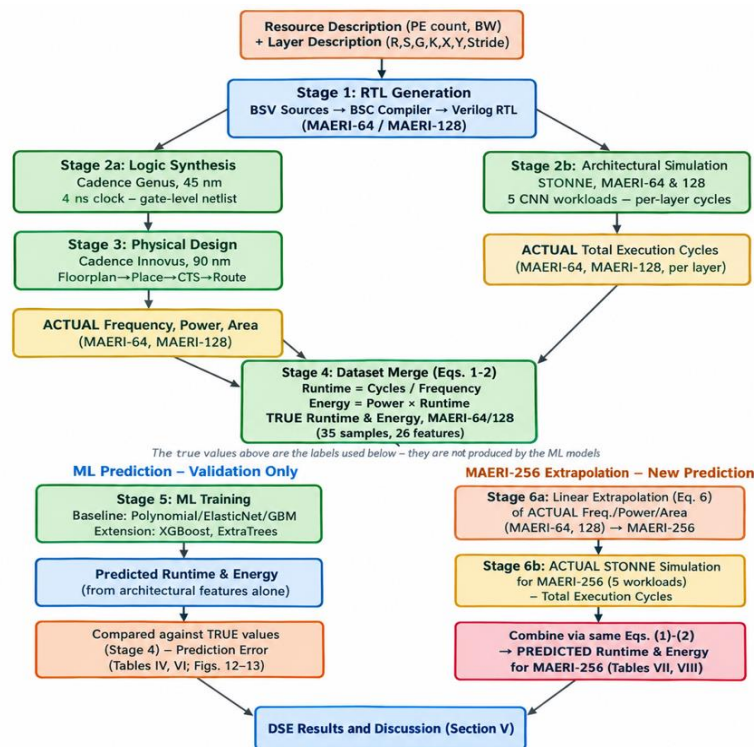


Fig. 5. End-to-end design flow.

C. Workloads and Layer-Level Features

Five representative CNN workloads were used: ResNet-50, ResNet-34, AlexNet, GoogLeNet and MobileNetV1, covering a range of layer depths, filter shapes and channel counts. For each convolutional layer of each workload, the following parameters were recorded: filter height (R), filter width (S), input channels (C), output channels/kernels (K), input feature-map height and width (X, Y), and stride, together with the number of input channels mapped per cycle (TC) and filter kernels mapped per cycle (TK) supported by the MAERI dataflow, and the resulting total execution cycles for that layer and PE configuration.

D. Machine Learning Models

Two families of models were used. The baseline models polynomial regression, ElasticNet-regularized polynomial regression, and gradient boosting were trained first, following the general modelling strategy of prior ML-based DSE work in which frequency-like saturating metrics are modelled with gradient boosting while power, runtime and energy, which vary smoothly but non-linearly with the architectural and workload parameters, are modelled with polynomial/ElasticNet regression [1]. For an input feature vector x (comprising #PE, memory bandwidth, R, S, C, K, X, Y, stride, TC and TK), the degree- d polynomial regression model used for the baseline power, runtime and energy predictors takes the form

$$\hat{y} = \beta_0 + \sum_j \beta_j \phi_j(x), \quad j = 1 \dots p \quad (3)$$

where $\hat{y}$ is the predicted metric, $\phi_j(x)$ are the degree- $\leq d$ monomial features generated from x , and β_j are the regression coefficients. To curb overfitting from this large monomial expansion, the ElasticNet-regularized variant minimizes

$$L = (1/2n) \sum_i (y_i - \hat{y}_i)^2 + \lambda_1 \|\beta\|_1 + 0.5\lambda_2 \|\beta\|_2^2 \quad (4)$$

where y_i and $\hat{y}_i$ are the actual and predicted values for the i -th of n training samples, $\|\beta\|_1$ and $\|\beta\|_2^2$ are the L1 and L2 norms of the coefficient vector, and λ_1, λ_2 are regularization weights balancing feature selection against coefficient shrinkage, following the same general ElasticNet formulation used for power prediction in prior ML-based DSE work [1]. As an extension, XGBoost and ExtraTrees regressors were trained on the same 355-sample, 26-feature dataset to predict per-workload runtime and energy, using hyperparameter tuning and grouped cross-validation (grouping by workload/layer to avoid leakage between layers of the same workload) to obtain a more reliable estimate of generalization performance. Model quality was assessed using the coefficient of determination (R^2), mean absolute error (MAE), root-mean-squared error (RMSE), and the mean/maximum percentage error relative to the simulated/measured runtime and energy,

$$\%Error_i = |y_i - \hat{y}_i| / y_i \times 100 \quad (5)$$

with the mean percentage error obtained by averaging $\%Error_i$ across all held-out samples of a given workload, and the maximum percentage error taken as the largest single-sample $\%Error_i$ for that workload, both reported throughout Section VI.

E. Extension to MAERI-256 via Linear Extrapolation

As a further extension beyond the two physically implemented configurations, the post-physical-design characterization of MAERI-64 and MAERI-128 was used to project the frequency, power and area of a third, larger configuration, MAERI-256 (256 PEs), for which no physical design was carried out. Because both MAERI-64 and MAERI-128 close timing at the same 4 ns target clock period (0.25 GHz) in Cadence Genus/Innovus, the effective operating frequency is carried forward unchanged to MAERI-256. Power and area, which scale in the observed data in an essentially linear fashion between the two implemented configurations, are extrapolated to 256 PEs using a two-point linear fit through the MAERI-64 and MAERI-128 post-physical-design values,

$$M(PE) = a \cdot PE + b, \quad a = (M_{128} - M_{64}) / (128 - 64) \quad (6)$$

where M denotes either power or area, M_{64} and M_{128} are its actual post-physical-design values for MAERI-64 and MAERI-128, and $b = M_{64} - a \cdot 64$ is the fitted intercept; evaluating Eq. (6) at $PE = 256$ yields a predicted power of 681.6 mW and a predicted area of 2,088,541 μm^2 for MAERI-256, as summarized in Section VI-F. Unlike frequency, power and area, the runtime and energy of MAERI-256 are not obtained by extrapolation; instead, MAERI-256 was additionally simulated architecturally using STONNE for the same five CNN workloads used elsewhere in this study, to obtain actual per-layer execution-cycle counts for the 256-PE configuration. These actual cycle counts are combined with the extrapolated frequency and power using the same Eqs. (1)–(2) to obtain a per-workload runtime and energy prediction for MAERI-256 without requiring a full synthesis-and-physical-design run at this larger PE count. Because no physical implementation of MAERI-256 exists, this projection cannot itself be validated against real Genus/Innovus results and is reported as an extrapolated projection rather than a directly measured result.

VI. RESULTS AND DISCUSSION

A. Synthesis Results

Table I summarizes the post-synthesis results for MAERI-64 and MAERI-128 obtained from Cadence Genus at a 4 ns target clock period in 45 nm technology. MAERI-128 requires roughly double the combinational and sequential instance count of MAERI-64, and correspondingly close to double the total cell area, while the internal and switching power per PE remain comparable between the two configurations at the synthesis stage; both configurations meet timing with a small positive slack. Figs. 6 and 7 show the corresponding Genus netlist views: a full-chip schematic and congestion view for MAERI-64 (Fig. 6), and a hierarchical sub-module view for MAERI-128 (Fig. 7).

TABLE I Post-Synthesis Results for Maeri-64 And Maeri-128 (45 Nm, 4 Ns Clock Period)

MAERI (PEs)	Comb. Instances	Seq. Instances	Total Area (μm^2)	Internal Power (mW)	Switching Power (mW)	WNS (ns)
64	113,821	33,933	744,132.4	60.00	39.99	+0.2
128	235,345	57,234	1,498,559.1	59.92	40.07	+0.0

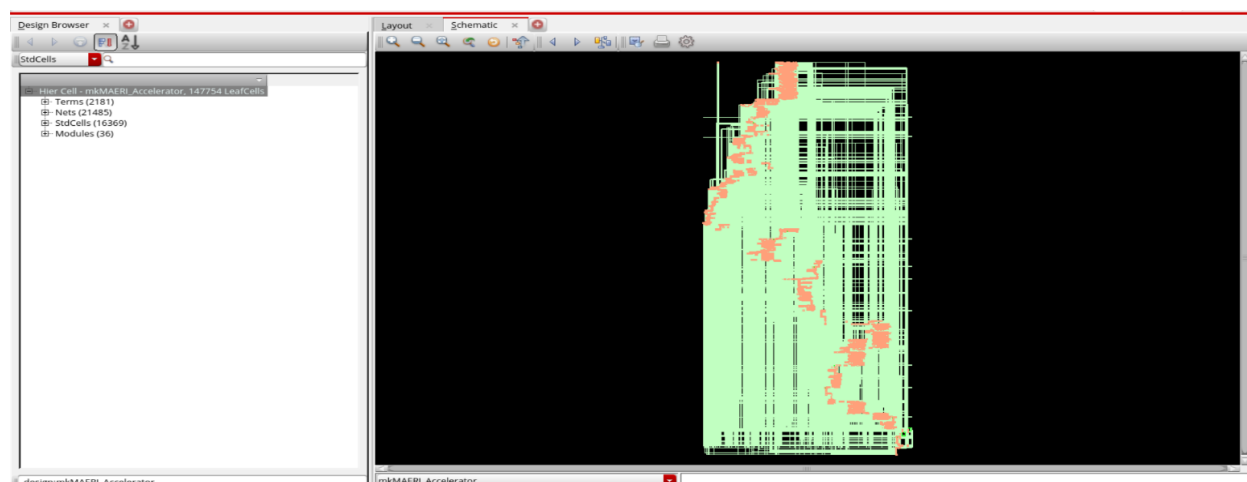


Fig. 6. Full-chip Genus schematic view of MAERI-64 (147,754 leaf cells, 21,485 nets), highlighting congestion (orange) across the die.

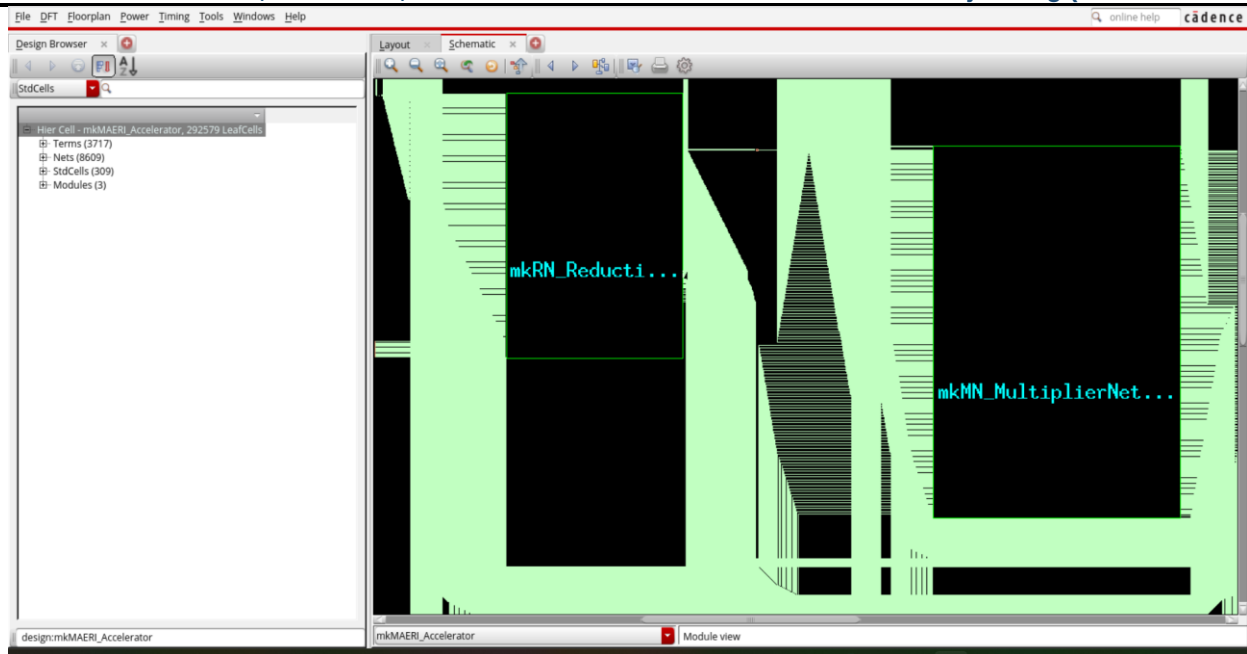


Fig. 7. Post-synthesis hierarchical netlist view of MAERI-128 in Cadence Genus, showing the reduction-network (mkRN_Reduction) and multiplier-network (mkMN_MultiplierNet) sub-modules.

B. Physical Design Results

Tables II and III report area and power at each physical-design stage Floor planning, placement, clock-tree synthesis (CTS) and routing for MAERI-64 and MAERI-128, respectively, obtained from Cadence Innovus. In both configurations, total placed/routed area is lower than the Floor planned area once optimization redistributes cells, while switching power increases from Floor planning to routing as parasitic-aware optimization and clock-network activity are accounted for. Figs. 8 and 9 show the corresponding final placed-and-routed Innovus layouts for MAERI-64 and MAERI-128.

TABLE II Physical Design Results — Maeri-64

Stage	Instances	Total Area (μm^2)	Internal Power (mW)	Switching Power (mW)
Floor planning	147,754	572,379	80.77	19.21
Placement	156,867	525,249	70.42	29.56
CTS	157,233	526,144	66.01	33.97
Routing	157,585	524,223	70.26	29.71

TABLE III Physical Design Results — Maeri-128

Stage	Instances	Total Area (μm^2)	Internal Power (mW)	Switching Power (mW)
Floor planning	292,579	1,172,287	78.79	21.19
Placement	326,124	1,044,794	67.18	32.80
CTS	327,221	1,045,662	67.13	32.85
Routing	327,221	1,045,662	67.72	32.26

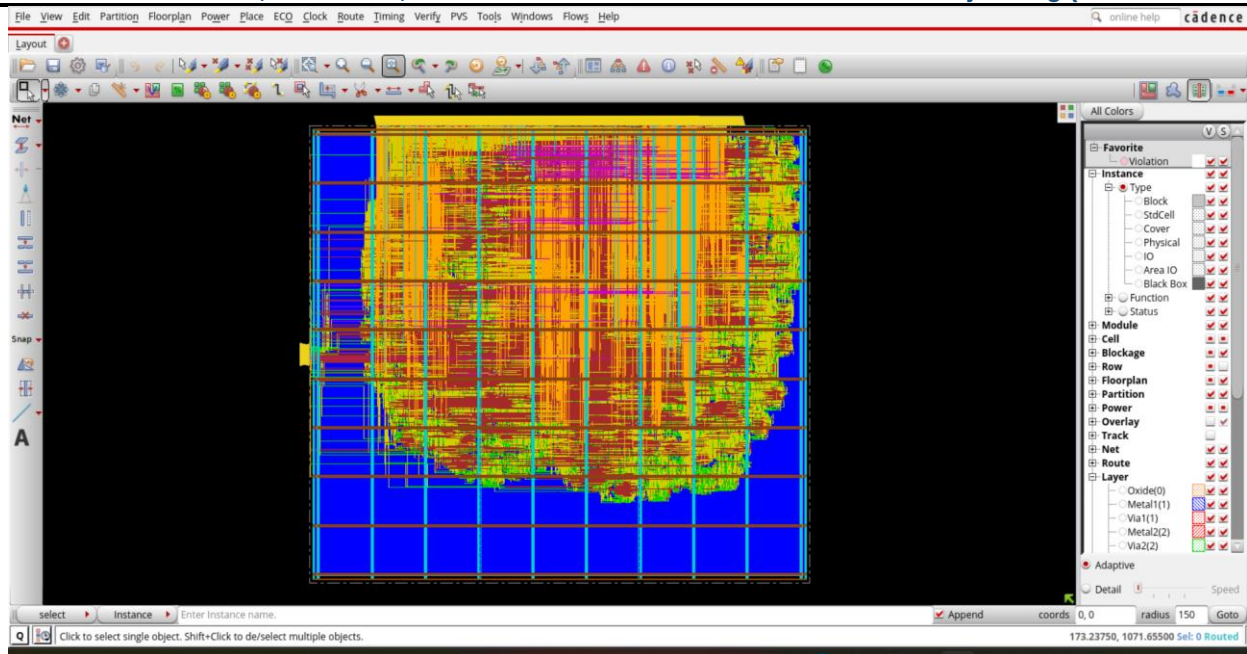


Fig. 8. Final placed-and-routed physical layout of MAERI-64 in Cadence Innovus (post-routing view).

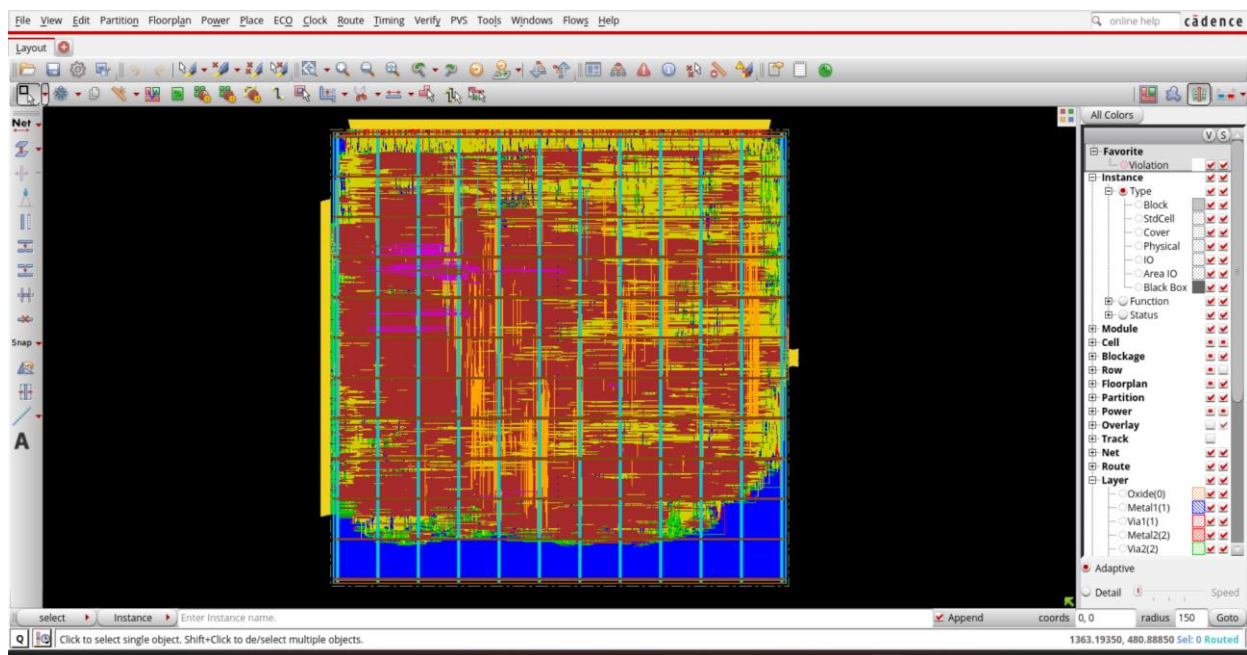


Fig. 9. Final placed-and-routed physical layout of MAERI-128 in Cadence Innovus (post-routing view).

C. Baseline ML Prediction Error Rates

Table IV lists the mean and maximum percentage prediction error of the baseline polynomial regression models for runtime and energy across three representative workloads. Errors are lowest for ResNet-50 (mean 1.9% for runtime, 0.2% for energy) and rise substantially for ResNet-34 and MobileNetV1/GoogLeNet, whose maximum errors exceed 35–78%, indicating that a single global polynomial model does not capture the layer-to-layer variability of all workloads equally well.

TABLE IV Baseline Polynomial Regression Error Rates (Maeri-64 And Maeri-128)

Metric	Algorithm	Workload	Mean Error, Max Error
Runtime	Polynomial Regression	ResNet-50	1.9%, 20.7%
		ResNet-34	10.7%, 78.6%
		MobileNetV1	9.0%, 49.5%
Energy	Polynomial Regression	ResNet-50	0.2%, 5.2%
		ResNet-34	2.5%, 35.0%
		GoogLeNet	4.4%, 47.6%

D. DSE Comparison: MAERI-64 vs. MAERI-128 on ResNet-50

Table V compares the actual ResNet-50 runtime, energy and power of MAERI-64 and MAERI-128 obtained directly from each configuration's real post-physical-design frequency and power (Table I–III) combined with its actual STONNE-simulated cycle counts via Eqs. (1)–(2), and not from the ML models of Section V-D, which are instead evaluated against these true values in Tables IV and VI. MAERI-128 executes ResNet-50 roughly twice as fast as MAERI-64 (120.5 ms vs. 240.0 ms) but draws about double the power (341 mW vs. 170.7 mW), while total energy is nearly identical between the two configurations. This shows that MAERI-

128 is the preferable configuration when runtime/throughput is the priority, whereas MAERI-64 is preferable when minimizing power (and hence area/cost) is the priority, consistent with the general area–performance trade-off reported for other accelerator families in the literature [1], [4]

TABLE V Resnet-50 DSE Comparison: Maeri-64 Vs. Maeri-128 (Actual, Measured Values)

Metric	MAERI-64	MAERI-128
Runtime (ms)	239.964	120.538
Energy (mJ)	40.960	41.103
Power (mW)	170.693	341.0

E. Proposed XGBoost/ExtraTrees Extension

To improve on the baseline polynomial models, XGBoost and ExtraTrees regressors were trained on the same dataset with grouped cross-validation. Table VI reports, for each workload, the configuration/model combination giving the lowest mean runtime and energy error. ResNet-34 achieves the best overall accuracy, with a runtime mean error of 2.52% (MAERI-128 + XGBoost) and an energy mean error of 5.49% (MAERI-128 + ExtraTrees). ResNet-50 achieves a runtime and energy mean error of 6.64% using MAERI-128/MAERI-64 with ExtraTrees, respectively. GoogLeNet and MobileNetV1 show higher errors (14.3–23.7% for runtime and 16.3–16.4% for energy), reflecting their more irregular layer shapes (e.g., 1×1 and depth wise-separable convolutions), which are more difficult for either model family to fit precisely from a training set limited to two PE configurations.

TABLE VI Best Runtime/Energy Prediction Results Per Workload (Proposed Models)

Workload	Best Runtime Config.	Mean Err. (%)	Best Energy Config.	Mean Err. (%)
ResNet-34	MAERI-128 + XGBoost	2.52	MAERI-128 + ExtraTrees	5.49
ResNet-50	MAERI-128 + ExtraTrees	6.64	MAERI-64 + ExtraTrees	6.64
GoogLeNet	MAERI-128 + XGBoost	14.26	MAERI-64 + XGBoost	16.28
MobileNetV1	MAERI-64 + ExtraTrees	23.67	MAERI-64 + XGBoost	16.35

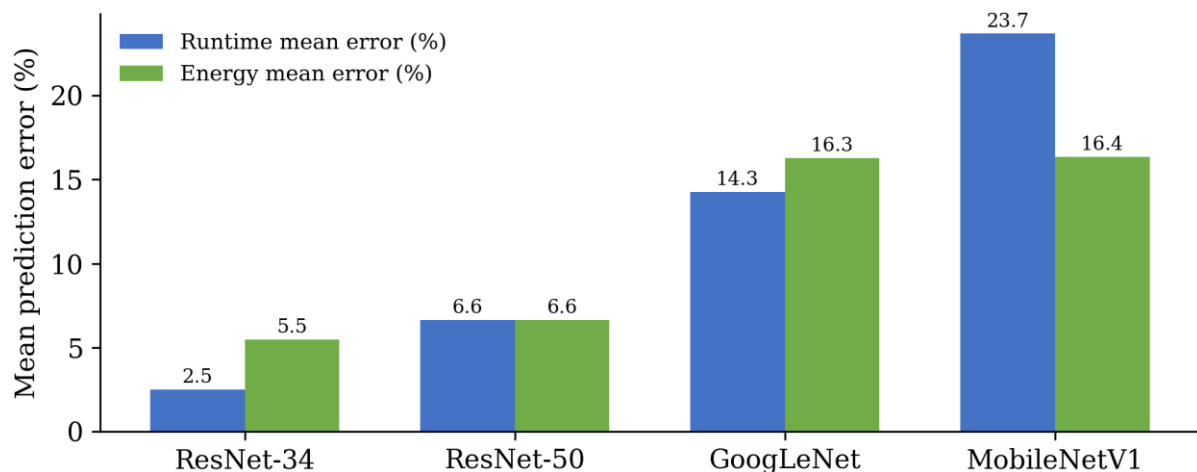


Fig. 10. Mean runtime and energy prediction error (%) of the proposed XGBoost/ExtraTrees models across the four evaluated workloads.

Overall, the ensemble models (XGBoost, ExtraTrees) reduce the worst-case prediction error relative to the polynomial-regression baseline for the more regular workloads (ResNet-34, ResNet-50), confirming that tree-based ensembles better capture MAERI's non-linear, utilization-dependent runtime and energy behaviour than a single global polynomial fit, at the cost of remaining less accurate on the more irregular GoogLeNet and MobileNetV1 workloads.

F. MAERI-256 Predicted Results

Table VII summarizes the predicted hardware characteristics of MAERI-256 alongside the actual post-physical-design values for MAERI-64 and MAERI-128 from which they were extrapolated, as described in Section V-E. The predicted power (681.6 mW) and area (2.09 mm²) for MAERI-256 are each approximately double the corresponding MAERI-128 values, consistent with the near-linear power/area-per-PE trend already observed between MAERI-64 and MAERI-128, while the operating frequency remains at the 0.25 GHz value common to all three configurations, as illustrated in Fig. 11.

TABLE VII Predicted Hardware Characteristics Of Maeri-256 (Linear Extrapolation from Maeri-64/128)

MAERI Configuration	Frequency (GHz)	Power (mW)	Area (μm^2)
MAERI-64 (actual)	0.25	170.693	524,223.4
MAERI-128 (actual)	0.25	341.0	1,045,662.6
MAERI-256 (predicted)	0.25	681.61	2,088,541.0

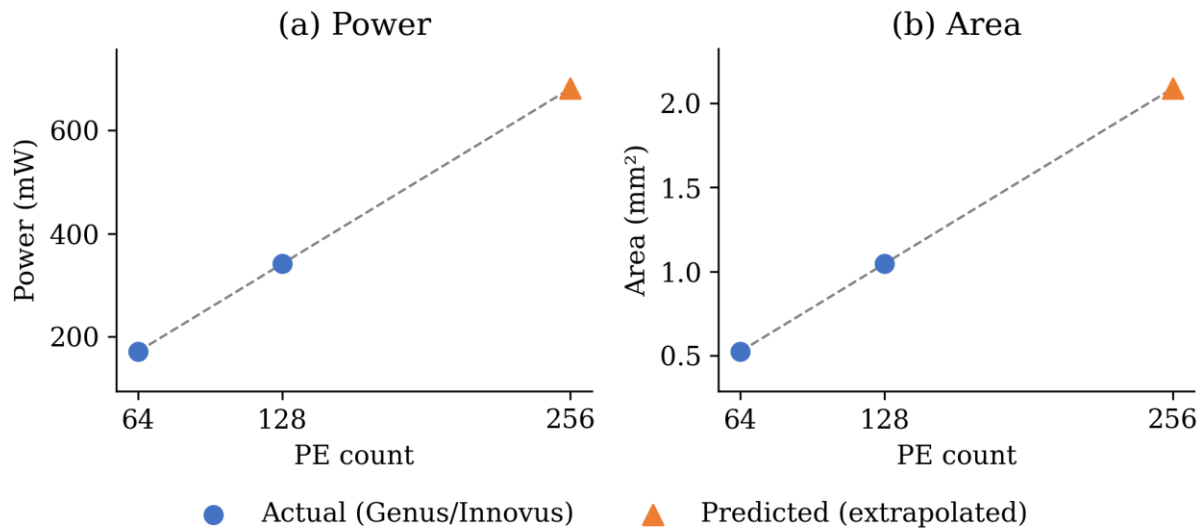


Fig. 11. Linear extrapolation of (a) power and (b) area from the actual MAERI-64/128 post-physical-design values to the predicted MAERI-256 operating point.

Table VIII reports the predicted per-workload runtime and energy for MAERI-256, obtained by combining this extrapolated frequency and power with actual STONNE architectural-simulation cycle counts for MAERI-256 across the same five CNN workloads used elsewhere in this study. As expected from the higher PE count, MAERI-256 workload runtimes range from 7.99 ms (AlexNet) to 63.73 ms (ResNet-50), as shown in Fig. 12, tracking the same relative workload ordering already seen for MAERI-64 and MAERI-128 in Sections VI-A–VI-D, with ResNet-50 and ResNet-34 remaining the two most compute-intensive workloads and AlexNet the least. Because no physical design of MAERI-256 has been performed, the figures in Tables VII and VIII are extrapolated projections rather than measurements, and should be read as an illustration of the proposed methodology's ability to reach beyond its physically implemented training points rather than as validated silicon-level results.

TABLE VIII Maeri-256 Predicted Per-Workload Runtime and Energy (Frequency = 0.25 GHz, Power = 681.6 Mw)

Workload	Total Cycles	Runtime (ms)	Energy (mJ)
AlexNet	1,996,928	7.988	5.445
GoogLeNet	5,394,329	21.577	14.707
MobileNetV1	4,147,376	16.590	11.308
ResNet-34	11,895,274	47.581	32.432
ResNet-50	15,932,338	63.729	43.439

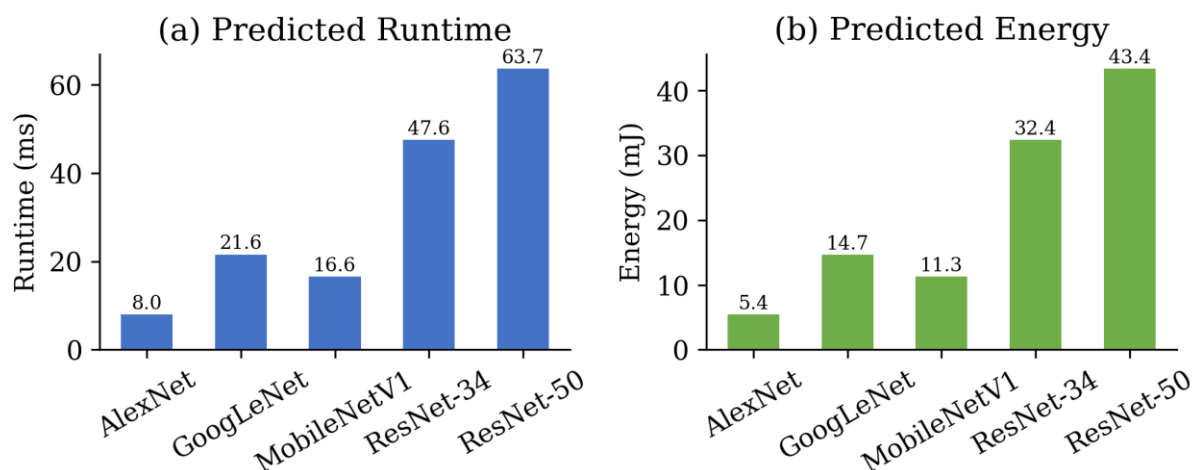


Fig. 12. MAERI-256 predicted (a) runtime and (b) energy for the five evaluated CNN workloads, computed from actual STONNE simulation cycle counts combined with the extrapolated frequency and power of Table VII.

VII. CONCLUSION AND FUTURE SCOPE

This work presented an end-to-end, ML-assisted methodology for predicting the runtime and energy of the MAERI DNN accelerator directly from architectural features, trained and validated against the actual, measured runtime and energy of two physically implemented PE configurations (64 and 128 PEs) synthesized in 45 nm and physically implemented in 90 nm technology. MAERI RTL was generated from Bluespec sources using BSC, synthesized and physically implemented using Cadence Genus and Innovus to obtain area, power and timing, and independently simulated for five CNN workloads inside a Ubuntu 26.04 VirtualBox environment to obtain per-layer execution cycles; combining these actual frequency/power and cycle-count values via Eqs. (1)–(2) gives the true MAERI-64/128 runtime and energy reported in Section VI-D, against which the ML models are evaluated rather than substituted. The merged 355-sample, 26-feature dataset was used to train polynomial regression, ElasticNet and gradient boosting baseline models, and subsequently extended with XGBoost and ExtraTrees regressors. The extended models reduced runtime and energy prediction error to as low as 2.5% and 5.5% respectively for ResNet-34, while remaining less accurate (14–24%) on more irregular workloads such as GoogLeNet and MobileNetV1. The resulting DSE comparison confirms that MAERI-128 is preferable for latency-critical applications, while MAERI-64 is preferable when power and area are the dominant constraints. The methodology was further extended to a third, unimplemented configuration, MAERI-256, by linearly extrapolating frequency, power and area from the MAERI-64/128 physical-design data and combining these with actual STONNE-simulated cycle counts to predict, rather than measure, its per-workload runtime and energy, ranging from 7.99ms (AlexNet) to 63.73ms (ResNet-50), and a corresponding predicted power and area of 681.6 mW and 2.09 mm².

Because no physical implementation of MAERI-256 exists, the linear extrapolation of its power and area, and the frequency held constant across all three configurations, remain unvalidated projections; a natural next step for future work is to carry MAERI-256 through the same Genus/Innovus physical-design flow used for MAERI-64 and MAERI-128 in this study, so that these extrapolated values and the trained XGBoost/ExtraTrees models themselves, as an alternative to a simple two-point linear fit can be checked directly against measured silicon-level data at this larger PE count. Additional future directions include broadening the workload set beyond convolutional layers to attention and fully connected layers relevant to transformer-based models, incorporating additional hardware configurations and technology nodes to improve model generalization, and comparing further ensemble and deep-learning-based regressors against XGBoost/ExtraTrees to develop a fully automated ML-based DSE tool for identifying low-runtime, low-energy MAERI configurations.

REFERENCES

- [1] G. Murali, M. G. Park, and S. K. Lim, “3DNN-Xplorer: A machine learning framework for design space exploration of heterogeneous 3-D DNN accelerators,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 33, no. 2, pp. 358–370, Feb. 2025.
- [2] H. Kwon, A. Samajdar, and T. Krishna, “A communication-centric approach for designing flexible DNN accelerators,” *IEEE Micro*, vol. 38, no. 6, pp. 25–35, Nov. 2018.
- [3] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
- [4] Y. Yu, Y. Li, S. Che, N. K. Jha, and W. Zhang, “Software-defined design space exploration for an efficient DNN accelerator architecture,” *IEEE Trans. Comput.*, vol. 70, no. 1, pp. 45–56, Jan. 2021.
- [5] H. Li, M. Bhargav, P. N. Whatmough, and H.-S. P. Wong, “On-chip memory technology design space explorations for mobile deep neural network accelerators,” in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, New York, NY, USA, Jun. 2019, pp. 1–6.
- [6] H. Esmailzadeh et al., “Physically accurate learning-based performance prediction of hardware-accelerated ML algorithms,” in *Proc. ACM/IEEE 4th Workshop Mach. Learn. CAD (MLCAD)*, Sep. 2022, pp. 119–126.
- [7] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, “SCALE-Sim: Systolic CNN accelerator simulator,” *arXiv:1811.02883*, 2018.
- [8] F. Muñoz-Martínez, J. L. Abellán, M. E. Acacio, and T. Krishna, “STONNE: Enabling cycle-level microarchitectural simulation for DNN inference accelerators,” in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Nov. 2021, pp. 201–213.
- [9] N. P. Jouppi et al., “A domain-specific supercomputer for training deep neural networks,” *Commun. ACM*, vol. 63, no. 7, pp. 67–78, Jun. 2020.
- [10] K. Jhansi Rani, “Video Denoising by Deep Learning Algorithm Using CNN Layers with Catalytic Integration of NLM”, *JICR*, Mar 2022.
- [11] H. Kwon, A. Samajdar, and T. Krishna. (2019). Maeri Project. Accessed: May 20, 2023. [Online]. Available: https://github.com/maeriproject/MAERI_bsv
- [12] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017, doi: 10.1109/JPROC.2017.2761740.
- [13] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, “Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019, doi: 10.1109/JETCAS.2019.2910232.
- [14] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, “Timeloop: A systematic approach to DNN accelerator evaluation,” in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2019, pp. 304–315, doi: 10.1109/ISPASS.2019.00042.
- [15] Y. N. Wu, J. S. Emer, and V. Sze, “Accelergy: An architecture-level energy estimation methodology for accelerator designs,” in *Proc. IEEE/ACM Int. Conf. Comput. -Aided Design (ICCAD)*, 2019, pp. 1–8, doi: 10.1109/ICCAD45719.2019.8942149.
- [16] B. Reagen, J. M. Hernández-Lobato, R. Adolf, M. Gelbart, P. Whatmough, G.-Y. Wei, and D. Brooks, “A case for efficient accelerator design space exploration via Bayesian optimization,” in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, 2017, doi: 10.1109/ISLPED.2017.8009208.

- [17] L. Mei, P. Houshmand, V. Jain, S. Giraldo, and M. Verhelst, “Zigzag: Enlarging joint architecture-mapping design space exploration for DNN accelerators,” *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1160–1174, Aug. 2021, doi: 10.1109/TC.2021.3059962.
- [18] X. Zhang, H. Ye, J. Wang, Y. Lin, J. Xiong, W.-M. Hwu, and D. Chen, “DNNEExplorer: A framework for modeling and exploring a novel paradigm of FPGA-based DNN accelerator,” in *Proc. IEEE/ACM Int. Conf. Comput. -Aided Design (ICCAD)*, 2020, Art. no. 9256813, doi: 10.1145/3400302.3415609.
- [19] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-M. Hwu, and D. Chen, “DNNBuilder: An automated tool for building high-performance DNN hardware accelerators for FPGAs,” in *Proc. IEEE/ACM Int. Conf. Comput. -Aided Design (ICCAD)*, 2018, doi: 10.1145/3240765.3240801.
- [20] Y. N. Wu, P.-A. Tsai, A. Parashar, V. Sze, and J. S. Emer, “Sparseloop: An analytical, energy-focused design space exploration methodology for sparse tensor accelerators,” in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2021, pp. 232–234, doi: 10.1109/ISPASS51385.2021.00043.
- [21] T. Mohaidat and K. Khalil, “A survey on neural network hardware accelerators,” *IEEE Trans. Artif. Intell.*, vol. 5, no. 8, pp. 3801–3822, Aug. 2024, doi: 10.1109/TAI.2024.3377147.

Accessed: May 20, 2023. [Online]. Available: https://github.com/maeriproject/MAERI_bsv