3DNN-Xplorer: A Machine Learning Framework for Design Space Exploration of Heterogeneous 3-D DNN Accelerators

Gauthaman Murali, Min Gyu Park, and Sung Kyu Lim, *Fellow, IEEE*

***Abstract*— This article presents 3DNN-Xplorer, the first machine learning (ML)-based framework for predicting the performance of heterogeneous 3-D deep neural network (DNN) accelerators. Our ML framework facilitates the design space exploration (DSE) of heterogeneous 3-D accelerators with a two-tier compute-on-memory (CoM) configuration, considering 3-D physical design factors. Our design space encompasses four distinct heterogeneous 3-D integration styles, combining 28- and 16-nm technology nodes for both compute and memory tiers. Using extrapolation techniques with ML models trained on 10-to-256 processing element (PE) accelerator configurations, we estimate the performance of systems featuring 75–16 384 PEs, achieving a maximum absolute error of 13.9% (the number of PEs is not continuous and varies based on the accelerator architecture). To ensure balanced tier areas in the design, our framework assumes the same number of PEs or on-chip memory capacity across the four integration styles, accounting for area imbalance resulting from different technology nodes. Our analysis reveals that the heterogeneous 3-D style with 28-nm compute and 16-nm memory is energy-efficient and offers notable energy savings of up to 50% and an 8.8% reduction in runtime compared to other 3-D integration styles with the same number of PEs. Similarly, the heterogeneous 3-D style with 16-nm compute and 28-nm memory is area-efficient and shows up to 8.3% runtime reduction compared to other 3-D styles with the same on-chip memory capacity.**

***Index Terms*— AI accelerator, design automation, heterogeneous 3D design, machine learning (ML), physical design.**

I. Introduction

DEEP neural network (DNN) accelerators find diverse applications across multiple domains, including computer vision, natural language processing, and autonomous vehicles. In recent years, DNN workloads have grown significantly, encompassing numerous layers and billions of parameters.

However, 2-D integration faces challenges in accommodating larger on-chip memory, resulting in worse performance and energy consumption due to the reliance on off-chip memories. Three-dimensional integration overcomes this issue by enabling the integration of multitier on-chip memories with large capacities. Academic research [1], [2], [3] and industry demonstrations, such as system on integrated chips (SoIC) [4] and Foveros [5], have successfully demonstrated multitier memory integration.

While the computational capabilities of accelerator systems progress rapidly, memory technology advances at a slower pace. The 3-D IC design addresses this challenge by allowing the integration of compute logic and memory at different technology nodes on a single chip. Heterogeneous 3-D integration offers diverse design possibilities for a single accelerator architecture. However, manually exploring and identifying the most suitable 3-D integration style for different accelerator configurations and workloads can be challenging.

While accelerator simulators provide a faster estimation of performance than performing the actual physical design, cycle-accurate simulators run for several hours to days. Furthermore, these simulators are primarily designed for 2-D systems where input/output features are shuttled between on-chip and off-chip memories due to limited on-chip memory capacity. On the other hand, in 3-D accelerators with larger on-chip memory capacities, output features of an entire DNN layer can fit within the on-chip memory, enabling efficient computation without the need for data transfers on-chip and off-chip memories.

Industrial and academic research efforts have resulted in the development of parameterizable accelerators, such as TPU [6], SIGMA [7], Eyeriss [8], and MAERI [9], which can be customized for specific applications. Although these accelerators have been optimized for 2-D designs, there is a lack of design space exploration (DSE) and optimization techniques tailored for 3-D technology. Our research aims to address this gap using a machine learning (ML)-based performance prediction framework for 3-D compute-on-memory (CoM) heterogeneous accelerators.

The contributions of this article are as follows.

1) We present 3DNN-Xplorer, the first ML-based framework for DSE of heterogeneous CoM 3-D accelerators, providing a reliable and close approximation to the actual physical design process.
2) We train frequency, power, runtime, and energy models using 16, 32, and 64 processing element (PE) MAERI and systolic array configuration with row and column size of 4, 8, and 16 and perform extrapolation to predict

Received 22 March 2024; revised 31 July 2024 and 2 September 2024; accepted 19 September 2024. This work was supported in part by the Semiconductor Research Corporation through the JUMP 2.0 Center Program (CHIMES 3136.002) and in part by the Samsung Advanced Institute of Technology (SAIT) through the AI for Semiconductors Program. *(Corresponding author: Gauthaman Murali.)*

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: gauthaman@gatech.edu; mgpark@gatech.edu; limsk@ece.gatech.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2024.3471496.

Digital Object Identifier 10.1109/TVLSI.2024.3471496

the performance of 128, 256, 512, 1024, and 2048 PE MAERI and systolic array design with 16, 32, 64, and 128 row and column size with a 13.9% maximum prediction error.

3) Our DSE shows that among the four integration styles combining 28- and 16-nm tech nodes, the style with 28-nm compute and 16-nm memory is energy-efficient, offering up to 50% energy savings and 8.8% runtime reduction over other 3-D designs with the same number of PEs.
4) The heterogeneous 3-D style with 16-nm compute and 28-nm memory is area-efficient and offers up to 8.3% runtime reduction over other 3-D designs with the same on-chip memory capacity.

II. Background and Related Works

Several studies have proposed different architectures and integration techniques for 3-D ML accelerators [3], [10], [11]. While 3-D accelerators offer inherent performance benefits compared to their 2-D counterparts, optimizing the architecture, integration approach, and workload dataflow can yield highly energy-efficient accelerators.

DSE techniques are commonly used to explore the extensive range of design parameters for 2-D accelerators. For example, Esmaeilzadeh et al. [12] proposed a technique that uses automatic ML (AutoML) [13] to predict the performance of hardware-accelerated ML algorithms. Their predictive models estimate various performance metrics, including design frequency, chip power, workload runtime, and energy usage for a given 2-D accelerator configuration. While they perform DSE for 2-D accelerators, their study is limited by interpolation techniques and a maximum prediction error rate of 53.61%, suggesting the possibility of better configurations beyond the training set.

In contrast, DSE studies for 3-D accelerators are limited. Mathur et al. [14] explored the thermal-aware design space for 3-D systolic ML accelerators. Their focus was on investigating options for multitier 3-D integration, but the study was restricted to a narrow range of accelerator architectural configurations and lacked performance prediction frameworks for larger configurations. Li et al. [15] conducted DSE on on-chip memory technology for mobile DNN accelerators using 3-D vertical RRAM. However, their study primarily concentrated on memory technology with an iso-throughput accelerator configuration, rather than exploring the overall accelerator design space.

While extensive research exists on optimizing DSE for 2-D accelerators [12], [16], detailed work on 3-D accelerators is limited. In this article, we present a comprehensive approach to DSE for heterogeneous 3-D DNN accelerators, considering various aspects such as accelerator configurations, 3-D integration methodology, compute and memory technology nodes, workload dimensions, clock frequency, power, and energy requirements. Importantly, our technique offers highly accurate performance predictions for larger accelerator designs with minimal training runtime, using insights from smaller designs. In addition to our previous work [17], we have included two different kinds of accelerator architectures and corresponding workloads to further enhance the DSE framework.

III. Design and Simulation Tools Used

The objective of our ML-based training framework is to develop trained models using small accelerators that can accurately predict significant physical design and workload metrics of large ones. This will aid in an efficient DSE of CoM 3-D accelerator designs. We use simple architecture, 3-D integration, and physical design features as input parameters for our training model (as detailed in Section IV-B). The chosen parameters simplify the DSE process, as it is not necessary to perform an initial synthesis or physical design to use the trained model.

A. Benchmark Architecture

We employ three distinct architectures—namely, MAERI [9], Axiline [18], and a conventional systolic array—to develop and showcase a comprehensive strategy for DSE of both homogeneous and heterogeneous 3-D ML accelerators. The three architectures are shown in Fig. 1.

MAERI is a DNN accelerator that boasts flexible/programmable interconnects between PEs, as shown in Fig. 1(a), allowing it to achieve high utilization on both sparse and dense workloads. This flexibility allows dataflow optimizations to be performed per layer, making MAERI highly energy efficient. In addition, MAERI is an open-source DNN accelerator that is highly configurable and includes a sophisticated workload simulator called STONNE [19] that allows for custom dataflow capabilities.

Axiline is a versatile template-based generator designed for shallow ML (SML) hardware acceleration, such as linear regression, logarithmic regression, and recommender systems, identifying common computational kernels in algorithms to construct an optimized pipelined accelerator. It maps the data flow graphs of ML instances to pipeline stages, optimizing for different data dimensions through custom algorithms. This method yields energy-efficient hardware that is tailored for both training and inference across multiple ML algorithms. Axiline is also an easily configurable open-source SML accelerator that comes with a register transfer level (RTL) generator and architecture simulator called VeriGOOD-ML [20]. Fig. 1(b) shows the template-style architecture of Axiline.

systolic array, a specialized accelerator for DNN, exhibits a rigid interconnect topology between PEs, as shown in Fig. 1(c). This topology can be classified into various dataflow paradigms: input stationary, weight stationary, output stationary, and row stationary, each delineated by its unique interconnect structure. Despite the systolic array's compact area footprint, attributed to its simplistic connectivity, it is prone to substantial underutilization, particularly within expansive arrays. Consequently, it is important for designers to navigate the design space of the systolic arrays. The RTL code for systolic arrays, alongside the workload simulator SCALE-Sim, is also open-source and can be easily reconfigured.

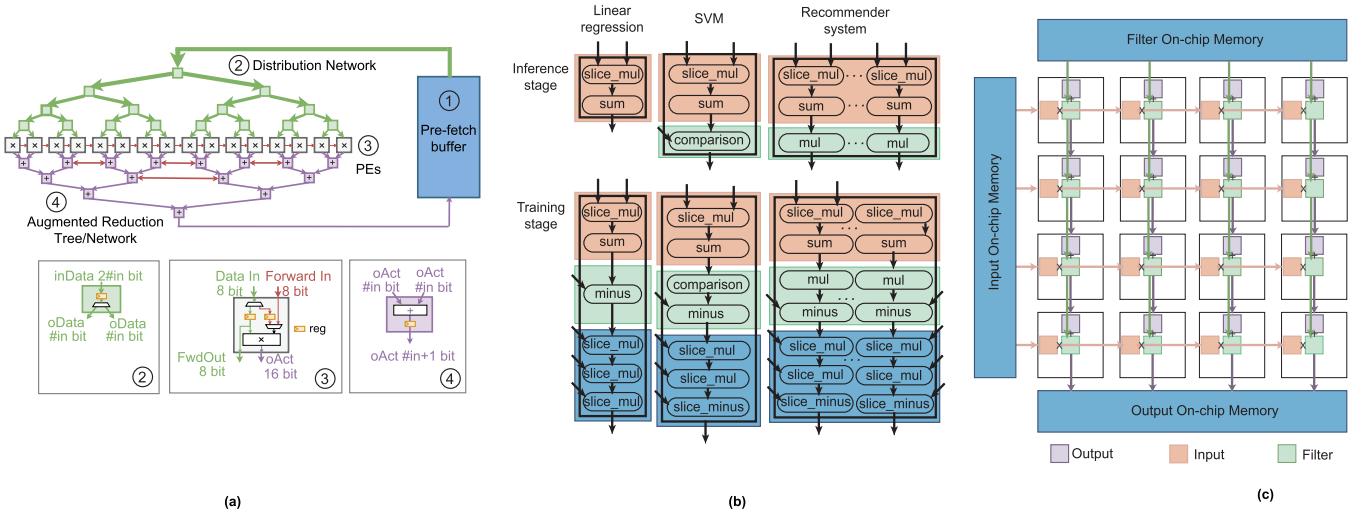

Fig. 1. Benchmark architectures used in this work.

While other accelerators can also be implemented using 3-D technology, we choose the abovementioned accelerators to demonstrate the generic nature of our DSE methodology. The methods we propose in this work are generic and can be applied to any type of ML accelerator.

B. Architectural Simulator

Simulating 3-D accelerator dataflow using a 2-D simulator presents a significant challenge. Three-dimensional ICs offer several advantages, including low access latency and large on-chip memory compared to their 2-D counterparts. These benefits result in reduced energy consumption and runtime, thereby improving the two crucial workload-performance metrics of an accelerator. STONNE [19] for simulating 2-D MAERI, VeriGOOD-ML [20] for Axiline, and SCALE-Sim [21] for 2-D systolic array do not model the advantages of having a large on-chip memory. The simulator handles each layer's simulation as an independent run, resulting in the loss of previously stored outputs in the large on-chip memory. Consequently, it retrieves inputs once again from the DRAM, discarding the results from any previous computations. We overcome this limitation by modeling dataflow between layers, as shown in Algorithm 1.

The upgraded simulation framework with modified dataflow is used to compute the runtime and energy of a given workload as follows.

1) The STONNE simulator [19], VeriGOOD-ML [20], and SCALE-Sim [21] are used to calculate the execution cycles (C) for a given DNN workload layer, assuming that all inputs are stored in the on-chip SRAM buffer.
2) Using the dataflow shown in Algorithm 1, we compute DRAM accesses and use it to modify the total execution cycles (TCs) for a given workload.
3) We obtain the TCs using the following equation:

$$\mathrm{TC} = \mathrm{C} + \mathrm{Accesses}_{\mathrm{DRAM}} \times \mathrm{Latency}_{\mathrm{DRAM}}. \quad (1)$$

4) To calculate the compute logic and SRAM access energy (E) required to execute a given workload, we provide STONNE the compute logic and on-chip memory power obtained from the physical design of the accelerator being simulated.
5) We calculate the total energy (TE) using the following equation:

$$\mathrm{TE} = \mathrm{E} + \mathrm{Accesses}_{\mathrm{DRAM}} \times \mathrm{DE} \quad (2)$$

where DE is the DRAM access energy per byte. We assume a DRAM access energy of 120 pJ/byte as suggested in [14].

Algorithm 1 Dataflow Used in 3-D MAERI Simulation Framework

```
if previous layer outputs fit in on-chip memory then
    read filters from DRAM;
    read previous layer outputs from on-chip memory;
else
    read previous layer outputs & filters from DRAM;
end
if current layer outputs fit in on-chip memory then
    write current layer outputs to on-chip memory;
else
    write current layer outputs to DRAM;
end
```

C. 3-D IC Physical Design and Simulation Tools

To generate the necessary Verilog files, we utilize the MAERI RTL generator [22] and Axiline's VeriGOOD-ML RTL generator. We use a generic systolic array RTL code for the designs presented in this work. Subsequently, we synthesize the netlist using Synopsys Design Compiler (DC). The physical design is then performed using Cadence Innovus. We provide the tool with the floorplan of the memory macros and use the Macro-3D [23] flow to perform the 3-D compute in memory design. Our 3-D designs feature a 3-D back-end of line (BEOL) with six metal layers on each tier. The spacing, width, and RC parasitics of the metal layers are adjusted according to the technology node of the respective tier.

TABLE I
ARCHITECTURAL FEATURES USED IN ML TRAINING. THE BW IS IN BYTES/CYCLE

Accelerator	Feature	Values	Description
MAERI [9]	PEs	16, 32, 64	Total #PEs
	Memory BW	$\frac{\#PE}{4}$, $\frac{\#PE}{2}$	Global buffer bandwidth
Axiline [18]	Processing unit dimensions	10, 25, 50	Total #PEs
	Input BW	#PE	Input buffer bandwidth
	Output BW	#PE×2	Output buffer bandwidth
	Benchmarks	Linear Regression, Logarithmic Regression Support Vector Machine, Recommender Systems	SML benchmarks
Systolic array	ROW	4, 8, 16	Row dimension
	COL	4, 8, 16	Column dimension
	BW	#ROW + 2*#COL	Total bandwidth

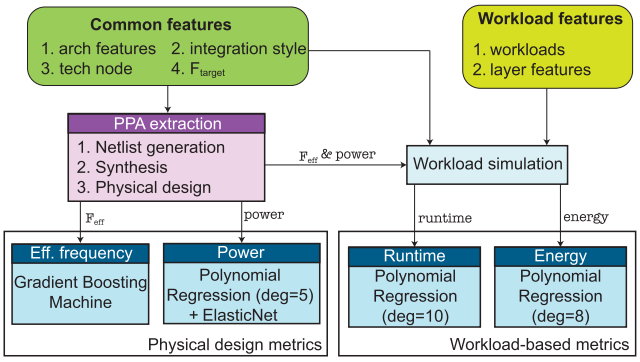

Fig. 2. 3DNN-Xplorer: training framework. The metric models shown are for MAERI architecture.

We extract the timing and power information of the standard cells and memories in the design using Cadence Tempus and use it to calculate the effective design frequency and chip power. We then incorporate these physical design metrics into the STONNE [19], VeriGOOD-ML [20], and SCALE-Sim [21] simulators to obtain accurate runtime and energy metrics for each accelerator configuration.

IV. ML PREDICTION MODEL DEFINITION

The 3DNN-Xplorer framework has four ML models to predict the performance of a given accelerator configuration. This section defines the models and the set of features used to train these models. Fig. 2 shows the models and features used in the 3DNN-Xplorer framework.

A. Four Separate Models Built in This Work

We create prediction models necessary for an easy and reliable DSE of 3-D accelerators. We use four significant performance metrics to evaluate an accelerator configuration.

1) *Effective Design Frequency (Based on Gradient Boosting):* The final operational frequency in gigahertz of a given accelerator configuration.
2) *Chip Power (Based on Polynomial Regression):* The average chip power in milliwatts, assuming an activity of 10%.
3) *Workload Runtime (Based on Polynomial Regression):* The overall time in milliseconds required to run an entire ML workload.

TABLE II
PHYSICAL DESIGN FEATURES USED IN ML TRAINING. AXILINE IS LIMITED TO USING A HOMOGENEOUS INTEGRATION STYLE

Feature	Values	Total	Description
Tech node	16nm, 28nm	2	Tech node of each tier
Integration style	homogeneous, heterogeneous	2	Compute-on-Memory 3D integration style
Frequency	0.1 - 4 GHz	40	Increments of 0.1 GHz

4) *Workload-Specific Energy (Based on Polynomial Regression):* The energy consumption of the design in millijoules to execute an entire workload.

B. Common Input Features

The following parameters are shared as inputs for our four ML-based prediction models. Tables I and II summarize the list of these common input features.

1) Architectural Parameters: The accelerator configuration is determined by the architectural parameters, specifically the number of PEs (#PE) and the total memory bandwidth (BW) of the on-chip SRAM buffer (#BW). For MAERI and Axiline, the memory BW chosen for training depends on the number of #PEs in the design. Our approach involves selecting the most favorable BW values that can provide reasonable throughput while minimizing or completely eliminating memory access stalling. In the case of Axiline, the hardware changes depending on the target workload benchmark, as listed in Table I. For the systolic array, the configuration is determined by the row size (#ROW) and column size (#COL). Based on the given row and column size, we can calculate the number of #PEs as #ROW × #COL and #BW as #ROW + 2×#COL. We use #ROW and #COL as input features for the frequency model and #PE and #BW for the other three prediction models.

2) Technology Nodes: We use two different technology nodes in this work: 16 and 28 nm.

3) Three-Dimensional Integration Style: The 3-D integration offers the advantage of integrating different technology nodes on different tiers. The computation speed and power depend greatly on the technology node of both compute and memory tiers. Furthermore, the capacity of on-chip memory is significantly influenced by the technology nodes and available silicon area of both tiers. In this work, we perform homogeneous and heterogeneous 3-D designs and compare them in terms of power, performance, and area (PPA), workload runtime, and energy. Our 3-D designs involve two tiers,

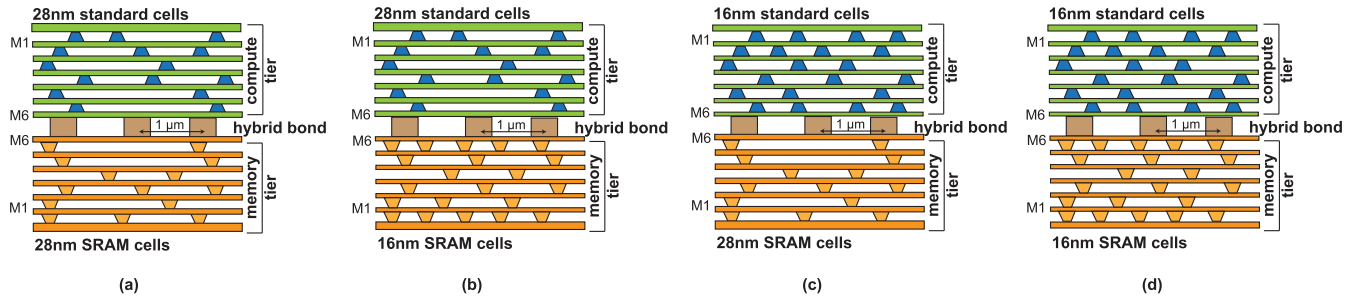

Fig. 3. Four different F2F CoM 3-D integration styles used in this work. We use hybrid bonds of 1-μm pitch and a six-metal layer BEOL in each tier. (a) Homogeneous 3-D 28 nm. (b) Heterogeneous 3-D 28-nm compute/16-nm memory. (c) Heterogeneous 3-D 16-nm compute/28-nm memory. (d) Homogeneous 3-D 16 nm.

TABLE III

DIFFERENT 3-D DESIGN STYLES EXPLORED IN THIS WORK. THE RELATION BETWEEN ON-CHIP MEMORY AND #PES VARIES BASED ON THE TECH NODE TO BALANCE THE COMPUTE AND MEMORY TIER AREAS (SEE TABLES IV AND V)

3D integration style	Compute Node	Memory Node
Homogeneous 1	28nm	28nm
Heterogeneous 1	28nm	16nm
Heterogeneous 2	16nm	28nm
Homogeneous 2	16nm	16nm

TABLE IV

ON-CHIP MEMORY CAPACITY IN kB OF MAERI AND AXILINE

3D integration style	MAERI	Axiline	
		Input	Output
Homogeneous 1	#PE	#PE	2×#PE
Heterogeneous 1	2×#PE	-	-
Heterogeneous 2	#PE/4	-	-
Homogeneous 2	#PE/2	#PE/2	#PE

as shown in Fig. 3: one for compute and one for memory. The two tiers are integrated using a memory-on-logic 3-D physical design methodology called Macro-3D [23]. As we are using two different technology nodes in this study, we construct two variants each of homogeneous and heterogeneous 3-D designs. In the case of Axiline, we only explore homogeneous 3-D designs, as the on-chip memory capacity does not greatly affect the throughput of the accelerator.

Table III shows the four different 3-D integration styles used in this work. We integrate the two tiers in a face-to-face (F2F) fashion using hybrid bonds of 1-μm pitch in homogeneous 3-D (16 or 28 nm) and heterogeneous 3-D (16 and 28 nm) fashions, as shown in Fig. 3. While typically, the memory is at an older node in heterogeneous 3-D designs [3], [24], in this work, we explore a heterogeneous 3-D style with memory at a more recent node than the compute. We chose F2F integration over TSV-based solutions for two main reasons: 1) to align with the current industrial trends of exploring hybrid bond-based 3-D integration solutions and 2) F2F integration offers finer pitches than TSV-based solutions, leading to higher on-chip memory and better PPA, which are crucial to achieving better throughput and energy efficiency in accelerators.

4) Target Frequency (F_{target}): The design frequency is a crucial factor in exploring the design space of any accelerator since it impacts the power consumption, workload runtime, and energy efficiency of the accelerator. We conduct a frequency sweep ranging from 0.1 to 4 GHz, in increments of 0.1 GHz, for every combination of architectural parameters and 3-D integration style. For both the 16- and 28-nm technology nodes, the maximum design frequency of three architecture configurations falls comfortably within the chosen range.

C. On-Chip Memory Capacity

Although there are various approaches to implementing the two tiers in a 3-D IC, we prioritize area balancing as a crucial factor for determining their respective areas. Given the architectural parameters, we choose the total on-chip memory capacity such that the area imbalance between the memory and compute tiers is within 5%.

1) Memory Selection for MAERI: MAERI uses a single on-chip SRAM hierarchy for inputs, filters, and outputs. The overall on-chip memory capacity scales directly with the number of PEs, as we target to balance the core areas between the compute and memory tiers. However, it should be noted that the total memory capacity on the chip varies according to the number of PEs and the technology. The transition from CMOS technology in the 28-nm node to FinFET in the 16-nm node leads to a reduction of more than 50% in the chip area. Consequently, the 28-nm accelerators can accommodate a higher on-chip memory capacity for a given number of PEs. Table IV shows the on-chip memory capacity in the MAERI design for different integration styles.

2) Memory Selection for Axiline: Similar to MAERI, the memory capacity in our Axiline designs also has a direct correlation with #PEs. Axiline uses separate memories for inputs and outputs and as the output BW is 2× that of the input BW, we use 2× memory capacity to store outputs compared to the inputs. Table IV shows the on-chip memory capacity in the Axiline design for different integration styles.

3) Memory Selection for Systolic Array: In contrast to the prior two architectures, the systolic array incorporates distinct on-chip memory units for inputs, filters, and outputs. It dictates that the input memory requires the total BW corresponding to #ROW, while the filter and output memory demand the BW equivalent to #COL. In adherence to the BW requirements, and the prerequisites for logic and area balance, the largest portion of the area is allocated to the input and output memory, with the residual area designated for filter memory. This strategy significantly diminishes the need for DRAM access, consequently improving both the runtime and the energy consumption of the system. Table V shows an example of

TABLE V
ON-CHIP MEMORY CAPACITY IN kB FOR 28-nm HOMOGENEOUS 3-D SYSTOLIC ARRAY

Design size	Input	Filter	Output
8×4	0.125	0.125	0.125
8×8	0.25	0.25	0.25
8×16	1	1.5	1
16×8	2	1	2
16×16	6	2	6
16×32	4	8	4
32×16	16	8	16
32×32	24	16	24

TABLE VI
DNN WORKLOAD-SPECIFIC PARAMETERS USED IN ML TRAINING FOR MAERI AND SYSTOLIC ARRAY

Parameter	Comments
R	filter row dimension
S	filter column dimension
C	Number of input channels
K	Number of filter kernels
X	input row dimension
Y	input column dimension
Strides	Stride of the layer
TC	Input channels mapped per cycle
TK	Filter kernels mapped per cycle

Workloads
ResNet-50
ResNet-34
GoogLeNet
AlexNet
MobileNetv1

on-chip memory capacity allocation for each design size on a 28-nm homogeneous 3-D systolic array.

D. Workload Features

The following parameters are exclusive to the workload-specific runtime and energy models.

1) Workload Parameters: Besides the architectural features, the overall workload execution runtime and energy are also influenced by various parameters related to the workload itself. These parameters include the size of the input features, the size of the filters, the number of channels, the number of kernels, and the stride size. Instead of training the workload-related models on the entire workload, we train them specifically using these features. This approach allows for more flexibility in predicting and extrapolating workload performance to unseen designs and workloads. Table VI lists the workload-specific features used in training the ML-based prediction models of MAERI and the systolic array. Axiline computes runtime and energy numbers directly from the frequency and power numbers. Therefore, in this work, we do not implement any ML models for their prediction.

E. Training Design Space

Combining the features listed in Tables I and II, we perform 960 MAERI, 1920 Axiline, and 800 systolic array design runs for synthesis and physical design using Synopsys dc and Cadence Innovus, respectively. These runs take close to 700 h and were performed on six 2.10-GHz Intel[1] Xeon[1] Gold 6130 servers, using 64 cores in each server. Note that this runtime is close to the design time of a single large MAERI accelerator with 2048 PEs.

[1]Registered trademark.

V. ML MODEL TRAINING METHODOLOGY

This section describes the training techniques used for each model (shown in Fig. 2) to perform an accurate prediction of unseen design configurations by extrapolation. Unseen configurations pertain to any values of the input parameters listed in Table II that were not employed during the training process.

As each performance metric exhibits distinct relationships with each input parameter, we use four different models to predict each metric. We employ exploratory data analysis methods to detect prevalent patterns and detect any anomalies within the training data. Our initial step involves analyzing the distribution of different input variables using graphical approaches, allowing us to draw approximate conclusions about the relationship between the independent and dependent variables. This examination helps to identify whether the relationship is direct or inverse, as well as whether it follows a linear or polynomial pattern. With these preliminary insights, we employ suitable supervised learning techniques to prioritize significant features and eliminate irrelevant ones from the training data. The goal of the trained model is to be able to predict the performance metrics by extrapolating our training sample.

The ML models for effective frequency and chip power are trained using the corresponding metrics obtained from the actual physical design of the 3680 accelerator configurations listed in Tables I and II. Sections V-A and V-B explain the training methodology used for effective frequency and power prediction models, respectively.

While the physical design metrics provide a general understanding of an accelerator's performance, workload performance prediction models offer an application-specific assessment. The overall runtime required to execute a workload is influenced by the effective clock frequency of the accelerator, and the energy consumption depends on the chip power and the DRAM access energy. Hence, workload performance prediction models offer a comprehensive assessment of various accelerator configurations. The ML models for predicting workload execution runtime and energy consumption models are explained in Sections V-C and V-D, respectively

A. Training Frequency Model

We first train the three frequency prediction models using the training dataset for each architecture. Unlike other performance metrics, there is a maximum limit on the effective design frequency. It is impossible to increase the design frequency beyond a certain limit depending on the technology nodes involved. Our training dataset includes the target frequency range necessary to determine this maximum limit for all the integration styles used in this study. Fig. 4(a) shows this target versus effective clock frequency trend in two designs used for training the model.

A significant proportion of the input configurations present in our training dataset exhibits maximum effective frequency saturation at approximately 3 GHz for MAERI, 2.5 GHz for Axiline, and 3.5 GHz for systolic array, as shown in Fig. 4(a). Nevertheless, we extend the target frequency range

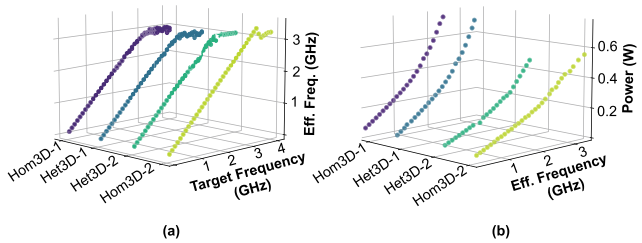

Fig. 4. (a) Frequency and (b) power trend observed in a 32-PE MAERI system across four different integration styles.

to 4 GHz and incorporate the resulting effective frequencies into the training dataset. This approach guarantees that the ML model is trained to accurately predict the saturation limits, considering the architectural features. The gradient boosting algorithm offers excellent extrapolation when trained using an extensive dataset and when the prediction values are within the training range [25]. The effective frequencies of larger designs are always less than those of the smaller designs. Therefore, we use the gradient boosting algorithm in the scikit-learn framework [26] with a depth = 5 and 1000 boosting stages to train the frequency model. We use the Huber function shown in (3) as the loss function (L_{freq}) in the gradient boosting algorithm, with the alpha-quantile (α) set to 0.9

$$L_{\text{freq}} = \begin{cases} \dfrac{\left(F_{\text{act}} - F_{\text{pred}}\right)^2}{2}, & \text{if}\left(F_{\text{act}} - F_{\text{pred}}\right) \le \alpha \\ \alpha\left|\left(F_{\text{act}} - F_{\text{pred}}\right)\right| - \dfrac{\alpha}{2}, & \text{otherwise} \end{cases} \tag{3}$$

where F_{act} is the actual observed effective frequency and F_{pred} is the effective frequency predicted by the model.

The loss function utilized in our frequency model incorporates both absolute and squared prediction errors. The alpha-quantile is set to 0.9 to enhance the robustness of the model against outliers present in the frequency curve, particularly when dealing with values beyond the maximum attainable effective frequency of the design. The accuracy of the trained model is demonstrated by its impressive R2 score of 0.999997 for MAERI, 0.99956 for Axiline, and 0.999992 for the systolic array.

B. Training Power Model

In the case of Axiline, the number of gates in the design scales linearly with the number of PEs. Therefore, the gradient boosting machine (GBM) works well for power prediction in Axiline, as well. With the same GBM model as used for frequency prediction, we trained the Axiline power prediction model and achieved an R2 score of 0.9899.

However, for the other two accelerator architectures, the number of gates in the design increases in a nonlinear fashion. Unlike frequency, power displays a direct yet nonlinear correlation with all the input parameters used in the training process, as shown in Fig. 4(b). Often, ensemble algorithms, such as GBM and random forest (RF), offer a poor prediction for such metrics on unseen configurations [27] (more details in Section V-C and Table VII). Therefore, we employ polynomial regression for training the power prediction model with input features of #PE, #BW, technology nodes, and target frequency. This method typically involves creating polynomial features from the input data in the preprocessing stage, followed by applying linear regression to these features. However, linear regression does not employ any regularization techniques and treats all polynomial features equally, leading to overfitting of the data. Therefore, we perform ElasticNet regression on the polynomial features extracted in our approach.

ElasticNet uses both $L1$ (lasso) and $L2$ (ridge) regularizations, offering a fine balance between feature selection and coefficient regularization. The extracted polynomial features involve higher order and codependent combinations of the input training parameters, such as #PE, F^n_{target}, #PE $\times$ #BW, and #BW $\times$ F_{target}. ElasticNet regression helps prioritizing the most relevant features, leading to highly accurate model generation. The loss function L_{power} of ElasticNet regression is given by the following equation:

$$L_{\text{power}} = \frac{\sum_{i=1}^{n} |P_{i,\text{act}} - P_{i,\text{pred}}|^2}{2n} + \lambda_1 \|\beta\|_1 + 0.5\lambda_2 \|\beta\|_2^2 \tag{4}$$

where $P_{i,\text{act}}$ and $P_{i,\text{pred}}$ are the actual observed and predicted power for a given input configuration i, respectively; $\|\beta\|_1$ and $\|\beta\|_2^2$ represent the $L1$ and $L2$ norms of the polynomial coefficients, respectively; and λ_1 and λ_2 are regularization parameters.

As a tradeoff between feature selection and coefficient regularization, we set $\lambda_1 + \lambda_2 = 0.5$ for MAERI and 0.9 for systolic array and $(\lambda_1/\lambda_1 + \lambda_2) = 0.5$ for MAERI and 0.3 for systolic array using the scikit-learn framework [26]. With the degree of the polynomial feature extraction stage set to 5 and 8, the trained power model has an R2 score of 0.9994 and 0.9906 for each architecture, respectively.

C. Training Runtime Model

The runtime prediction model used in our study incorporates both architectural parameters from Table II and workload-specific parameters described in Table VI. This comprehensive approach improves the accuracy and reusability of our prediction models for runtime estimation of various workloads.

To generate the runtime dataset, we simulate ResNet-50, ResNet-34, AlexNet, GoogLeNet, and MobileNetV1 using STONNE [19] and SCALE-Sim [21]. Axiline estimates runtime and energy numbers directly from the frequency and power numbers using a spreadsheet-based calculator. Therefore, we do not build runtime and energy prediction models for Axiline. Instead, we use the estimated power and frequency numbers from physical design models to directly compute the energy and runtime. For MAERI, its high reconfigurability allows multiple mapping possibilities per convolution layer, leading to suboptimal mappings that underutilize the available PEs in the design. Since our goal was to identify the best design, we specifically simulated mappings that maximized the utilization of all PEs. We adopted a weight stationary flow for the systolic array, ensuring a fixed mapping for each convolution layer.

TABLE VII
ENSEMBLE VERSUS REGRESSION TECHNIQUES FOR RUNTIME PREDICTION. ML FRAMEWORK USED: SCIKIT-LEARN [26]

Ensemble methods				Regression	
Algorithms	Parameter	Range	Max R2 Score	Degree	R2 Score
Ensemble of GBM, RF, DT	n_estimators	[100-1000]	-0.15	6	0.772
	learning_rate	[0.1-1.0]		8	0.978
	max_depth	[3,100]		10	0.998

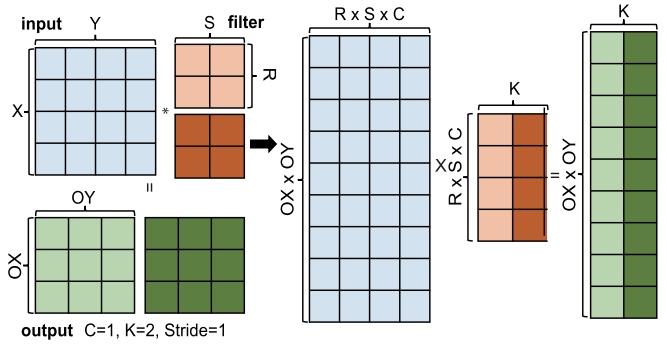

Fig. 5. Example of the matrix multiplication format used to perform convolution operations on a systolic array.

We explored two ML approaches to create a highly accurate model, as detailed in Table VII: 1) stacked ensemble of GBM, RF, and decision trees (DTs); and 2) polynomial regression. Although the ensemble algorithms exhibit strong performance when interpolating within the training dataset, their effectiveness diminishes beyond the convex hull of the dataset [27]. After assessing the model scores, we use the polynomial regression model of degree 10, which exhibits the highest prediction score on extrapolation to new data. To address features having an inversely proportional relationship (#PEs and #BW) with the response variable, we transform them by taking their reciprocals during the training process.

The PE utilization of systolic arrays is significantly affected by variations in workload parameters, row size, and column size, which is a notable difference from MAERI. For example, in the 25th layer of ResNet-50, the utilization rates of 88.6%, 78.7%, and 64.3% were observed for systolic arrays with sizes 8×8, 16×16, and 32×32, respectively. These utilization metrics are directly proportional to runtime. However, the runtime modeling technique used for MAERI does not account for these variations, resulting in mean and maximum errors of 20.9% and 50.9%, respectively.

To address this discrepancy, we propose two novel approaches for the runtime model. Initially, rather than relying on simplistic workload parameters as listed in Table VI, we introduce R $\times$ S $\times$ C, OX $\times$ OY, and K as the input features for modeling workload parameters. This approach is based on the realization that systolic arrays primarily perform matrix multiplication, as depicted in Fig. 5. The chosen input features correspond to the dimensions of the rows and columns of three matrices. Also, to simplify the model's ability to compute the number of tiles, we incorporate (1/#ROW) and (1/#COL) together as input features.

Second, we use utilization as the target value of the prediction model, instead of runtime value. We are then able to calculate the predicted runtime using the following equation:

$$\text{Calculated Runtime on } N \times M \text{ Sys. Array} = \frac{(\text{Runtime on } 8 \times 8) * (\text{Utilization on } 8 \times 8)}{(\text{Predicted Utilization on } N \times M) * \frac{N*M}{64}}. \quad (5)$$

Leveraging the ground truth values for runtime and utilization from smaller systolic arrays, our DSE framework is equipped to estimate runtime for larger arrays. This methodological refinement facilitates a better understanding of runtime calculation and the relationships between systolic array sizes.

Extrapolation of multiple features, as we aim to do in our model, often prevents ensemble algorithms from converging on an optimal solution, even after tuning various training parameters. Taking into account the characteristics of our data and the bounds of extrapolation, the excellent prediction scores achieved by polynomial regression make it a suitable choice for our research.

D. Training Energy Model

We train the energy model to predict the compute energies of the linear network of PEs. Although we can analytically calculate the energies associated with SRAM and DRAM accesses, the compute energies cannot be derived analytically due to the lack of per-cycle activity information. To overcome this limitation, we employ a model that predicts the compute energies with input features of #PE, #BW, technology nodes, and workload parameters in Table VI. Following the methodology outlined in Section III-B and conducting experiments similar to those presented in Table VII, we select a polynomial regression model. Furthermore, the strategy of separating models based on the filter dimensions—specifically distinguishing the 1×1 filter from larger sizes—was adopted for the systolic array. Due to the rigid interconnect in the systolic array, layers with filter size 1×1 suffer from severe underutilization, resulting in longer overall cycles. Therefore, our methodology helps the model to learn two different tendencies on energy consumption.

VI. ML INFERENCE METHODOLOGY

The first step in the prediction flow is to find the effective frequency. We calculate the effective frequency of the new design being explored using the following equation:

$$F_{\text{eff}} = \text{pred\_eff\_frequency}(\text{pe}, \text{bw}, \text{is}, F_{\text{target}})$$
$$\text{pe} = 2^x, x \in [4, 11], \ \text{bw} \in \left\{\frac{\text{pe}}{4}, \frac{\text{pe}}{2}\right\}, \ \text{is} \in \{0, 1, 2, 3\} \quad (6)$$

where F_{eff} is the predicted effective frequency; pe and bw are the total number of PEs and the on-chip memory BW in the design being explored, respectively; is denotes the integer-coded 3-D integration style; and F_{target} is the target frequency. We have tested the prediction model for up to a total #PE count of 2^{11} or 2048, and therefore, we restrict the input variable pe to 2048. For systolic arrays, we use row and col instead of pe and bw. The configurations range from (8, 4) to (128, 128),

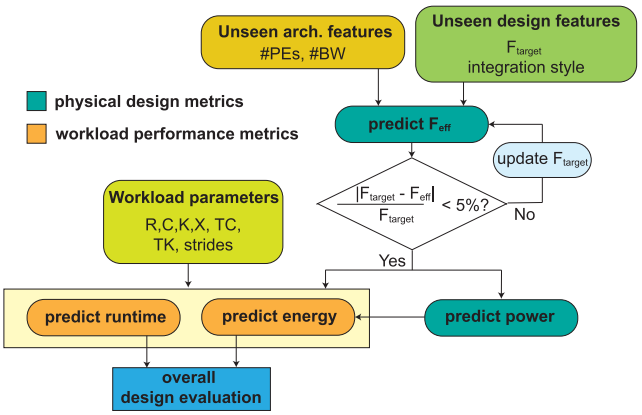

Fig. 6. Inference methodology used in 3DNN-Xplorer to evaluate unseen accelerator configurations during DSE.

TABLE VIII

PREDICTION MODEL ERROR RATES BASED ON RANDOMLY SELECTED 128 AND 256 PE CONFIGURATIONS FOR MAERI ARCHITECTURE

Model	ML Algorithm used	Mean error, Max. Error
Frequency	Gradient Boosting Machine	0.8%, 4.9%
Power	Polynomial Regression	3.1%, 8.3%
Runtime	Polynomial Regression	ResNet-50: 2.8%, 7.7%
		ResNet-34: 3.4%, 8.0%
		GoogLeNet: 1.6%, 8.9%
		AlexNet: 0.1%, 0.12%
		MobileNetv1: 0.5%, 2.0%
Energy	Polynomial Regression	ResNet-50: 7.3%, 12.4%
		ResNet-34: 8.4%, 10.4%
		GoogLeNet: 8.2%, 10.0%
		AlexNet: 6.9%, 10.9%
		MobileNetv1: 8.7%, 12.7%

encompassing various combinations such as (8, 4), (8, 8), (8, 16), ..., (64, 128), (128, 64), and (128, 128).

Based on the prediction methodology shown in [12], we define a frequency range of interest (F_{ROI}) for the prediction models to ensure the maximum accuracy. F_{ROI} is given by the following equation:

$$F_{\text{ROI}} \in \quad \forall F_{\text{eff}} : \frac{|F_{\text{target}} - F_{\text{eff}}|}{F_{\text{target}}} \leq 5\%. \tag{7}$$

If the predicted effective frequency is beyond a 5% margin from the desired target frequency, it becomes futile to investigate that particular design configuration further. We keep updating the target frequency until the predicted effective frequency is within the 5% limit. We then predict the chip power. Subsequently, using the effective frequency, we predict the runtime for one of the five workloads used in training. Using the chip power and frequency values, we use the energy model to predict the overall energy for a chosen workload. Fig. 6 shows the prediction methodology employed in 3DNN-Xplorer. The predicted energy and runtime are the final metrics used to evaluate the selected design configuration.

Our primary objective is to develop resilient prediction models that facilitate precise extrapolation of performance metrics for unseen accelerator configurations. Large accelerator designs demand substantial design time, rendering the DSE for such configurations extremely time-consuming. Using appropriate training strategies tailored to each performance metric, we are able to construct prediction models capable of extrapolating these metrics to designs larger than those exposed to the ML models in the training phase. Tables VIII and IX show the accuracy of our models for MAERI and systolic array, respectively, compared to the actual physical design and simulation data for various metrics. All our models have a maximum error $\leq$ 13.9%.

TABLE IX

PREDICTION MODEL ERROR RATES BASED ON RANDOMLY SELECTED 16 × 32, 32 × 16, AND 32 × 32 CONFIGURATIONS FOR SYSTOLIC ARRAY ARCHITECTURE

Model	ML Algorithm used	Mean error, Max. Error
Frequency	Gradient Boosting Machine	0.5%, 4.6%
Power	Polynomial Regression	5.8%, 13.9%
Runtime	Polynomial Regression	ResNet-50: 5.4%, 11.3%
		ResNet-34: 1.2%, 6.7%
		GoogLeNet: 3.6%, 9.5%
		MobileNetv1: 2.2%, 7.3%
Energy	Polynomial Regression	ResNet-50: 8.3%, 13.4%
		ResNet-34: 6.2%, 11.1%
		GoogLeNet: 7.2%, 11.4%
		MobileNetv1: 9.7%, 13.7%

VII. DSE RESULTS

A. Experimental Setup

We conduct an extensive DSE encompassing the design configurations shown as follows.

Design space explored for MAERI				
#PEs	#BW	Frequency	Nodes	Styles
128, 256, 512, 1024, 2048	$\frac{\#PE}{4}$, $\frac{\#PE}{2}$	0.5-4 GHz	16nm, 28nm	Homogeneous 3D, Heterogeneous 3D

Design space explored for Axiline				
#PEs	#BW	Frequency	Nodes	Styles
75, 100, 150, 200	In: #PE Out: #PE×2	0.5-4 GHz	16nm, 28nm	Homogeneous 3D

Design space explored for Systolic Array				
#ROW	#COL	Frequency	Nodes	Styles
16, 32, 64, 128	16, 32, 64, 128	0.5-4 GHz	16nm, 28nm	Homogeneous 3D, Heterogeneous 3D

By sweeping the target frequency range from 0.5 to 4 GHz, we determine the maximum effective design frequency for each integration style and configuration. Subsequently, we estimate the chip power based on the predicted maximum design frequency. Utilizing the predicted physical design metrics, we estimate the TE consumption (compute + SRAM + DRAM) and runtime to execute ResNet-50 on each configuration using the parameterized workload prediction models.

We perform two different DSEs using the predicted ResNet-50 runtime and energy values.

1) We compare designs with the same number of PEs across different integration styles. In this case, the on-chip memory varies based on the integration style according to the relation given in Tables IV and V.
2) We compare designs with the same on-chip memory across various integration styles. In this case, #PEs differ.

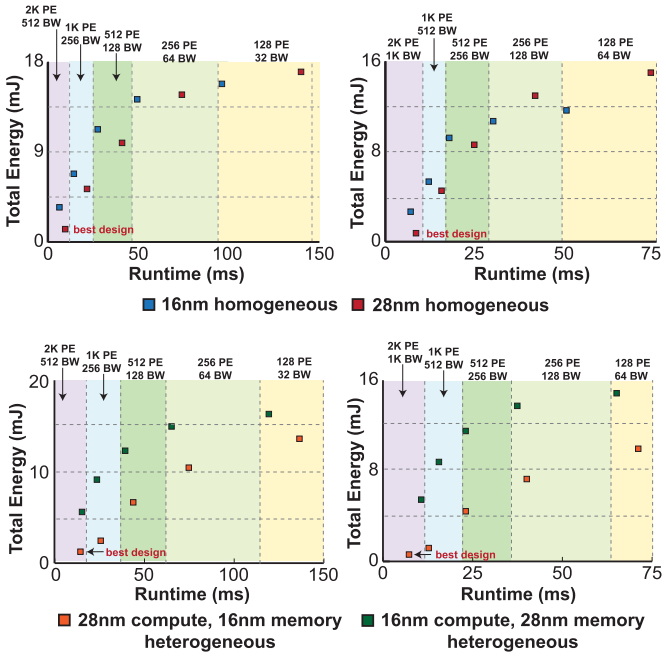

Fig. 7. DSE of the two homogeneous (top) and two heterogeneous (bottom) 3-D MAERI configurations with the same #PEs and #PE/4 (left) or #PE/2 (right) words/cycle bandwidth. The on-chip memory capacity varies based on the tech node. Workload used: ResNet-50.

TABLE X

DSE OF 3-D MAERI CONFIGURATIONS WITH THE SAME ON-CHIP MEMORY CAPACITY. THE #PES VARIES BASED ON THE TECH NODE. WORKLOAD USED: RESNET-50

		Homogeneous 3D						Heterogeneous 3D					
		#PE		Runtime (ms)		Energy (mJ)		#PE		Runtime (ms)		Energy (mJ)	
On-chip memory	#BW	16nm	28nm	16nm	28nm	16nm	28nm	28nm C 16nm M	16nm C 28nm M	28nm C 16nm M	16nm C 28nm M	28nm C 16nm M	16nm C 28nm M
256 KB	$\frac{\#PE}{4}$	512	256	31.24	74.69	10.97	14.24	128	1024	129.86	23.60	14.06	9.44
	$\frac{\#PE}{2}$			18.87	43.44	8.52	12.53			71.14	15.59	9.95	8.78
512 KB	$\frac{\#PE}{4}$	1024	512	18.91	43.81	6.78	9.69	256	2048	70.42	15.34	10.79	5.79
	$\frac{\#PE}{2}$			**12.17**	26.07	**5.4**	8.50			40.07	**11.17**	7.27	**5.4**

We assume that the compute and memory tiers have a balanced total silicon area in both comparisons. This ensures a fair and consistent evaluation of the different configurations, allowing us to isolate and analyze the impact of other design parameters on performance and efficiency. Our approach combines the power of advanced prediction models with the flexibility of DSE, allowing us to make informed decisions regarding integration styles and design parameters.

B. Homogeneous 3-D Integration

This section presents a comparison of runtime and energy consumption for the ResNet-50 workload across different configurations, designed in 16- and 28-nm homogeneous 3-D styles.

1) MAERI: Fig. 7 shows the DSE of homogeneous 3-D MAERI designs with the same number of PEs across different integration styles. Our DSE reveals that the optimal design in the space explored is 28-nm homogeneous 3-D MAERI with 2048 PEs, a BW of 1024 words/cycle, and 2-MB on-chip memory. This configuration achieves a lower energy consumption of 1.22 mJ and a runtime of 9.35 ms on the ResNet-50 workload. While the runtime of the 16-nm homogeneous 3-D design with 2048 PEs is better than that of its 28-nm counterpart, it uses almost 5× more energy to execute ResNet-50. This is due to the fact that the 16-nm 2048 PE design has half the on-chip memory capacity of its 28-nm counterpart.

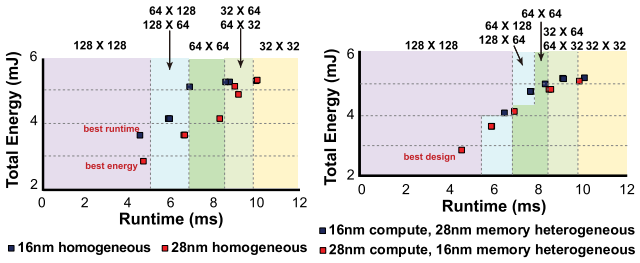

Fig. 8. DSE of the two homogeneous (left) and two heterogeneous (left) 3-D systolic array configurations with the same #ROW and #COL. The on-chip memory capacity varies based on the tech node. Workload used: ResNet-50.

In contrast, the exploration focusing on maintaining the same on-chip memory capacity (see Table X) identifies 16-nm homogeneous 3-D MAERI as a better design over corresponding 28-nm designs as it offers better energy and runtime.

Overall, the 28-nm 3-D MAERI are more energy-efficient than the 16-nm ones for #PEs $\geq$ 512. Conversely, the smaller area and shorter runtime of the 16-nm 3-D MAERI make them more area-efficient compared to the 28-nm MAERI.

TABLE XI
DSE OF 3-D SYSTOLIC ARRAY CONFIGURATIONS WITH THE SAME ON-CHIP MEMORY CAPACITY. #ROW AND #COL VARY BASED ON THE TECH NODE. WORKLOAD USED: RESNET-50

		Homogeneous 3D						Heterogeneous 3D					
		(#ROW, #COL)		Runtime (ms)		Energy (mJ)		(#ROW, #COL)		Runtime (ms)		Energy (mJ)	
Input memory	Output memory	16nm	28nm	16nm	28nm	16nm	28nm	28nm C 16nm M	16nm C 28nm M	28nm C 16nm M	16nm C 28nm M	28nm C 16nm M	16nm C 28nm M
64 KB	64 KB	(64, 64)	(32, 64)	6.88	8.74	5.05	5.07	(32, 32)	(64, 64)	8.75	8.29	5.06	5.05
256 KB	256 KB	(64,128)	(64,64)	**5.88**	8.28	**4.10**	4.11	(64, 64)	(128, 128)	6.90	**6.39**	4.13	**4.08**

2) Axiline: In the case of Axiline, we use support vector machine (SVM) as the benchmark workload to perform the DSE. As SVM is a shallow model with a single layer, the amount of on-chip capacity does not affect the performance of the accelerator significantly. Therefore, an Axiline system of 200 dimensions is the most energy and runtime-efficient for both homogeneous 28- and 16-nm 3-D designs. They offer an execution runtime of 0.36 and 0.27 ms, and energy of 33 and 26 μJ on executing SVM with 200 weights, 500 000 feature vectors, and 200 model topology.

3) Systolic Array: Results for homogeneous 3-D systolic arrays are depicted in Fig. 8. Our design space highlights two Pareto optimal configurations at 28- and 16-nm homogeneous, each with a systolic array size of 128 $\times$ 128. These designs allocate on-chip memory for input, filter, and output data as 1 MB, 64 kB, and 1 MB for the 28-nm homogeneous design, and 0.5 MB, 64 kB, and 0.5 MB for the 16-nm homogeneous design, respectively. In evaluating these designs under the ResNet-50 workload, the 28-nm homogeneous design demonstrates better energy with a consumption of 2.84 mJ and a runtime of 4.73 ms. For a 16-nm homogeneous design, benefiting from its higher maximum performance reduces the runtime to 4.57 ms at the cost of increased energy consumption at 3.62 mJ.

Further analysis, as presented in Table XI, maintains constant input and output on-chip memory sizes across designs. It reveals that the 64 $\times$ 128 sized 16-nm 3-D systolic array with 256-kB input and output on-chip memory is the best design in both runtime and energy. In summary, a 28-nm 3-D systolic array emerges as more energy-efficient due to its lower power consumption, while the 16-nm design exhibits higher area efficiency, attributed to its reduced physical footprint.

C. Heterogeneous 3-D Integration

1) MAERI: Fig. 7 shows the DSE of heterogeneous 3-D MAERI with the same number of PEs for various integration styles. Among all heterogeneous 3-D designs, the design with 28-nm compute and 16-nm memory, featuring 2048 PEs, a BW of 1024 words/cycle, and an on-chip memory capacity of 4 MB is the most optimal design. This design executes ResNet-50 in 7.25 ms with an energy consumption of 0.61 mJ.

Conversely, when maintaining the same memory capacity across the two heterogeneous styles (see Table X) and striving for area balance between tiers, the heterogeneous style with 16-nm compute and 28-nm memory emerges as the most favorable option. This configuration entails 2048 PEs, a BW of 1024 words/cycle, and an on-chip memory capacity of 512 kB. It offers the shortest runtime of 11.17 ms and a low energy consumption of 5.4 mJ to execute ResNet-50 among other designs with the same on-chip memory. In summary, the heterogeneous style with 28-nm compute exhibits higher energy efficiency, while the other style with 16-nm compute demonstrates superior area efficiency.

2) Systolic Array: The exploration of design spaces for heterogeneous 3-D systolic arrays, as depicted in Fig. 8, indicates a variety of configurations across different sizes. Among these, the heterogeneous 3-D systolic array design comprising a 128 $\times$ 128 configuration with 28-nm compute and 16-nm memory emerges as the most efficient. It shows exemplary performance on the ResNet-50 workload with a runtime of 4.51 ms and an energy consumption of 2.86 mJ.

In contrast, analysis with fixed on-chip memory capacity, detailed in Table XI, identifies the 128 $\times$ 128 sized heterogeneous 3-D systolic array, utilizing 16 nm for compute and 28 nm for memory, as the optimal design in terms of both runtime and energy. This configuration, which includes a substantial on-chip memory capacity of 256 kB and extensive logic area, achieves significant reductions in energy and runtime, consuming 4.08 mJ and completing tasks in 6.39 ms, respectively.

From an overall perspective, the heterogeneous 3-D systolic array with 28-nm compute and 16-nm memory stands out for its energy efficiency. In comparison, the alternative heterogeneous configuration, which leverages 16 nm for compute and 28 nm for memory, is distinguished by its area efficiency.

D. Homogeneous Versus Heterogeneous 3-D Integration

In this section, we compare the energy- and area-efficient designs of homogeneous integration styles against their heterogeneous versions. In the case of Axiline, we do not have any heterogeneous 3-D designs and the 16-nm homogeneous 3-D design of dimension 200 is both energy- and area-efficient.

1) MAERI: The energy-efficient heterogeneous design with 28-nm compute and 16-nm memory, featuring 2048 PEs and 1024 words/cycle BW, achieves 50% energy savings and an 8.8% reduction in runtime over the 28-nm homogeneous 3-D design with the same number of PEs. The area-efficient heterogeneous design with 16-nm compute and 28-nm memory, featuring 512-kB on-chip memory and 1024 words/cycle BW, demonstrates an 8.3% runtime improvement over its area-efficient 16-nm homogeneous counterpart. These findings highlight the superior performance of the heterogeneous 3-D designs, showcasing their potential for achieving both energy savings and runtime improvements.

2) Systolic Array: The energy-efficient 3-D systolic array is heterogeneous 3-D design with 28-nm compute and 16-nm memory. This particular arrangement, which features a 128 $\times$ 128 array size coupled with 3-MB on-chip memory capacity, stands out for its performance metrics and delivers a 3.3% reduction in runtime compared to a 28-nm homogeneous 3-D

TABLE XII

PPA ANALYSIS OF THE BEST CONFIGURATIONS. RUNTIME AND ENERGY METRICS ARE BASED ON RESNET-50 WORKLOAD FOR MAERI AND SYSTOLIC ARRAY AND BASED ON SVM FOR AXILINE. THE ACTUAL RUNTIME AND ENERGY NUMBERS OBTAINED POST-PHYSICAL DESIGN ARE SHOWN IN BLUE

Metric	Energy-efficient 3D MAERI	Area-efficient 3D MAERI	Energy/Area-efficient 3D Axiline	Energy-efficient 3D Sys. Array	Area-efficient 3D Sys. Array
Compute node	28nm	16nm	16nm	28nm	16nm
Memory node	16nm	28nm	16nm	16nm	16nm
#PEs	2048	2048	200	128×128	16×16
#BW (words/cycle)	1024	1024	In:200, Out:400	384	48
On-chip mem (MB)	4	0.5	0.3	3	20/1024
Chip area (mm^2)	15.2	6.7	1.04	6.25	0.041
Max. freq. (GHz)	1.67	1.94	1.81	1.59	3.83
Chip power (W)	28.3	18.5	2.31	12.44	0.251
Runtime (ms)	7.0 (7.25)	10.4 (11.16)	0.27 (0.31)	4.58 (4.75)	12.1 (12.5)
Energy (mJ)	0.62 (0.61)	5.35 (5.46)	0.026 (0.024)	2.86 (2.89)	5.25 (5.28)
Energy eff. ($\frac{TOPS}{W}$)	**19.4**	2.24	**0.31**	**12.3**	1.96
Area eff. ($\frac{TOPS}{mm^2}$)	0.11	**0.18**	**0.69**	1.22	**20.4**

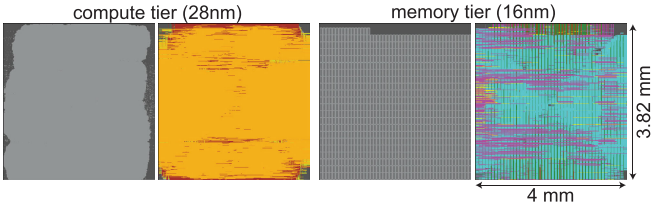

Fig. 9. Final layouts of energy-efficient 2048 PE 3-D MAERI.

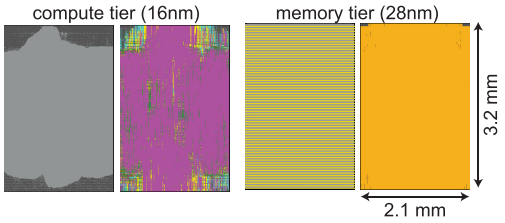

Fig. 10. Final layouts of area-efficient 2048 PE 3-D MAERI.

design and a 20.9% decrease in energy consumption compared to a 16-nm homogeneous 3-D design, maintaining the same array dimensions. On the other hand, the area-efficient 3-D systolic array is 16-nm homogeneous 3-D design. This design features an array size of 16 × 16 and is equipped with 20 kB of on-chip memory. Unlike the MAERI, the systolic array's area efficiency does not scale linearly with its size. This phenomenon is attributed to the significant underutilization of PEs in larger arrays. The underutilization leads to a scenario where the increase in TOPs does not match the proportional expansion in the number of PEs, whereas the chip area continues to scale directly with the array size.

We perform the physical design of the best configurations of the accelerators. Figs. 9 and 10 show the final layouts of the most energy- and area-efficient 3-D MAERI designs, respectively. Performance analyses of these designs are summarized in Table XII. These results align with the predictions of our DSE study. The actual energy and runtime numbers are very close to our predictions and are shown in blue in Table XII. Integrating 16-nm memories on top of 28-nm compute logic allows having large on-chip memory, which reduces the number of DRAM accesses and, thereby, the overall computation energy.

E. Thermal and Power-Delivery Considerations

3DNN-Xplorer is a physical and technology-aware simulation framework that does not consider power and thermal issues in heterogeneous 3-D IC designs in detail. Our training designs use different supply voltages for 28- and 16-nm nodes and the conversion is assumed to be handled outside the logic and memory tiers. While realizing the actual design, the strategies detailed in [28] can help realistically implement this assumption and achieve performance results closer to those estimated from our simulation framework.

VIII. LIMITATIONS AND FUTURE SCOPE OF 3DNN-XPLORER

Only the RTL of our designs is open-source, not the GDS, as we are using commercial PDKs for our designs. Hence, the dataset is not publicly available. However, the methodology explained in this article can be used to validate the authenticity of this work.

The 3DNN-Xplorer framework currently supports three distinct types of accelerators:

1) rigid interconnect or systolic array-like architectures—such as TPU, Eyeriss, and Intel Gaudi;
2) flexible interconnect architectures—such as MAERI and SIGMA;
3) template-based application-specific accelerators—such as Axiline.

While incorporating a new accelerator topology into the framework requires additional training, particularly when the architecture diverges significantly from those already supported, the current methodology remains adaptable to emerging accelerator technologies. Future work will focus on devising a more resource-efficient approach, aiming to minimize dataset generation time and model training.

3DNN-Xplorer effectively evaluates the performance of convolution layers, which are fundamental to CNNs. However, beyond image-related workloads, accelerators are increasingly utilized in diverse domains such as natural language processing and graph processing, each with distinct core building blocks. For instance, attention layers in transformer-based large language models, primarily composed of fully connected

layers and requiring KV caching, exhibit significantly different computational and memory access characteristics compared to convolution layers. Future work will extend the framework to accommodate a wider range of workloads.

Furthermore, the 3-D designs used in training involve F2F integration, limiting the exploration study to two-tier 3-D designs. In the future, TSV-based designs could be considered to build multitier 3-D designs and evaluate them.

IX. Conclusion

In this study, we conducted an extensive DSE of CoM 3-D accelerators, encompassing both flexible and rigid architectures, and designed using homogeneous and heterogeneous 3-D integration styles. To navigate this vast design space, we introduced an ML-based performance prediction framework called 3DNN-Xplorer. This framework facilitated the evaluation of various accelerator configurations and integration styles, delivering reliable and robust performance estimates. By training prediction models on different smaller accelerators, we achieved highly accurate performance extrapolation for larger systems. We have also proposed methods to manipulate performance metrics for rigid architectures, where utilization is not always 100%, to achieve better and more accurate performance predictions. Our findings identified the most optimal design configurations for both homogeneous and heterogeneous approaches while maintaining the same number of PEs or on-chip memory capacity. In particular, heterogeneous integration styles emerged as exemplary configurations, offering significant energy savings and runtime reductions compared to their homogeneous counterparts.

Furthermore, our results highlighted the tradeoff between energy efficiency and runtime performance when considering different technology nodes. The 28-nm 3-D accelerators demonstrated higher energy efficiency, while 16-nm accelerators exhibited the improved runtime performance. Therefore, heterogeneous integration offers an excellent tradeoff between runtime and energy efficiency. Overall, our study contributes valuable insights to the field of 3-D accelerator design, advancing our understanding of both homogeneous and heterogeneous 3-D integration styles for designing accelerators. The methodologies presented in this work are generic and can be applied to any DNN accelerator design.

References

[1] P. Shukla, V. F. Pavlidis, E. Salman, and A. K. Coskun, "Temperature-aware monolithic 3D DNN accelerators for biomedical applications," in *Proc. Design, Autom. Test Eur. Conf. (DATE)*, 2022, pp. 1–3.

[2] W. Lu, P.-T. Huang, H.-M. Chen, and W. Hwang, "An energy-efficient 3D cross-ring accelerator with 3D-SRAM cubes for hybrid deep neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 4, pp. 776–788, Dec. 2021.

[3] G. Murali, X. Sun, S. Yu, and S. K. Lim, "Heterogeneous mixed-signal monolithic 3-D in-memory computing using resistive RAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 2, pp. 386–396, Feb. 2021.

[4] M.-F. Chen, F.-C. Chen, W.-C. Chiou, and D. C. H. Yu, "System on integrated chips (SoIC(TM) for 3D heterogeneous integration," in *Proc. IEEE 69th Electron. Compon. Technol. Conf. (ECTC)*, May 2019, pp. 594–599.

[5] D. B. Ingerly et al., "Foveros: 3D integration and the use of face-to-face chip stacking for logic devices," in *IEDM Tech. Dig.*, Dec. 2019, p. 19.

[6] 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.

[7] E. Qin et al., "SIGMA: A sparse and irregular GEMM accelerator with flexible interconnects for DNN training," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2020, pp. 58–70.

[8] 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.

[9] 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.

[10] K. Shiba, T. Omori, M. Hamada, and T. Kuroda, "A 3D-stacked SRAM using inductive coupling technology for AI inference accelerator in 40-nm CMOS," in *Proc. 26th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2021, pp. 97–98.

[11] C.-Y. Lo, P.-T. Huang, and W. Hwang, "Energy-efficient accelerator design with 3D-SRAM and hierarchical interconnection architecture for compact sparse CNNs," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Aug. 2020, pp. 320–323.

[12] H. Esmaeilzadeh 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.

[13] E. LeDell and S. Poirier, "H2O AutoML: Scalable automatic machine learning," in *Proc. 7th ICML Workshop Automated Mach. Learn. (AutoML)*, Jul. 2020. [Online]. Available: https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf

[14] R. Mathur, A. K. A. Kumar, L. John, and J. P. Kulkarni, "Thermal-aware design space exploration of 3-D systolic ML accelerators," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 7, no. 1, pp. 70–78, Jun. 2021.

[15] H. Li, M. Bhargav, P. N. Whatmough, and H.-S. Philip 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.

[16] 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.

[17] G. Murali, A. Iyer, N. Ravichandran, and S. K. Lim, "3DNN-xplorer: A machine learning framework for design space exploration of heterogeneous 3D DNN accelerators," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Oct. 2023, pp. 1–9.

[18] Z. Zeng and S. S. Sapatnekar, "Energy-efficient hardware acceleration of shallow machine learning applications," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023, pp. 1–6.

[19] 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.

[20] H. Esmaeilzadeh et al., "VeriGOOD-ML: An open-source flow for automated ML hardware synthesis," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2021, pp. 1–7.

[21] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "SCALE-sim: Systolic CNN accelerator simulator," 2018, *arXiv:1811.02883*.

[22] H. Kwon, A. Samajdar, and T. Krishna. (2019). *Maeri Project*. Accessed: May 20, 2023. [Online]. Available: https://github.com/maeri-project/MAERI_bsv

[23] L. Bamberg, A. Garcia-Ortiz, L. Zhu, S. Pentapati, D. E. Shim, and S. K. Lim, "Macro-3D: A physical design methodology for face-to-face-stacked heterogeneous 3D ICs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 37–42.

[24] S. S. K. Pentapati and S. K. Lim, "Heterogeneous monolithic 3D ICs: EDA solutions, and power, performance, cost tradeoffs," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 925–930.

[25] A. Malistov and A. Trushin, "Gradient boosted trees with extrapolation," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2019, pp. 783–789.

[26] F. Pedregosa et al, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[27] R. Yousefzadeh and X. Cao, "To what extent should we trust AI models when they extrapolate?" 2022, *arXiv:2201.11260*.

[28] V. F. Pavlidis, I. Savidis, and E. G. Friedman, *Three-Dimensional Integrated Circuit Design*, 2nd ed., San Francisco, CA, USA: Morgan Kaufmann, 2017.