NPUWattch: ML-based Power, Area, and Timing Modeling for Neural Accelerators

Sehyeon Kim*, Minkwan Kim*, Chanho Park*,
Hanmok Park†, Seonghoon Kim†, Taigon Song†, and William J. Song*
*School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea
{ikamusume, mk9617, ch.park, wjhsong}@yonsei.ac.kr
†School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, South Korea
{phm2021, tjdgns4202, tsong}@knu.ac.kr

2026 IEEE International Symposium on High Performance Computer Architecture (HPCA) | 979-8-3315-9302-5/26/$31.00 ©2026 IEEE | DOI: 10.1109/HPCA68181.2026.11408488

***Abstract*—Pre-silicon modeling tools for characterizing power, area, and timing (PAT) have enabled numerous architectural studies, but traditional analytical and table-based models begin to exhibit limitations in their applicability as architectural design complexity increases and process technology scales below 5nm with the emergence of advanced transistors. Previous modeling techniques typically assume static scaling factors across different designs and technology nodes, derived from small circuit design benchmarks using old processes. Consequently, they do not reflect complex design variability and nonlinear projection to advanced technology nodes. Moreover, reference logic and SRAM implementations serving as the baseline for design and technology scaling are often created using different technologies and design rules, leading to significant estimation inaccuracies that distort the relative contributions of individual components. To address these challenges, this paper introduces *NPUWattch*, a machine learning-based PAT modeling framework for neural accelerators. It leverages neural network regression models to learn complex nonlinear relationships in technology and design scaling based on diverse post-layout logic and SRAM design datasets formulated using unified technology libraries. To this end, we developed technology libraries from 65nm to 2nm, constructed and validated diverse logic and SRAM datasets, and trained neural network models using an adaptive loss function to reinforce underrepresented regions of the design space. NPUWattch is validated against the post-layout results of numerous open-source neural accelerators, and evaluation results demonstrate that NPUWattch outperforms existing tools with an average estimation error of 2.7%, offering reliable and accurate PAT estimation.**

***Index Terms*—Neural accelerator, modeling, machine learning, design scaling, technology library**

I. Introduction

Pre-silicon modeling tools for power, area, and timing (PAT) estimation have significantly contributed to architectural research, enabling rapid design space exploration (DSE) in various domains [4], [22], [23], [33], [46]. However, continued technology scaling and growing architectural complexity have started to reveal fundamental limitations in traditional modeling approaches. Analytical and table-based models, typically calibrated to old technology nodes, struggle to capture nonlinear trends when applied to emerging nodes and unconventional designs, where assumptions from earlier generations and referenced designs no longer hold. While detailed back-end place-and-route (PnR) flows can provide accurate estimation, they are prohibitively time-consuming for use in early-stage DSE.

The widespread adoption of neural networks (NNs) across diverse applications has spurred the development of hardware accelerators. Accurate PAT characterization of these accelerators is crucial during pre-silicon analysis, and existing tools like Aladdin [33] and Accelergy [46] have demonstrated the value of fast, pre-RTL estimation using analytical equations, lookup tables, and microbenchmarking. However, traditional modeling methods have limitations due to their reliance on static technology and design scaling factors, often derived from inconsistent design conditions. As architectural design complexity increases and process technology scales into sub-20nm nodes using fin field-effect transistors (FinFETs) and gate-all-around FETs (GAAFETs), the assumptions underpinning these models break down, causing significant estimation errors when extrapolated beyond their calibration bounds.

Several challenges underscore the need for a more robust modeling framework. First, traditional approaches based on static scaling factors, typically derived from small reference designs at legacy nodes, fail to capture nonlinear technology scaling trends when projected onto advanced nodes with distinct properties. Second, analytical and table-based modeling methods to extrapolate design costs from reference implementations do not correctly reflect the variability across diverse design configurations, leading to inaccuracies in estimating larger, more intricate structures. Lastly, the coherence between logic and memory modeling has been overlooked, as existing tools often apply inconsistent assumptions and technology libraries to them (e.g., combining Aladdin [33] and CACTI [4] models), producing misleading aggregate results.

To address the challenges, this paper introduces *NPUWattch*, a machine learning (ML)-based modeling framework for rapid and accurate estimation of PAT characteristics in neural accelerators. NPUWattch leverages NN regression models that directly predict the power, area, and timing values of accelerator designs by learning complex nonlinear relationships between various design parameters and the PAT metrics across a broad range of technology nodes and diverse design configurations. For this purpose, we developed technology libraries spanning from 65nm to 2nm nodes, including FinFET and GAAFET devices, and implemented a curated suite of representative logic and SRAM components for modeling neural accelerators, using commercial electronic design automation (EDA)

tools to ensure design accuracy and process fidelity. At each process node, both the logic and SRAM components were designed using a single, unified transistor model, adhering to consistent design rules. The register transfer level (RTL)-level implementations of logic and SRAM components were parameterized and swept across a wide range of architectural configurations and technology nodes to extract post-layout power, area, and timing data, which are used to construct design datasets for training NN prediction models. To account for the imbalanced distribution of design points in the datasets, the NN models were trained using an adaptive loss function to maintain robustness in underrepresented regions of the design space. NPUWattch is validated against the post-layout results of numerous open-source neural accelerators [26], [38]–[40], [42], and evaluation results demonstrate that it outperforms existing tools, offering reliable and accurate PAT estimation. In summary, this paper makes the following key contributions:

1) We build technology libraries spanning *advanced process nodes down to 2nm* and curate a comprehensive post-layout dataset that *reflects actual circuit characteristics*.
2) Unlike many NN-based studies employing neural models mainly to guide search or optimization, NPUWattch *directly estimates nonlinear PAT values* using a high-accuracy regression model, enabling a drop-in replacement for existing rule-based models in architectural frameworks.
3) To enhance model robustness and accuracy, we introduce *new input features* (e.g., cell count ratio) crucial for cross-technology scaling and propose *an adaptive loss function* that mitigates dataset imbalance during training.

II. Background and Related Work

A. Modeling Neural Accelerators

Neural accelerators are primarily composed of *digital logic* and *SRAM* structures. Digital logic forms the computational core of accelerators, executing operations and managing control flow within datapaths. The logic components include processing elements (PEs), each typically comprised of multiply-accumulate (MAC) units and registers. Additionally, routing logic and interconnections also play a crucial role in steering data movements. The large number of PEs occupy a significant design area and mainly contribute to the power consumption of accelerators induced by dynamic computational activities.

SRAM structures serve primarily as on-chip buffers to store weights, activations, and intermediate data. They aim to supply data efficiently to PEs and maximize local data reuse, reducing costly off-chip DRAM access. Due to the significant memory requirements of neural operations, large on-chip SRAM arrays consume a considerable share of the die area.

B. Modeling Digital Logic

Accurately modeling digital logic entails a full digital design flow, involving multiple computer-aided design (CAD) tools and substantial compute resources. While high-level synthesis facilitates the abstraction of design details, it still requires resource-intensive RTL implementations. Moreover, such a circuit-level analysis significantly restricts the explorable range of design space because even a minor modification to the design incurs costly re-simulation effort. Therefore, a more agile approach is generally employed for the efficient exploration of various design alternatives.

TABLE I: Power, Area, and Timing Modeling Frameworks

Framework	Target	Modeling Method
CACTI [4]	SRAM	Analytical
McPAT [23]	CPU	Analytical
GPUWattch [22]	GPU	Analytical
AccelWattch [20]	GPU	Analytical
Aladdin [33]	Accelerator	Table-based
Accelergy [46]	Accelerator	Table-based & analytical
NPUWattch (this work)	Accelerator	ML-based regression

A common strategy for reducing the complexity of modeling large logic structures begins with defining a set of primitive components, ranging from basic elements (e.g., registers, crossbars) to macros (e.g., arithmetic units). To calculate the total design area, the area contributions of these components are summed while accounting for integration overhead.

Power estimation typically uses the same set of primitives but considers the specific operational behaviors of individual components (e.g., read/write counts). To address this, cycle-level simulators are engaged to generate execution traces that specify the activity counters of architectural elements. Then, the power activities of all constituent components are aggregated to calculate the total power consumption. This modular approach has been widely adopted in the related literature as a practical compromise between accuracy and efficiency. Table I summarizes widely used modeling frameworks, and the following subsections describe their modeling methods.

1) Analytical Models: This approach uses design parameters and circuit-level model equations for the PAT estimation of logic units. It decomposes an architecture into basic structures and calculates the area and power of each block based on its circuit-level characteristics and switching activities, which are then aggregated to compute the total design costs.

McPAT [23] is a CPU modeling framework that employs an analytical method to predict the design costs of standard cell-based regular structures and utilizes curve-fitting models for custom, area-optimized units such as arithmetic units. To handle technology scaling, it defines parameter data for several anchor nodes (i.e., 22-90nm) and references them to interpolate the power and area estimates for a queried node.

Similarly, GPUWattch [22] uses an analytical method to estimate GPU power and incorporates an execution unit model derived from a Verilog design. It utilizes the area and power data calibrated at the 45nm node as the baseline and scales prediction results to the target node by International Technology Roadmap for Semiconductors (ITRS) projections. A follow-up study, AccelWatch [20], offers updated power models tailored for more recent GPU architectures and technology nodes.

2) Table-based Models: This approach utilizes predefined lookup tables to deduce the PAT estimates of a design. For example, Aladdin [33] defines several primitive components

(e.g., counters, adders) and constructs their pre-RTL timing and power models. Each component is synthesized under various clock constraints at the 40nm node, and the resulting area and power estimates are stored in a table. To evaluate the design cost of a logic design, Aladdin consults the pre-characterized table of primitive components and multiplies integer scaling coefficients. However, the design parameters of these primitive components are largely fixed, limiting the modeling flexibility and accuracy.

Accelergy [46] extends the table-based approach with linear interpolation. It integrates Aladdin's pre-characterized table of primitive components at the 40nm node as a plug-in model and further combines them to express compound structures. To estimate the area of a logic design, Accelergy references the plug-in lookup table and interpolates the data according to the given architectural configurations. For power estimation, it utilizes predetermined scaling factors to model additional behaviors in neural accelerators, such as `reused` to indicate the operand reuse of MAC operations and `gated` to represent a disabled clock. Notably, its integration with Timeloop [27] has facilitated the evaluation of accelerator energy consumption across different dataflows and hardware configurations. For technology scaling, Accelergy uses a helper function based on the study of technology scaling factors [36].

C. Modeling SRAM Structures

While previous frameworks used either analytical or table-based models for digital logic [22], [23], [33], [46], they commonly relied on CACTI [4], [25] for SRAM analysis. This modeling process begins at the memory cell level, typically using six-transistor configurations with cross-coupled inverters for storing bits and access transistors for read/write operations. These SRAM cells are organized in arrays with shared wordlines and bitlines. Peripheral circuits are also essential for SRAM, including row decoders, wordline drivers, bitline precharge circuits, and sense amplifiers for data retrieval.

After defining SRAM cells and peripheral circuits, the modeling process advances to the subarray level. Instead of forming one large monolithic array, the memory is partitioned into subarrays to reduce wire length and delay. Each subarray includes decoders, sense amplifiers, and output drivers, all of which are scaled to subarray dimensions. By specifying the number of rows, columns, and subarrays, the model can address performance targets and layout constraints.

As a large SRAM block comprises multiple subarrays, its design cost is calculated by aggregating respective timing, area, and power overheads. CACTI has been widely used for this, leveraging analytical formulas and technology-specific parameters to estimate access time, read/write energy, and leakage power. It provides a comprehensive view of expected SRAM performance for the given architectural configurations.

D. Other Models

Other prior studies have proposed design cost estimation models with different objectives from this paper. For instance, ORION [19] targets pre-synthesis estimation of area and power for network-on-chip (NoC) designs. It uses regression models to relate architectural parameters (e.g., port count, buffer size) to cost metrics obtained from post-layout evaluations.

More recently, NN-based approaches have gained traction in hardware implementation. Several studies utilized NN models to rank design candidates, effectively pruning the search space and guiding optimization solvers [9], [47]. At the RTL and gate level, other works have demonstrated the acceleration of power sign-off by inferring switching activity from netlists, achieving high accuracy and significantly faster runtime than commercial CAD tools [11], [48], [49].

In contrast to these efforts, this paper presents a modeling methodology focused on accurate power, area, and timing estimation at a higher level of architectural abstraction over a wide range of technology scaling and design variations. The proposed NPUWattch framework enables comprehensive architectural design exploration for neural accelerators.

III. Motivation

Previous analytical and table-based models made significant contributions to architectural design studies, offering reasonable modeling accuracy and efficiency. However, as design complexity grows beyond their calibration bounds and technology scaling advances into sub-20nm nodes with new FinFET and GAAFET devices, these modeling approaches begin to expose critical limitations in accurately estimating the PAT characteristics of diverse logic and SRAM designs under modern process technologies. This section examines the challenges in detail and presents the motivation for the development of an ML-based modeling framework.

A. Technology Scaling Factors: One Size Never Fits All

For power and area estimation, *technology scaling factors* are frequently used to project post-layout design costs onto newer technologies, enhancing the portability of models across process nodes. While such technology projection is often represented by a "single scalar factor," our evaluation reveals that technology scaling factors actually have a wide distribution and cannot be expressed as a static parameter.

As shown in recent studies [7], [12], [14], [37], technology scaling factors are commonly used to normalize the design costs of various implementations at different process nodes for comparative analyses, based on the methodology proposed by Stillmaker et al. [36]. Since these technology scaling factors are typically derived from a small set of transistor-level CMOS benchmarks [31], [36], they should be applied only to similar CMOS designs. Moreover, due to the simplicity of scaling factor-based projections, many design-dependent parameters are often disregarded in related studies, such as wire loads whose variability across technology nodes cannot be easily determined. These inherent limitations restrict the applicability of static scaling approaches only to small circuits. Our review of prior works above reveals that scaling factors were used to extrapolate design implementations from academically accessible mature technology nodes to advanced sub-20nm

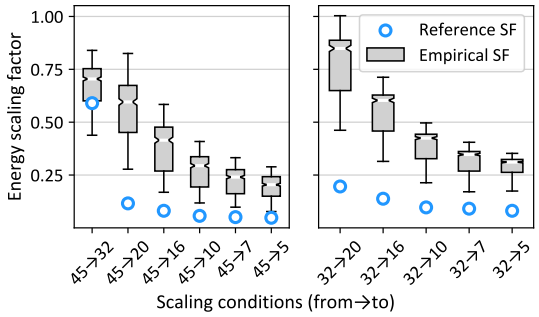


Fig. 1: Comparison between reference scaling factors (blue rings) and the distribution of empirical scaling factors (box and whisker) derived from post-layout energy measurements when scaling the benchmark designs from traditional 45 or 32nm to advanced technology nodes.

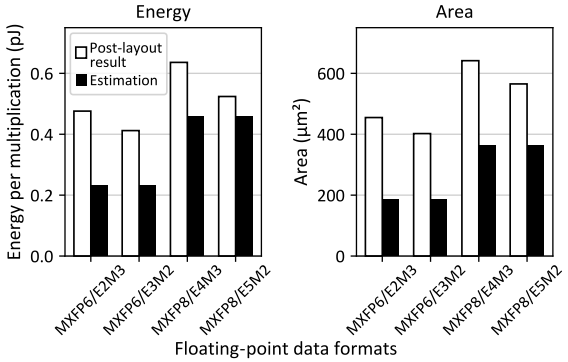


Fig. 2: The energy and area comparison of floating-point multipliers with varying operand widths between Accelergy estimates (with the Aladdin plug-in) and post-layout results.

technologies employing FinFETs or GAAFETs, despite the physical and electrical incompatibilities.

To assess the efficacy of scaling factor-based approaches in larger designs, we developed transistor models for 20nm, 16nm, 10nm, 7nm, and 5nm FinFET technologies[1] and compiled corresponding technology libraries compatible with commercial EDA tools. Also, following the methodology of Stillmaker et al. [36], we re-established *reference scaling factors* for our technology libraries. Using these libraries, we extracted post-layout design costs from a suite of benchmark designs[2] and computed the corresponding *empirical scaling factors*.

Fig. 1 compares the reference scaling factors with the distribution of empirical values across the selected nodes. Notably, for energy scaling, empirical factors show substantial variance, but the reference scaling factors fail to capture this variability, consistently underestimating energy costs due to disregarded design-dependent parameters. The evaluation results indicate that previous static scaling approaches should be used only for small designs with limited transistor counts. Therefore, accurate estimation for larger, more complex designs requires a fundamentally different modeling methodology.

B. Nonlinearity Pitfall in Design Scaling

Design scalability must be considered not only in terms of technology nodes but also with respect to architectural configurations, where *nonlinearity* poses a critical challenge. To illustrate this issue, we implemented floating-point multipliers[3] with varying operand widths, extracted their post-layout design costs, and compared the results against extrapolated estimates generated by Aladdin [33] and its plug-in model in Accelergy [46]. Fig. 2 plots a comparison between these analytical estimates and post-layout results for various microscaling (MX) floating-point formats [30] using 6- or 8-bit representations (i.e., MXFP6 or MXFP8) with randomly generated operand bit patterns. Since Accelergy derives energy and area estimates by scaling the data from a standard FP32 implementation, it exhibits increasing estimation errors at lower bitwidths due to nonlinear design scaling effects. Furthermore, it does not account for variations in data formats (i.e., the number of exponent and mantissa bits) for the same bitwidth (e.g., E2M3 vs. M3E2), which is crucial for modeling low-precision operations in modern neural accelerators.

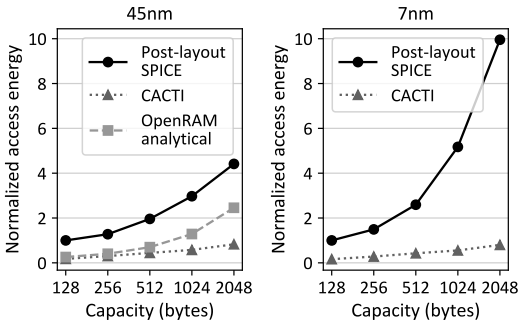


Fig. 3: Normalized random access energy of SRAM with 32-bit line size for increasing capacities, based on CACTI and post-layout SPICE simulations at 45nm and 7nm nodes.

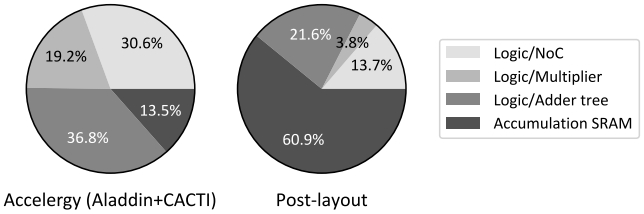
Compared to digital logic modeling, SRAM presents unique challenges due to its dependence on analog circuitry with distinct scaling behavior, making it difficult to model accurately via traditional analytical methods. As a result, architectural studies generally rely on specialized tools such as CACTI [4], [25] for estimation. To evaluate the accuracy of the CACTI model, we implemented SRAM arrays[4] at two distant technology nodes (i.e., 45nm and 7nm) and obtained access energy estimates using CACTI 7.0 [4] through the Accelergy-CACTI plug-in. Fig. 3 reveals that the built-in analytical model of OpenRAM better tracks the overall trend of measured

[1]The transistor models are based on predictive technology model (PTM) [24] and BSIM-CMG [6]. The detailed development process is described in Section IV-C.

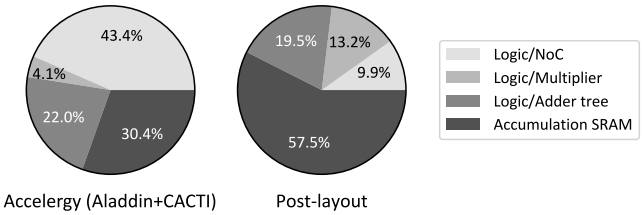
[2]The test suite includes academically available benchmark circuits and custom-developed components for neural accelerators, such as arithmetic units, FIFOs, crossbars, etc.

[3]The floating-point multiplier designs were implemented using FreePDK45 [35] to similarly match the 40nm design baseline in Aladdin [33], to which Accelergy [46] is calibrated. Then, the extracted design costs were integrated into Accelergy's lookup tables to ensure a fair comparison.

[4]We generated SRAM macros at the 45nm node, using FreePDK45 [35] and the OpenRAM compiler [15], and designed comparable arrays for the PTM-based 7nm model. To improve accuracy, we extracted post-layout parasitics and performed transient SPICE simulations to obtain precise energy and timing measurements.



(a) Energy breakdown



(b) Area breakdown

Fig. 4: (a) Energy and (b) area breakdown of SIGMA's Flex-DPE-128 module with 128KB SRAM [29]. Estimates from Accelergy [46] and CACTI [4] are compared against 45nm post-layout measurements using FreePDK45 [35] and OpenRAM [15].

energy, whereas CACTI shows a different scaling trajectory as SRAM capacity increases. In all cases, both analytical models in OpenRAM and CACTI consistently underestimate energy and area, suggesting the need for more advanced modeling techniques that can account for the nonlinear scaling behavior of SRAM design costs.

C. Pitfall of Heterogeneous Modeling

If a monolithic system, comprising both digital logic and SRAM components, is evaluated using multiple heterogeneous models, it must ensure that the models are characterized under the same process conditions. Otherwise, the resulting compound estimates can be significantly distorted and misleading. For instance, Aladdin [33] addresses this issue by employing a memory power model derived from a 40nm memory compiler that uses the same commercial cell library as its digital logic models, preserving consistency across components. On the other hand, Accelergy [46] uses a combination of heterogeneous plug-in models, including Aladdin for modeling digital logic components and CACTI for SRAM structures, to characterize neural accelerator architectures.

To examine the impact of such a heterogeneous modeling approach with multiple plug-in models, we used Accelergy and CACTI to model SIGMA's Flex-DPE-128 module with a 128KB SRAM buffer [29]. The resulting energy and area estimates are compared to 45nm post-layout results obtained using FreePDK45 [35] and OpenRAM [15]. In Fig. 4, the darkest segments are SRAM portions. Compared to the post-layout data, the heterogeneous models significantly underestimate SRAM energy by 42% and area by 33%. These inaccuracies not only misrepresent SRAM costs but also skew the relative contributions of other components, distorting the overall system-level breakdown. This highlights the significance of maintaining consistency between logic and SRAM models.

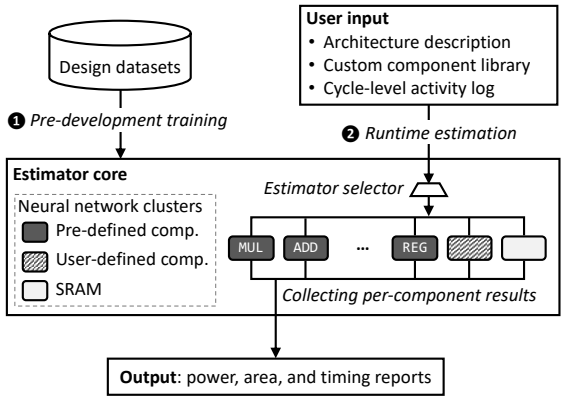


Fig. 5: Overall structure of the NPUWattch framework, comprising component-specific NN clusters to estimate PAT characteristics.

D. Summary

While existing methods for estimating PAT characteristics of logic and SRAM designs have significantly contributed to systems research, they exhibit inherent limitations for continued technology scaling and growing architectural complexity. First, static technology scaling factors, typically derived from small circuit benchmarks at old process nodes, fail to capture design variability when projected onto advanced nodes. Second, analytical and table-based models to extrapolate design costs from small reference structures do not accurately reflect nonlinearity in design scaling. Lastly, plug-and-play methods that combine heterogeneous models calibrated under different process conditions significantly distort the relative contributions of components, misrepresenting overall design costs.

To address the limitations, this paper proposes *NPUWattch*, an ML-based modeling framework that directly predicts the power, area, and timing of neural accelerators. By leveraging NN-based models, NPUWattch learns complex nonlinear behaviors in technology and design scaling based on logic and SRAM design datasets compiled using unified libraries over a wide range of technology nodes and design variations, enabling reliable PAT characterization.

IV. Methodology

A. NPUWattch Overview

This section describes the organization and design procedure of the NPUWattch framework, including i) the frontend interface for users, ii) the development of technology libraries, iii) the dataset construction process, and iv) the design and training of NN models for accurate PAT characterization[5].

Fig. 5 shows the input/output of the NPUWattch framework and outlines its two core stages: pre-development training and runtime estimation. In the pre-development step, NN clusters are trained on a curated dataset of post-layout measurements, covering ten technology nodes (i.e., 65nm to 2nm) and diverse design configurations. At runtime, users provide a high-level

[5]The NPUWattch framework and accompanying datasets are available at: https://github.com/yonseicasl/NPUWattch

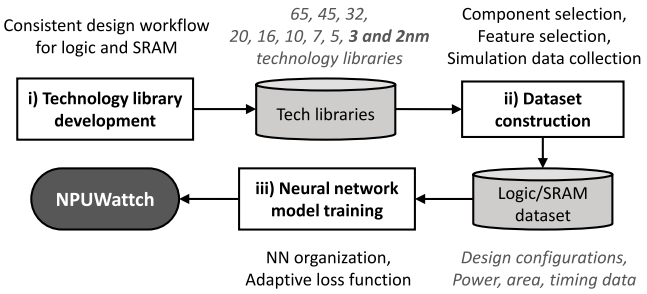


Fig. 6: Overall design procedure of NPUWattch development.

architectural description, along with optional inputs such as per-component activity counters or custom design cost libraries. NPUWattch then aggregates power, area, and timing reports. The following outlines the framework organization.

1) Input: The frontend of the framework takes an architecture description as input, which specifies modeled blocks and their design parameters (e.g., line size, capacity) in the target design. Optionally, it can take i) a custom component library containing design cost data for user-defined modules and ii) a cycle-level activity log for workload-aware energy estimation.

2) Estimator Core: NPUWattch includes separate estimators for logic and SRAM components. The logic estimator comprises two sub-engines: pre-defined and user-defined component estimators. The pre-defined component estimator builds NN clusters for common design blocks, such as arithmetic units and registers. To account for the nonlinearity of design scaling and variability across component types, a separate NN model is created for each component-metric pair (i.e., power, area, timing) to ensure estimation accuracy. The user-defined component estimator allows users to provide the data of custom designs (e.g., bespoke control logic) at specified technology nodes, similar to plug-in models of Accelergy. In the SRAM estimator, a dedicated NN cluster handles SRAM macros, supporting a wide range of memory configurations.

3) Output: After generating per-component PAT estimates, NPUWattch aggregates them to produce system-level power, area, and timing reports. Total energy is calculated based on the activity counts of individual components provided by the user. If activity counters are not given, NPUWattch defaults to an average random switching activity (e.g., $\sim$10% [1], [18], [41]), offering a first-order estimate.

B. Overall Design Procedure

The design methodology for NPUWattch consists of three key steps: i) technology library development, ii) dataset construction, and iii) NN model training, as outlined in Fig. 6. In the first step, technology libraries are created [21], including traditional CMOS (>20nm) and emerging device types such as FinFET ($\leq$20nm) and GAAFET (<5nm). Each library includes transistor-level SPICE models, standard cell GDSII layouts, and a memory compiler. For unified logic and SRAM modeling, all circuit elements are designed using a single transistor model, adhering to consistent design rules. Commercial EDA tools are used throughout this process to ensure design accuracy and process fidelity, as listed in Table II.

TABLE II: EDA Tools (Synopsys) for Post-layout Implementations

Design flow	Tools	Design flow	Tools
Synthesis	Design Compiler	Place-and-route	IC Compiler II
Sign-off	PrimeTime	Layout	Custom Compiler
LVS	IC Validator	RC extraction	StarRC, Raphael
Circuit sim.	HSPICE, PrimeSim	TR modeling	TCAD

In the second step, design datasets are built to train the NN models. Configurable RTL implementations of accelerator components are created, each designed to span a broad range of architectural parameters. Meanwhile, a memory compiler generates diverse SRAM structures. Sweeping these design parameters across technology nodes yields detailed post-layout power, area, and timing data, forming comprehensive datasets that capture the effects of both design and technology scaling.

In the final step, a neural network architecture and its hyperparameters are determined based on the datasets constructed in the previous stage. Separate NN models are trained for each component type and PAT metric, as they have distinct scaling behaviors with respect to design and technology parameters. For user-defined logic components, prediction models must cover broad inputs while being robust even in underrepresented regions of the design space. For robustness and training efficiency, NPUWattch employs an adaptive loss function to reflect the statistical properties of training data as identified through preliminary dataset analysis.

C. Technology Library Development

Recent advances in device technology have moved beyond FinFETs, embracing more advanced transistor types such as GAAFET. Accurate design cost estimation across both mature and emerging technology nodes is critical to ensure modeling consistency and quantify cost differences between process nodes. However, acquiring commercial process design kits (PDKs) for advanced nodes is prohibitively difficult in academic environments, and access to their detailed physical layouts is even more restricted. To overcome this problem, we developed technology libraries down to the 2nm node to enable NPUWattch to estimate PAT metrics across a broad range of emerging nodes [34]. For sub-20nm processes, several open-source technology libraries are available, including FreePDK15 [5], ASAP7 [44], and ASAP5 [43].

Fig. 7 outlines the workflow for developing technology libraries. Guided by semiconductor trends and related studies on technology library design [34], [44], we selected FinFETs for the 20nm, 16nm, 10nm, 7nm, and 5nm nodes, and GAAFETs for the 3nm and 2nm nodes (specifically, nanosheet FET for 3nm and forksheet FET for 2nm). To design a library that supports both logic circuits and SRAM, we followed the library development process based on Synopsys tools.

The device characteristics of the latest transistors have already been well established through experiments on real silicon [3], [21], [45]. Using commercial EDA tools, we con-

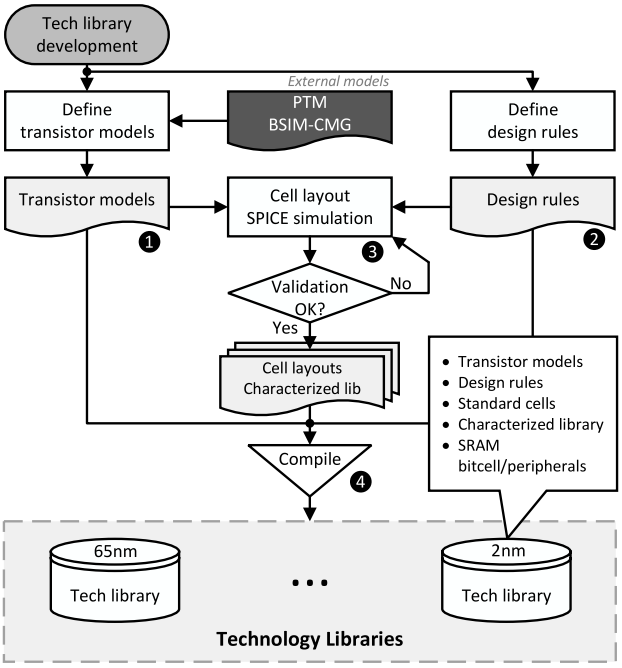


Fig. 7: Technology library development workflow.

structed technology libraries following an industry-standard design flow. ❶ The process begins by creating transistor models that characterize each device type. We designed and modeled actual devices using Sentaurus Technology Computer-Aided Design (TCAD) and simulated their performance at the transistor level. For this purpose, we used PTM [24] and BSIM-CMG [6] to develop accurate SPICE models for FinFETs and GAAFETs. ❷ We defined strict design rules, including minimum feature sizes, metal pitch, and layer dimensions, to guide physical layout. ❸ Then, we constructed layouts for standard cells[6], SRAM bitcells, and peripheral circuits using Custom Compiler, followed by parasitic extraction via StarRC (or Raphael for nodes below 5nm), as listed in Table II. The functionality of each cell was verified through simulations based on post-layout RC-extracted netlists using HSPICE. Regarding validation, we confirmed that all designs were DRC/LVS-clean via IC Validator[7]. ❹ Once all cells are verified, we extracted LEF files from the layouts using Custom Compiler and generated library files using PrimeLib. These files were used to create NDM files that enable full chip-level implementation in IC Compiler II. At this stage, transistor and process variability were considered by adjusting parameters in the PTM- and BSIM-CMG-derived SPICE models[8]. Each library was characterized across multiple process, voltage, and temperature (PVT) corners to cover a wide range of operating conditions[9]. We also built technology-specific memory compilers for end-to-end SRAM generation. A Python flow emits GDSII layouts for the main array and corresponding physical libraries according to the design rules, while periphery circuits are designed and validated in the same flow.

D. Dataset Construction

The central principle of our workflow is to give deliberate attention to often-overlooked steps while maintaining a holistic view of components with diverse characteristics. This principle guides the entire dataset construction process. In contrast to prior works such as Aladdin [33], which relied solely on post-synthesis PAT reports to collect design costs, we recognize that synthesis tools offer only preliminary estimates. Thus, these approaches lack physical implementation details, such as interconnect routing overhead and spatial distribution of clock networks, which become available after the PnR step.

To address these limitations, we performed a full design flow for each design using the toolchain in Table II. After completing PnR, we extracted parasitic RC information for power simulations, which is an increasingly important factor as technology nodes scale down. The design area is obtained directly from the layout, encompassing all associated design overheads. While the most accurate power simulations ideally require activity waveforms from actual workloads, incorporating such steps significantly increases the complexity and effort involved in dataset construction. To balance between accuracy and scalability, we measured leakage and total dynamic power by applying a uniform average toggle rate, assuming random switching activity across all nodes, in line with standard industry practice [1], [18], [41].

To ensure consistency across components with diverse electrical characteristics, all datasets for a given node were generated using a unified transistor model. This minimizes variability within design datasets and streamlines the system-level modeling of heterogeneous components. The datasets are organized into two subcategories: a logic dataset, which contains designs based on combinational gates and flip-flops, and an SRAM dataset, which includes mixed-signal circuitry.

1) Logic Dataset: The logic dataset has two subsets: a predefined component dataset and a general logic scalar dataset. The predefined component dataset includes parameterized primitive designs[10] curated for modeling neural accelerators, with design cost metrics collected via parameter sweeps. The general logic scalar dataset covers a broader range of designs, necessary for estimating scalar projections to any technology nodes with minimal user input. This requires diverse design

[6]We considered 48 standard cells with various device strengths based on related studies [21].

[7]We also developed the rules for layout versus schematic (LVS) and design rule checking (DRC) processes based on the syntax of IC Validation.

[8]We modeled transistors using Sentaurus TCAD accounting for process variations (e.g., doping concentration, thermal diffusion), fitted the I-V characteristics to reference data, and generated SPICE model cards.

[9]Each technology node includes two transistor models: high-performance and low-power. Each model was characterized across three process corners (e.g., TT, SS, FF), seven voltage conditions (nominal Vdd ± 0.05/0.10/0.15V steps), and three temperature conditions (-40/25/85°C), covering all accelerator designs evaluated in our experiments.

[10]Building on prior studies [33], [46], we selected fourteen primitive designs, including integer and floating-point adders, multipliers, MAC units, register files, FIFOs, crossbars, wires, and others. For each component, 20-150 models were generated by sweeping architectural parameters (e.g., bitwidth). Each component operates at 200-1200MHz with 200MHz steps, and each model incorporates data from 30+ characterized libraries per node.

cases for effective NN model training. To support this, we gathered design cost data from various benchmark circuits and open-source IP cores. Each design entry in the logic dataset contains design specification, synthesized technology node, combinational and sequential cell counts and areas, total area, dynamic energy, leakage power, critical path delay, and operating voltage.

2) SRAM Dataset: Using the SRAM compiler described in Section IV-C, we generated a diverse set of SRAM instances[11]. To maintain methodological consistency with the logic dataset, we performed LVS checks and parasitic extraction (PEX) using the same toolchain. This ensures that the resulting measurements are grounded in accurate post-layout circuit representations. Mixed-signal simulations were conducted in PrimeSim to measure read and write energy values, as well as leakage power. Each SRAM instance in the dataset includes bitwidth, word count, bank count, total area, average read and write energy, leakage power, and operating voltage.

E. Neural Network Model and Training

1) NN Architecture and Organization: NPUWattch utilizes lightweight multilayer perceptron (MLP) models to balance modeling accuracy and efficiency. Each MLP consists of an input, an output, and 2-6 hidden layers, chosen based on the dimensionality and nonlinearity of training data. As illustrated in Fig. 5, the PAT estimators are organized into independent NN clusters, each dedicated to a specific component class. NPUWattch includes a total of sixteen such estimator clusters[12]. Given an architectural description input, it queries the relevant cluster to compute PAT metrics.

2) Input Feature Selection: When scaling designs across technology nodes, defining a single universal scaling factor is infeasible, as shown in Fig. 1. Resolving this problem requires identifying representative parameters that capture key design traits using only limited early-stage information. To this end, we selected the counts and footprint sizes of sequential and combinational standard cells as additional features. This choice is motivated by two observations: i) combinational logic gates and sequential elements (e.g., D flip-flops) exhibit distinct scaling behaviors as technology nodes shrink [2], and ii) the clock network associated with sequential logic significantly affects chip-level power consumption [8], [32]. These four metrics (i.e., sequential and combinational cell counts and areas) can be extracted as early as the logic synthesis stage (i.e., the first step in the implementation flow).

Based on these features, we computed the sequential cell count ratio (SCR) and sequential cell area ratio (SAR), which serve as informative inputs for training the NN models and improve generalization across diverse designs. For primitive designs, their SCR and SAR are already incorporated into the dataset and do not need to be specified explicitly. The user-defined component estimator offers SCR and SAR presets for common design classes (e.g., registers, compute units), facilitating extension to new designs. Optionally, users may supply PAT values explicitly along with SCR and SAR. If an RTL implementation is available, the cell counts reported in the synthesis log can be used directly.

[11]The SRAM dataset includes 25 configurations per node (e.g., bitwidths).

[12]Fourteen clusters are for predefined components, and the remaining two are for user-defined components and SRAM.

TABLE III: Adaptive Loss Function Terms

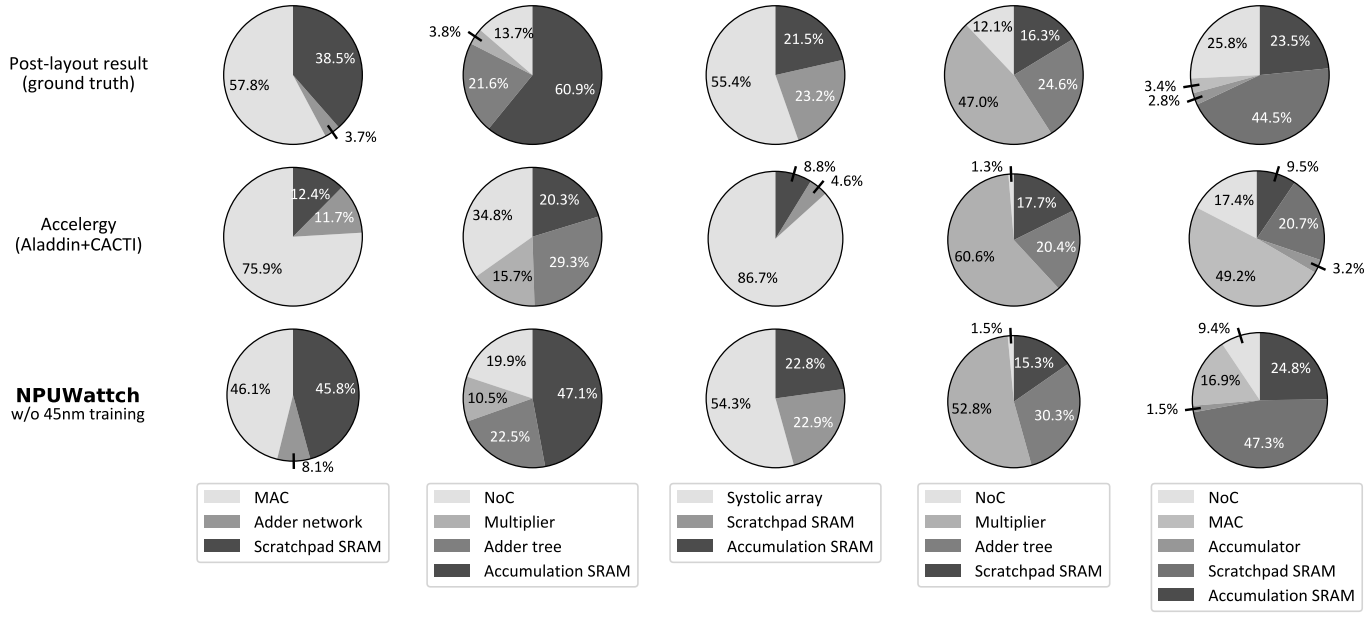
Term	Description
B	Mini-batch size used during training (from `DataLoader`)
$\hat{y}_i$	Predicted output of the model for sample i
y_i	Ground-truth label or target value for sample i
μ	Gain factor applied to combined histogram-based weights
w_i	Final adaptive weight for sample i, computed as $\mu\sqrt{\tilde{\omega}_c\tilde{\omega}_a}$
SCR_i	Sequential cell count ratio for sample i
SAR_i	Sequential cell area ratio for sample i
$b_c(\cdot)$, $b_a(\cdot)$	Bin-index functions mapping SCR/SAR to histogram bins
$\tilde{\omega}_c(k)$	Normalized inverse-frequency weight for count-ratio bin k
$\tilde{\omega}_a(k)$	Normalized inverse-frequency weight for area-ratio bin k
$\mathrm{pmf}(\cdot)$	Probability mass function
$\langle\cdot\rangle$	Arithmetic mean across histogram bins (for normalization)
ε	Small constant (i.e., 10^{-6}) to prevent division by zero
$\mathcal{L}_{\mathrm{wL1}}$	Weighted L1 loss: $\frac{1}{B}\sum_i w_i\lvert\hat{y}_i - y_i\rvert$

During dataset construction that reflects the physical characteristics of each design, we observed a long-tailed distribution. This imbalance arises because power and area samples for commonly used designs dominate the lower range, whereas very large designs are rare due to the layout overhead required to generate them. As the mean absolute error (MAE) uniformly aggregates loss across samples, the model naturally focuses on dense regions of the distribution. To mitigate this bias and promote balanced learning, we employ a custom loss function: *gain-scaled, histogram-weighted ℓ_1 loss*. Before training, the dataset is scanned to compute sample counts per interval (histogram bin), and per-interval errors are weighted inversely to their frequencies. These weights are applied to each sample during training and factored into the total loss. This approach adapts automatically to extensible datasets without requiring an explicit parametric fit to the underlying distribution. The weights are computed using the SCR and SAR metrics introduced in Section IV-E2, which are unevenly distributed across the dataset and affect PAT scaling behavior.

The custom loss function can be expressed as follows. $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_B)$ denotes the network output for a mini-batch of size B, and $\mathbf{y} = (y_1, \ldots, y_B)$ is the ground-truth targets. Sample weights are given by $\mathbf{w} = (w_1, \ldots, w_B)$. Then, the adaptive loss function is defined as Eq. (1), with terms detailed in Table III.

$$\mathcal{L}_{\mathrm{wL1}}(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{w}) \;=\; \frac{1}{B}\sum_{i=1}^{B} w_i \left|\hat{y}_i - y_i\right| \tag{1}$$

In the equation, the weight assigned to each sample i is computed as Eq. (2), where μ is a global gain factor applied



(a) FEATHER (b) Flex-DPE (c) Gemmini (d) MAERI (e) NVDLA

Fig. 8: Energy breakdown of neural accelerators at 45nm, from the post-layout simulation, Accelergy, and NPUWattch.

before storing the weights.

$$w_i = \mu \sqrt{\tilde{\omega}_c\big(b_c(\mathrm{SCR}_i)\big)\,\tilde{\omega}_a\big(b_a(\mathrm{SAR}_i)\big)} \tag{2}$$

The terms $\tilde{\omega}_c(\cdot)$ and $\tilde{\omega}_a(\cdot)$ are inverse-frequency histogram weights derived from SCR and SAR, respectively. These weights are normalized to unit mean, as defined in Eq. (3).

$$\tilde{\omega}_{(\cdot)}(k) = \Big(\frac{1}{\mathrm{pmf}_{(\cdot)}(k)+\varepsilon}\Big)\Big/\Big(\Big\langle \frac{1}{\mathrm{pmf}_{(\cdot)}+\varepsilon} \Big\rangle\Big) \tag{3}$$

V. Evaluation

A. Experiment Setup

1) Post-layout Simulation Setup: We performed comparative analysis at the 45nm node with Accelergy [46] and Aladdin [33], as they were primarily calibrated at 40nm. To establish reliable baselines, we conducted post-layout simulations for both logic and memory using the EDA tools in Table II. Nangate Cell Library was used for 45nm logic, and all 45nm memory instances were generated with OpenRAM [15] using full SPICE models (not simplified). For sub-20nm nodes, our technology libraries were utilized. All power analyses were performed on back-annotated post-layout netlists. Following common practice for vectorless power simulation, which assumes 10-20% switching activity [1], [18], [41], we adopted a default value of 10%.

2) Accelerator Designs: We selected neural accelerators with open-source RTL implementations [26], [38]–[40], [42] listed in Table IV to demonstrate the accuracy of NPUWattch in estimating hardware costs. Operating frequencies were set to 80% of the maximum values reported by timing analysis to represent feasible operating conditions. The energy results obtained from EDA tools at these frequencies serve as ground truth PAT metrics. Identical frequencies were supplied to both NPUWattch and Accelergy [46] for fair comparison. While the RTL implementations included all module blocks comprising the accelerators, we excluded functionally unique, design-specific components such as internal flow processors and communication interfaces from the evaluation, as these are not relevant during the DSE stage.

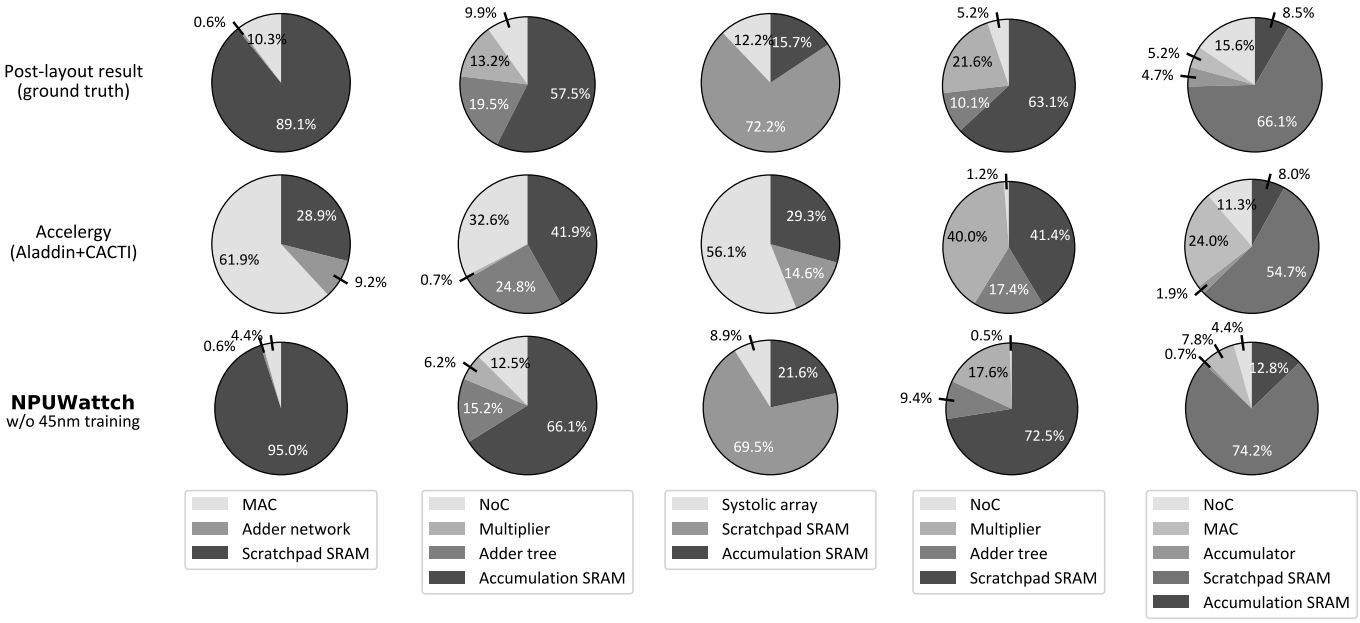
TABLE IV: Neural Accelerator Specifications

Accelerator	Data format	Local buffer	PEs	Network topology	Global buffer
FEATHER [38]	int8	9B	16×16	Spatial/adder tree	256KB
Flex-DPE [40]	bfloat16	2B	1×128	Adder tree	128KB
Gemmini [42]	int8	4B	16×16	Systolic	320KB
MAERI [39]	int16	6B	1×128	Adder tree	80KB
NVDLA [26]	int8	2B	8×8	Adder tree	65KB

3) Modeling Frameworks: We compared NPUWattch to Accelergy [46] equipped with the CACTI plug-in. To ensure fairness, the default Aladdin 40nm design cost data in Accelergy were replaced with our 45nm data tables, and the entire 45nm dataset was excluded from NPUWattch training. For all other technology nodes, NPUWattch utilized the comprehensive training datasets constructed in Section IV-D. All model training was performed on a system with a Ryzen 7700X CPU and 32GB of memory, without a GPU. Owing to the lightweight nature of the MLP model, training typically took 3-10 minutes per model, depending on the dataset.

B. Energy and Area Breakdown Comparison

To quantify the modeling accuracy of design breakdowns, we employed the mean absolute percentage error (MAPE) to measure global numerical deviation from the ground truth, and



(a) FEATHER (b) Flex-DPE (c) Gemmini (d) MAERI (e) NVDLA

Fig. 9: Area breakdown of neural network accelerators at 45nm, from the post-layout simulation, Accelergy, and NPUWattch.

Spearman's rank correlation coefficient (ρ) was used to assess monotonic similarity with ground truth rankings. Spearman's coefficient assigns ranks to each component, compares paired ranks to determine their relative differences, and aggregates these into a single value between -1 and 1, indicating how closely the two orderings align.

Fig. 8 compares the energy breakdowns of different neural accelerators at 45nm obtained from post-layout simulation, Accelergy, and NPUWattch. NPUWattch achieves an MAPE of 50.55% with ρ = 0.94, whereas Accelergy yields an MAPE of 129.3% with ρ = 0.62 across the five accelerators. Overall, NPUWattch closely matches post-layout results and accurately identifies the energy-bottleneck components.

Accelergy underestimates the SRAM portions of Flex-DPE and Gemini by 40.6% and 31.3%, respectively. These errors primarily result from heterogeneous modeling of logic and SRAM, with distortion increasing for larger SRAM sizes. This inaccuracy skews overall breakdowns and misrepresents underlying trends. Even when the analysis is limited to logic components, substantial discrepancies persist. Accelergy estimates that the adder network of FEATHER and the NoC of Flex-DPE account for 48.5% and 43.7% of logic energy, whereas empirical measurements show only 8.8% and 35.0%. Such gaps highlight the limited extrapolative capability of rule-based models and demonstrate that logic-only modeling can yield significantly misleading estimates. In contrast, NPUWattch exhibits only a 3.2% error for Gemmini, which has the largest memory capacity among the compared designs (320KB). The minor deviations in NPUWattch's logic estimates arise from distributed control logic embedded in RTL, which is typically excluded during DSE and thus not a major concern. Overall, NPUWattch's error rate is substantially lower than that of

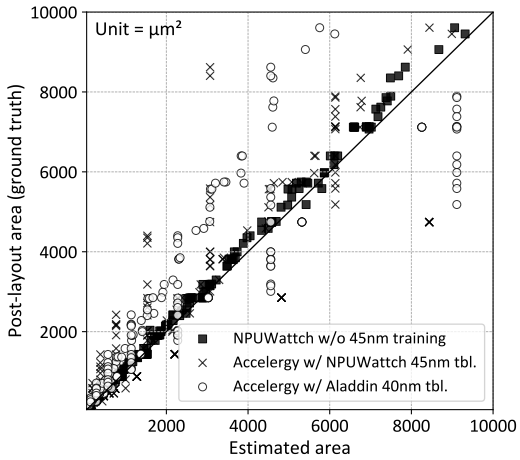


Fig. 10: Comparison between estimated and post-layout areas of hardware primitives under 10,000μm^2 across three estimation methods.

Accelergy, demonstrating its accuracy and robustness across diverse memory configurations.

Fig. 9 displays the area breakdowns of FEATHER, Flex-DPE, Gemmini, MAERI, and NVDLA. The results exhibit trends similar to the energy breakdown patterns. NPUWattch shows superior accuracy with an average MAPE of 32.56% (ρ = 0.90) across accelerators, compared to Accelergy's 225.5% MAPE (ρ = 0.32). In particular, the 256KB global buffers in FEATHER and Gemmini are designed as 32-bank SRAMs. Under this configuration, Accelergy significantly underestimates the multi-bank SRAM area, suggesting that a different modeling approach is required even for banked SRAMs.

Errors in logic components, such as MAC units within PEs, can accumulate over successive iterations and significantly am-

TABLE V: Neural Network Benchmarks

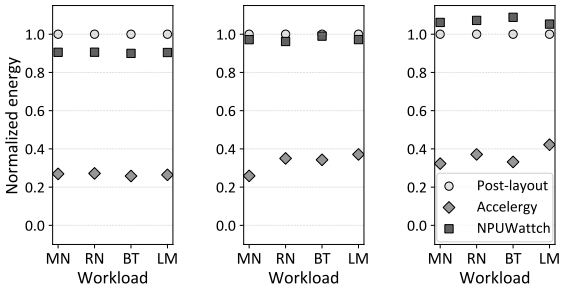
Networks	Abbr.	MACs	Parameters	Features
MobileNetV3	MN	57.8M	2.5M	Depthwise convolution
ResNet-50	RN	3.7G	19.5M	Dense convolution
BERT	BT	43.8G	109.8M	Encoder-only transformer
Llama-3	LM	7.6T	8.0B	Generative AI

plify breakdown distortion. Hence, accurately predicting each individual logic component is critically important. To evaluate the accuracy of NPUWattch, we measured the area deviance of frequently used primitive components, including multiplexers, register files, and arithmetic units. We chose components with areas below 10,000$\mu\mathrm{m}^2$ at the 45nm node, constraining the comparison to realistically sized primitives typically explored during the DSE stage. Post-layout results were then compared across three estimators: NPUWattch trained without 45nm data, Accelergy using a 45nm table derived from NPUWattch, and Accelergy using the default 40nm table from Aladdin.

In Fig. 10, the x-axis represents the estimated values, and the y-axis shows the post-layout results; deviations from the diagonal indicate increasing error. NPUWattch's estimates are tightly clustered along the diagonal, demonstrating strong agreement with post-layout results. In contrast, Accelergy's estimates deviate substantially, with several samples aligned vertically in the estimated-area range of 8,000-10,000$\mu\mathrm{m}^2$, implying that key parameters affecting design cost are not properly captured due to model limitations. We observed that it does not adequately account for the exponent and mantissa bitwidths of floating-point arithmetic units, resulting in identical outputs across multiple configurations. As a result, while NPUWattch achieves a standard deviation of 179.6, Accelergy using the 45nm and 40nm tables records 1478.2 and 1448.2, respectively. NPUWattch delivers estimates 8.2x more consistent than those of Accelergy for logic components, highlighting that rule-based methods can yield substantial errors even when SRAM is excluded. Thus, beyond estimator consistency, component-level nonlinearity must also be considered, and NPUWattch effectively captures this.

C. Workload-level Energy Estimation

PAT modeling frameworks are often integrated with architecture simulators to estimate energy consumption across diverse workloads. We used NeuroSpector [28] to identify the optimal scheduling scheme for each neural layer across different workloads on the target accelerator designs. From these optimal schedules, we extracted activity counts and estimated total energy consumption based on unit access costs derived from NPUWattch and Accelergy at the 45nm node. Using post-layout simulation results as the ground truth, both NPUWattch and Accelergy were evaluated against this baseline. Table V lists the NN benchmarks used in this evaluation, including MobileNetV3 [17], ResNet-50 [16], BERT [10], and Llama-3 [13]. These workloads encompass a diverse set of neural models, ranging from lightweight to foundation models, with varying model sizes and computational complexities.



(a) FEATHER (b) Gemmini (c) NVDLA

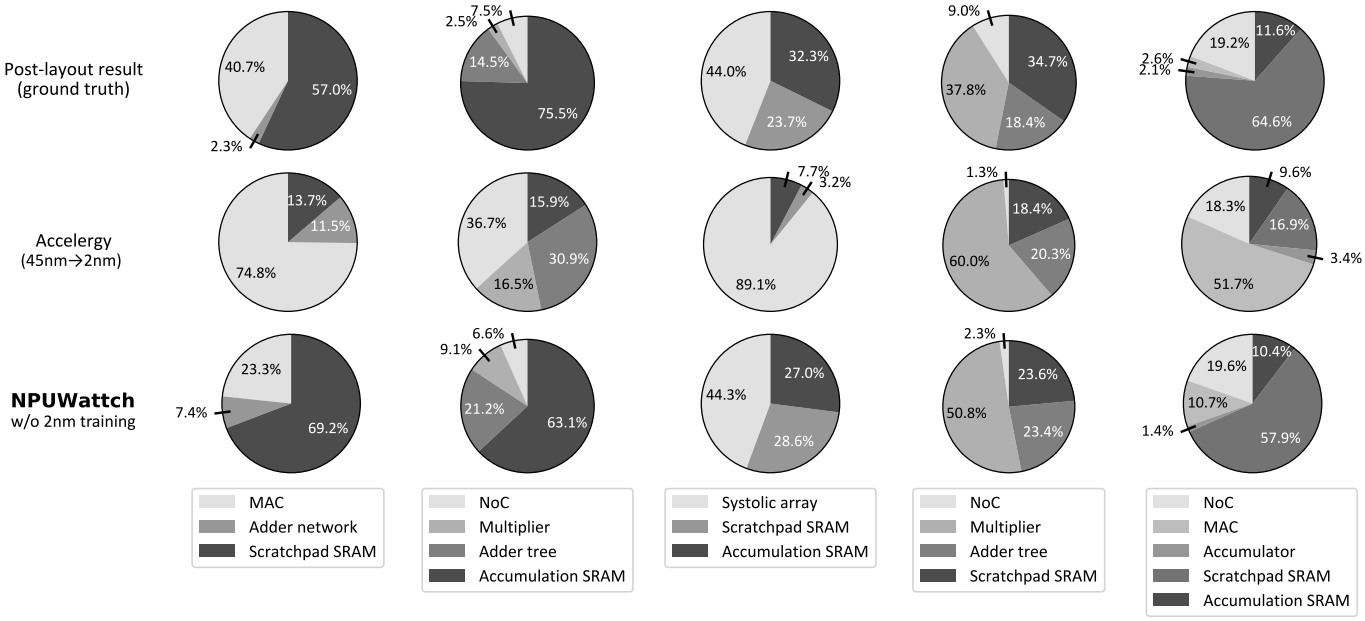
Fig. 11: Relative accelerator energy between NPUWattch and Accelergy cost models using the identical activity counts derived from different workloads on (a) FEATHER, (b) Gemmini, and (c) NVDLA. Energy values are normalized to post-layout results at 45nm.

Fig. 11 shows the total energy consumption of different workloads executed on FEATHER, Gemmini, and NVDLA. All results are normalized to NeuroSpector estimates using post-layout simulation-based energy tables. NPUWattch achieves average error rates of 9.6%, 2.62%, and 6.9% for FEATHER, Gemmini, and NVDLA, respectively, which are 73.4%, 66.93%, and 63.8% more accurate than Accelergy. The discrepancy between post-layout results and Accelergy estimates becomes more pronounced for MobileNetV3, which exhibits the highest global buffer access ratio among all workloads. By underestimating SRAM power, these frequent buffer accesses further reduce the estimated SRAM contribution to total energy consumption.

In contrast, for Llama-3, the feed-forward block imposes a significantly higher computational load, leading to increased overall energy demand. Since Accelergy reports relatively lower error rates for MAC units than for SRAM, it allocates a larger proportion of total energy to MACs as computational intensity increases. As a result, the total energy gap between the ground truth and Accelergy becomes smaller than that observed for MobileNetV3, while the energy breakdown becomes more distorted. Accelergy also exhibits higher energy estimation error for FEATHER compared to the other accelerators. Although Gemmini and FEATHER both incorporate 256 PEs, their global buffer organizations differ in that Gemmini uses separate scratchpad and accumulation memories, whereas FEATHER employs a single unified scratchpad. FEATHER performs partial-sum accumulation within its 256KB unified global buffer, and Accelergy tends to underestimate the corresponding global-buffer access energy for FEATHER relative to Gemmini. As the buffer access count increases in FEATHER, the total energy becomes increasingly underestimated. In contrast, NPUWattch consistently produces closely matching energy estimates across diverse workloads and accelerator configurations, maintaining an average error of 2.7%.

D. Case Studies: Scaling to Advanced Technologies

1) Extrapolation to 2nm Node: With the adoption of new device types, traditional scaling trends observed at earlier technology nodes no longer hold at advanced nodes. Since SRAM



(a) FEATHER (b) Flex-DPE (c) Gemmini (d) MAERI (e) NVDLA

Fig. 12: Energy breakdown of neural network accelerators at 2nm, from the post-layout and NPUWattch without 2nm training.

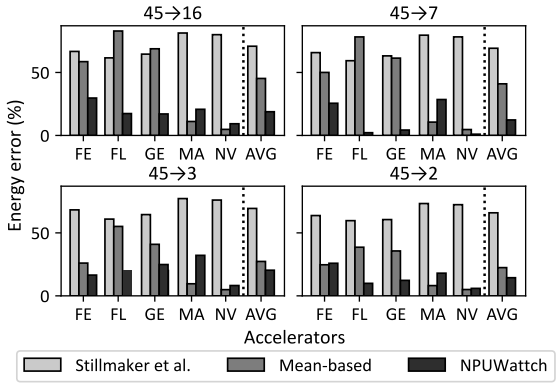


Fig. 13: Energy estimation errors for FEATHER (FE), Flex-DPE (FL), Gemmini (GE), MAERI (MA), NVDLA (NV) under technology scaling from 45nm to 10nm, 7nm, 3nm, and 2nm nodes. NPUWattch achieves consistent energy estimation with an average error of 16.5% across all scaling scenarios.

scales less favorably than logic, the memory contribution to overall design cost is generally expected to increase. Fig. 12 presents the estimated energy breakdowns of accelerators at the 2nm node. The results show that, while the overall memory scaling pattern exists, the rate of increase varies considerably across designs. Comparing post-layout results between 45nm (Fig. 8) and 2nm (Fig. 12), the memory portion increases by 24% for Flex-DPE, 25.2% for Gemmini, and 48% for FEATHER. This divergence is attributed to logic scalability discussed in Section IV-E2, where different logic components exhibit distinct scaling characteristics. In particular, the large NoC in Flex-DPE and the pipeline registers in Gemmini show limited technology scaling, whereas the compact interconnection network in FEATHER scales more efficiently, amplifying the relative memory portion of total energy. NPUWattch effectively extrapolates this behavior even without 2nm data during training, but Accelergy produces nearly identical breakdowns between the 45nm and 2nm results by applying fixed scaling factors, resulting in distorted estimates. Across five accelerators, Accelergy yields an MAPE of 203.93% ($\rho = 0.44$), while NPUWattch achieves a substantially lower MAPE of 61.18% ($\rho = 0.84$), closely tracking the ground truth.

2) Nonlinearity in Scaling Factors: As discussed in Section III-B, a scaling factor cannot be represented by a single static value. To evaluate the accuracy of technology node scaling across arbitrary designs, we analyzed the energy estimates produced by three scaling methods on the accelerators listed in Table IV. The first method applies re-established scaling factors for our transistor models, derived using the approach of Stillmaker et al. [36]. The second method simply uses the arithmetic mean of each scaling scenario shown in Fig. 1, and the third method utilizes NPUWattch by predicting results for a target technology node without the corresponding node data in the training dataset.

Fig. 13 shows the percentage errors for five accelerator models under four different scaling conditions (e.g., 45→16nm). For each technology node, the corresponding post-layout result serves as the baseline. To the left of the dotted line, each bar group represents an accelerator and summarizes the errors produced by the three scaling methods. To the right, the plot reports the mean error across all five accelerators for the given condition. In all cases, the approach of Stillmaker et al. fails to capture the scaling trend of larger designs. The arithmetic-mean method yields a lower mean error than the former but exhibits high variability in accuracy. In contrast, NPUWattch consistently achieves the lowest errors across all

scenarios. These results demonstrate that the NN-based PAT estimation model introduced in Section IV-E3, which leverages carefully selected input features and an adaptive loss function, outperforms conventional and heuristic scaling approaches.

VI. Conclusion

This paper presents *NPUWattch*, an ML-based framework for pre-silicon modeling of power, area, and timing (PAT) in neural accelerators. As design complexity increases and technologies scale to sub-20nm nodes with emerging transistor types such as FinFETs and GAAFETs, traditional analytical and table-based models fail to capture nonlinear scaling and struggle with modeling inconsistencies across logic and memory components. NPUWattch addresses these challenges by building technology libraries spanning advanced nodes down to 2nm and training neural networks on a unified post-layout dataset. It also introduces scaling-aware design features and an adaptive loss function to improve robustness in underrepresented regions of the dataset during training. Experimental results across multiple open-source accelerators demonstrate that NPUWattch outperforms existing tools with an average estimation error of 2.7%, offering reliable and accurate PAT estimation. While the ML-based approach of NPUWattch may diminish model explainability compared to traditional analytical methods, its data-driven nature enables broad portability across different technologies and designs using a consistent methodology. To ensure reliable and accurate PAT estimation, it necessitates a comprehensive dataset constructed under a consistent process conditions for both logic and memory blocks. Moreover, as NPUWattch directly estimates PAT values via regression, training without bias to unevenly distributed design samples in the dataset is crucial. We foresee that NPUWattch can be extended to open hardware including CPUs and GPUs as well as emerging accelerator designs, encompassing chiplets and in-memory computing.

Acknowledgment

We thank all the anonymous reviewers for their constructive comments. This work was supported by the Korea Planning and Evaluation Institute of Industrial Technology (Grant RS-2023-00236772), the National Research Foundation of Korea (Grants RS-2025-00560896, RS-2020-NR047144, and 2022R1I1A3073214 under the Basic Science Research Program), the Institute of Information and Communications Technology Planning and Evaluation (Grant RS-2025-09942968), and Axion, Co., Ltd. The EDA tools were supported by the IC Design Education Center (IDEC) of Korea. Taigon Song and William J. Song are co-corresponding authors.

References

[1] Advanced Micro Devices, "Xilinx Power Estimator User Guide," Publication UG440, pp. 1–128, Nov. 2024. [Online]. Available: https://docs.amd.com/r/en-US/ug440-xilinx-power-estimator

[2] R. Aitken, V. Chandra, B. Cline, S. Das, D. Pietromonaco, L. Shifren, S. Sinha, and G. Yeric, "Predicting Future Complementary Metal-Oxide-Semiconductor Technology – Challenges and Approaches," *IET Computers and Digital Techniques*, vol. 10, no. 6, pp. 315–322, Nov. 2016.

[3] G. Bae, D. Bae, M. Kang, S. Hwang, S. Kim, B. Seo, T. Kwon, T. Lee, C. Moon, Y. Choi, K. Oikawa, S. Masuoka, K. Chun, S. Park, H. Shin, J. Kim, K. Bhuwalka, D. Kim, W. Kim, J. Yoo, H. Jeon, M. Yang, S. Chung, D. Kim, B. Ham, K. Park, W. Kim, S. Park, G. Song, Y. Kim, M. Kang, K. Hwang, C. Park, J. Lee, D. Kim, S. Jung, and H. Kang, "3nm GAA Technology Featuring Multi-Bridge-Channel FET for Low Power and High Performance Applications," *IEEE International Electron Devices Meeting*, Dec. 2018, pp. 28.7.1–28.7.4.

[4] R. Balasubramonian, A. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "CACTI 7: New Tools for Interconnect Exploration in Innovative Off-Chip Memories," *ACM Transactions on Architecture and Code Optimization*, vol. 14, no. 2, pp. 1–25, June 2017.

[5] K. Bhanushali and W. Davis, "FreePDK15: An Open-Source Predictive Process Design Kit for 15nm FinFET Technology," *ACM International Symposium on Physical Design*, Mar. 2015, pp. 165–170.

[6] BSIM Group, University of California, Berkeley, "BSIM-CMG Model." [Online]. Available: http://bsim.berkeley.edu/models/bsimcmg

[7] I. Chaturvedi, B. Godala, Y. Wu, Z. Xu, K. Iliakis, P. Eleftherakis, S. Xydis, D. Soudris, T. Sorensen, S. Campanoni, T. Aamodt, and D. August, "GhOST: A GPU Out-of-Order Scheduling Technique for Stall Reduction," *ACM/IEEE International Symposium on Computer Architecture*, June 2024, pp. 1–16.

[8] Y. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[9] H. Choi, M. Lee, C. Lee, J. Yang, and R. Seong, "Machine Learning Optimal Ordering in Global Routing Problems in Semiconductors," *Scientific Reports*, vol. 14, no. 31077, pp. 1–17, Dec. 2024.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2019, pp. 4171–4186.

[11] Y. Du, Z. Guo, X. Jiang, Z. Chai, Y. Zhao, Y. Lin, R. Wang, and R. Huang, "PowPrediCT: Cross-Stage Power Prediction with Circuit-Transformation-Aware Learning," *ACM/IEEE Design Automation Conference*, no. 36, June 2024, pp. 1–6.

[12] G. Gerogiannis and J. Torrellas, "Micro-Armed Bandit: Lightweight & Reusable Reinforcement Learning for Microarchitecture Decision-Making," *IEEE/ACM International Symposium on Microarchitecture*, Oct. 2023, pp. 698–713.

[13] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret *et al.*, "The Llama 3 Herd of Models," arXiv:2407.21783, pp. 1–92, July 2024.

[14] Y. Gu, A. Subramaniyan, T. Dunn, A. Khadem, K. Chen, S. Paul, M. Vasimuddin, S. Misra, D. Blaauw, S. Narayanasamy, and R. Das, "GenDP: A Framework of Dynamic Programming Acceleration for Genome Sequencing Analysis," *ACM/IEEE International Symposium on Computer Architecture*, no. 25, June 2023, pp. 1–15.

[15] M. Guthaus, J. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, "OpenRAM: An Open-Source Memory Compiler," *IEEE/ACM International Conference on Computer-Aided Design*, no. 93, Nov. 2016, pp. 1–6.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2016, pp. 770–778.

[17] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. Le, and H. Adam, "Searching for MobileNetV3," *IEEE/CVF International Conference on Computer Vision*, Oct. 2019, pp. 1314–1324.

[18] Intel Corporation, "Early Power Estimator User Guide," Publication UG-01070, pp. 1–49, July 2021. [Online]. Available: https://www.intel.com/programmable/technical-pdfs/683272.pdf

[19] A. Kahng, B. Lin, and S. Nath, "ORION3.0: A Comprehensive NoC Router Estimation Tool," *IEEE Embedded Systems Letters*, vol. 7, no. 2, pp. 41–45, June 2015.

[20] V. Kandiah, S. Peverelle, M. Khairy, J. Pan, A. Manjunath, T. Rogers, T. Aamodt, and N. Hardavellas, "AccelWattch: A Power Modeling Framework for Modern GPUs," *IEEE/ACM International Symposium on Microarchitecture*, Oct. 2021, pp. 738–753.

[21] T. Kim, S. Woo, H. Kim, M. Kim, S. Ryu, Y. Oh, and T. Song, "NS3K: A 3nm Nanosheet FET Standard Cell Library Development and Its Impact," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 31, no. 2, pp. 163–176, Feb. 2023.

[22] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. Kim, T. Aamodt, and V. Reddi, "GPUWattch: Enabling Energy Optimizations in GPGPUs," *ACM/IEEE International Symposium on Computer Architecture*, June 2013, pp. 487–498.

[23] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," *IEEE/ACM International Symposium on Microarchitecture*, Dec. 2009, pp. 469–480.

[24] Microelectronics Co-design Research Group, University of Minnesota, "Predictive Technology Model (PTM)." [Online]. Available: https://mec.umn.edu/ptm

[25] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "CACTI 6.0: A Tool to Understand Large Caches," HP Laboratories, pp. 1–20, Jan. 2009.

[26] NVIDIA Corporation, "NVDLA." [Online]. Available: https://github.com/nvdla

[27] A. Parashar, P. Raina, S. Shao, Y. Chen, V. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. Keckler, and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," *IEEE International Symposium on Performance Analysis of Systems and Software*, Mar. 2019, pp. 304–315.

[28] C. Park, B. Kim, S. Ryu, and W. Song, "NeuroSpector: Systematic Optimization of Dataflow Scheduling in DNN Accelerators," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 8, pp. 2279–2294, Aug. 2023.

[29] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training," *IEEE International Symposium on High-Performance Computer Architecture*, Feb. 2020, pp. 58–70.

[30] B. Rouhani, R. Zhao, A. More, M. Hall, A. Khodamoradi, S. Deng, D. Choudhary, M. Cornea, E. Dellinger, K. Denolf, S. Dusan, V. Elango, M. Golub, A. Heinecke, P. James-Roxby, D. Jani, G. Kolhe, M. Langhammer, A. Li, L. Melnick, M. Mesmakhosroshahi, A. Rodriguez, M. Schulte, R. Shafipour, L. Shao, M. Siu, P. Dubey, P. Micikevicius, M. Naumov, C. Verrilli, R. Wittig, D. Burger, and E. Chung, "Microscaling Data Formats for Deep Learning," arXiv preprint arXiv:2310.10537, pp. 1–9, Oct. 2023.

[31] S. Sarangi and B. Baas, "DeepScaleTool: A Tool for the Accurate Estimation of Technology Scaling in the Deep-Submicron Era," *IEEE International Symposium on Circuits and Systems*, May 2021, pp. 1–5.

[32] G. Shahidi, "Chip Power Scaling in Recent CMOS Technology Nodes," *IEEE Access*, vol. 7, pp. 851–856, Dec. 2018.

[33] S. Shao, B. Reagen, G. Wei, and D. Brooks, "Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures," *ACM/IEEE International Symposium on Computer Architecture*, June 2014, pp. 97–108.

[34] Y. Shin, D. Park, D. Koh, D. Heo, J. Park, H. Lee, J. Kim, H. Lee, and T. Song, "FS2K: A Forksheet FET Technology Library and a Study of VLSI Prediction for 2nm and Beyond," *IEEE International Symposium on Circuits and Systems*, May 2024, pp. 1–5.

[35] J. Stein, J. Chen, I. Castellanos, G. Sundararajan, M. Qayam, P. Kumar, J. Remington, and S. Sohoni, "FreePDK v2.0: Transitioning VLSI Education Towards Nanometer Variation-Aware Designs," *IEEE International Conference on Microelectronic Systems Education*, July 2009, pp. 100–103.

[36] A. Stillmaker and B. Baas, "Scaling Equations for the Accurate Prediction of CMOS Device Performance From 180nm to 7nm," *Integration*, vol. 58, pp. 74–81, June 2017.

[37] J. Stojkovic, C. Liu, M. Shahbaz, and J. Torrellas, "uManycore: A Cloud-Native CPU for Tail at Scale," *ACM/IEEE International Symposium on Computer Architecture*, June 2023, pp. 1–15.

[38] Synergy Lab, Georgia Institute of Technology, "FEATHER." [Online]. Available: https://github.com/maeri-project/FEATHER

[39] Synergy Lab, Georgia Institute of Technology, "MAERI." [Online]. Available: https://github.com/maeri-project/MAERI_bsv

[40] Synergy Lab, Georgia Institute of Technology, "SIGMA." [Online]. Available: https://github.com/georgia-tech-synergy-lab/SIGMA

[41] Synopsys, "PrimeTime PX User Guide," Version L-2016.06, pp. 1–191, June 2016.

[42] UC Berkeley Architecture Research, University of California, Berkeley, "Gemmini." [Online]. Available: https://github.com/ucb-bar/gemmini

[43] V. Vashishtha and L. Clark, "ASAP5: A Predictive PDK for the 5nm Node," *Microelectronics Journal*, vol. 126, p. 105481, Aug. 2022.

[44] V. Vashishtha, M. Vangala, and L. Clark, "ASAP7 Predictive Design Kit Development and Cell Design Technology Co-optimization," *IEEE/ACM International Conference on Computer-Aided Design*, Dec. 2017, pp. 992–998.

[45] S. Wu, C. Chang, M. Chiang, C. Lin, J. Liaw, J. Cheng, J. Yeh, H. Chen, S. Chang, K. Lai, M. Liang, K. Pan, J. Chen, V. Chang, T. Luo, X. Wang, Y. Mor, C. Lin, S. Wang, M. Hsieh, C. Chen, B. Wu, C. Lin, C. Liang, C. Tsao, C. Li, C. Chen, C. Hsieh, H. Liu, P. Chen, C. Chen, R. Chen, Y. Yeo, C. Chui, W. Chang, T. Lee, K. Huang, H. Lin, K. Chen, M. Tsai, K. Chen, X. Chen, Y. Cheng, C. Wang, W. Shue, Y. Ku, S. Jang, M. Cao, L. Lu, and T. Chang, "A 3nm CMOS FinFlex™ Platform Technology with Enhanced Power Efficiency and Performance for Mobile SoC and High Performance Computing Applications," *IEEE International Electron Devices Meeting*, Dec. 2022, pp. 27.5.1–27.5.4.

[46] Y. Wu, J. Emer, and V. Sze, "Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs," *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2019, pp. 1–8.

[47] Z. Xie, Y. Huang, G. Fang, H. Ren, S. Fang, Y. Chen, and J. Hu, "RouteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network," *IEEE/ACM International Conference on Computer-Aided Design*, Mar. 2018, pp. 1–8.

[48] Y. Zhang, H. Ren, and B. Khailany, "GRANNITE: Graph Neural Network Inference for Transferable Power Estimation," *ACM/IEEE Design Automation Conference*, no. 60, June 2020, pp. 1–6.

[49] Y. Zhou, H. Ren, Y. Zhang, B. Keller, B. Khailany, and Z. Zhang, "PRIMAL: Power Inference Using Machine Learning," *ACM/IEEE Design Automation Conference*, no. 39, June 2019, pp. 1–6.