Data-pattern-based Predictive On-Chip Power Meter in DNN Accelerator

Jian Peng, *Member, IEEE*, Tingyuan Liang, *Member, IEEE*, Jingbo Jiang, *Member, IEEE*, Yipu Zhang, *Member, IEEE*, Zhe Lin, *Member, IEEE*, Zhiyao Xie, *Member, IEEE*, and Wei Zhang, *Senior Member, IEEE*

***Abstract*—Advanced power management techniques, such as voltage drop mitigation and fast power management, can greatly enhance energy efficiency in contemporary hardware design. Nevertheless, the implementation of these innovative techniques necessitates accurate and fine-grained power modeling, as well as timely responses for effective coordination with the power management unit. Additionally, existing performance-counter-based and RTL-based on-chip power meters have difficulty in providing sufficient response time for fast power and voltage management scenarios. In this paper, we propose PROPHET, a data-pattern-based power modeling method for multiply-accumulate-based (MACC) DNN accelerators. Our proposed power model extracts the pre-defined data patterns during memory access and then a pre-trained power model can predict the dynamic power of the DNN accelerators. Thus, PROPHET can predict dynamic power and provide sufficient responding time for power management units. In the experiments, we evaluate our predictive power model in four DNN accelerators with different dataflows and data types. In power model training and verification, our proposed data-patterns-based power model can realize the 2-cycle temporal resolution with $R^2 > 0.9$, $NMAE < 7\%$, and the area and power overhead lower than 4.5%.**

***Index Terms*—Power modeling, on-chip power meter, power prediction, data patterns, DNN accelerator.**

I. Introduction

DEEP neural network (DNN) accelerator is a specialized architecture designed for DNN algorithms. With the tremendous success of machine learning and neural networks in various domains such as computer vision, numerical statistics, and design automation, DNN accelerators have gained significant attention in recent years due to their high energy efficiency and throughput. As DNN algorithms continue to evolve, the performance demand for DNN accelerators are increasing. Therefore, there is a need to integrate more processing elements and complex architectures into DNN accelerators.

For example, Diannao [1], one of the earliest DNN accelerators proposed in 2014, had limited processing capabilities with 64 integer multipliers and adders and did not support floating-point data types. In 2015, Pudiannao [2] was introduced, which had 276 multipliers and 784 adders, supporting both integer and floating-point data types for machine learning acceleration. Similarly, Google developed its TPU series, with the first generation TPUv1 launched in 2017 featuring 65536 8-bit integer multipliers and adders [3]. Subsequent generations, TPUv2 and TPUv3, became more complex, incorporating multiple cores and supporting multiple data types for DNN inference and training tasks [4]. Additionally, Huawei's Davinci [5], introduced in 2019, offers higher throughput and complexity compared to TPUv3.

However, as the complexity and parallelism of DNN accelerators increase, power consumption becomes a critical concern in their design. Power management for DNN accelerators faces two challenges. The first challenge is allocating power consumption among different cores within a limited power budget. The second challenge is maintaining power intensity while ensuring high energy efficiency. The large parallelism in DNN accelerators leads to significant power fluctuations and voltage drops during operation. These voltage drops will degrade energy efficiency and system reliability. For instance, NVDLA [6], an open-source DNN accelerator developed by NVIDIA, can exhibit peak power levels that are more than five times higher than its average power consumption. Consequently, it becomes imperative to allocate an adequate voltage margin to mitigate voltage drops and their adverse impact on energy efficiency.

In power management, on-chip power meters (OPMs) play a crucial role in providing precise and timely power traces. The specific requirements for OPMs may vary depending on the application. To address the first challenge, dedicated dataflow scheduling and dynamic voltage and frequency scaling (DVFS) techniques can be employed. Previous researches [7]–[13] have demonstrated substantial energy savings achieved through the implementation of energy-oriented scheduling and fine-grained DVFS techniques in DNN accelerators. These techniques, typically managed by system firmware and/or operating systems (OS), often necessitate coarse-grained temporal resolution in power tracing.

Addressing the second challenge involves mitigating the voltage noise effects caused by parasitic parameters in the power delivery network (PDN). The voltage noise effects can span from 10 to 200 clock cycles [14]. For instance, the parasitic inductor in the package can induce voltage noise within the dynamic range of 1-10MHz [15]. Moreover, the LdI/dt

Manuscript received XXXX; revised XXXX; accepted XXXX. Data of publication XXXX; Date of current version XXXX. This work is partially funded by Hong Kong RGC GRF 16213521 and Huawei Hong Kong Research Center. Corresponding author: Wei Zhang

J. Peng (jpengai@connect.ust.hk), J. Jiang (jjiangan@connect.ust.hk), Y. Zhang (yzhangqg@connect.ust.hk), Z. Xie (eezhiyao@ust.hk), and W. Zhang (wei.zhang@ust.hk) are with the Dept. of Electronic and Computer Engineering, Hong Kong University of Science and Technology.

T. Liang (tliang@connect.ust.hk) is with Huawei. This work was done when he was with HKUST.

Z. Lin (linzh235@mail.sysu.edu.cn) is with the School of Integrated Circuit, Sun Yat-sen University

voltage noise effects caused by on-chip parasitic inductors can occur within 10 clock cycles in modern computing architectures [16]. To tackle these issues and enhance reliability and energy efficiency, advanced power management techniques have been proposed in recent years. Previous studies conducted on CPU and GPU platforms [14], [17]–[19] have shown that implementing appropriate voltage drop techniques can reduce the required voltage margin by up to 20%. However, these advanced techniques impose stringent requirements on OPMs. They must be capable of providing power traces with fine-grained temporal resolution and sufficient response time for the power management unit.

Researchers have made efforts to develop fine-grained and lightweight OPMs. Some automatic frameworks have been proposed to construct RTL-based OPMs with low overhead and fine-grained temporal resolution. These frameworks typically select representative signals as proxies and monitor their switch activities to estimate power consumption. For example, PowerProbe [20] achieves an average error lower than 10% and a temporal resolution at the level of tens of cycles within area overhead of 7%. Simmani [21] achieves per-cycle temporal resolution but is computationally expensive due to quadratic regression in model training. State-of-the-art works, APOLLO [16] and DEEP [22], can achieve per-cycle resolution for microprocessors with less than 1% overhead. However, we argue that these RTL-based OPMs may not be the optimal solution for DNN accelerators. Firstly, as data-stream accelerators, DNN accelerators have limited event signals, and their power consumption is heavily influenced by input data. Secondly, due to their monitoring principle, RTL-based OPMs must track switch activities of proxies leading to insufficient response time for downstream power management units.

The previous study by Liu et al. [8] leverages the sparsity of feature data to predict the workload of DNN accelerators and optimize energy consumption through proper DVFS settings. However, this approach is limited to predicting the workload within several milliseconds, and it becomes less accurate when both weight data and feature data are sparse. The main challenge for data-pattern-based predictive OPMs lies in identifying precise and fine-grained data patterns that can be easily sampled during memory access without imposing significant overhead. To address these challenges, we propose a novel data-pattern-based OPM called PROPHET, as illustrated in Figure 1. The core concept of PROPHET is to define and analyze specific patterns that accurately reflect the dynamic power of DNN accelerators and can be efficiently extracted during memory access with minimal overhead. By sampling these patterns during memory access, we achieve advanced prediction of forthcoming power consumption several clock cycles in advance. Moreover, compared to existing approaches that rely on proxies derived from RTL signals or performance counters, our proposed power model enables accurate power prediction with low overhead.

A preliminary version of this work appears in [23]. The previous version only explored and evaluated data patterns from input feature data. The OPM was designed to sample data patterns from feature data ports. Additionally, the hardware platform for data patterns evaluation was limited to the output-stationary systolic array (SA). However, PE arrays in different DNN accelerators may consist of various data flows in real-world applications. Meanwhile, weight data is also sparse leading to more difficulties in data pattern sampling. In this paper, PROPHET will take both feature and weight data into consideration. Then we proposed a 2D sampler array to extract data patterns from weight and feature data simultaneously. The data flow of the sampler array should be the same as the PE array for fine-grained power prediction. Since the sampler array can recover the dataflow of PE array instead of using a large sliding window to record, our PROPHET can realize higher accuracy and fine-grained temporal resolutions. In the experiment, we make a comprehensive exploration and evaluation of PROPHET in four hardware platforms which consist of different dataflows and data types. Compared with previous works that select proxies from RTL signals or performance counters, our proposed power model can achieve accurate power prediction with low overhead. Our experiments have demonstrated PROPHET's accuracy and fine-grained prediction capabilities. Our new contributions are summarized below.

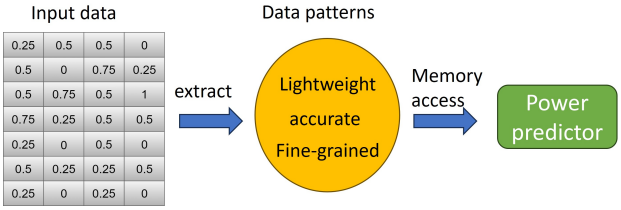

Fig. 1: We analyze and define some lightweight, accurate, and fine-grained data patterns. PROPHET can extract them from input data of DNN accelerators and then predict runtime power based on sampling data patterns.

- We propose PROPHET, a fine-grained predictive power model for DNN accelerators that enables improved power management and increased energy efficiency. To our best knowledge, PROPHET is the first predictive power model for DNN accelerators. PROPHET achieves power prediction in advance by sampling target data patterns during memory access.
- We make a comprehensive evaluation for our proposed PROPHET with PE arrays with different data flows and data types. Furthermore, we explain and analyze our proposed data patterns by comparing the selected proxies in RTL-based OPM constructions.
- We proposed a 2D sampler array to sample data patterns from both weight and feature data ports. The sampler consists of several logic gates resulting in a low overhead implementation.
- PROPHET can achieve low overhead and high performance at the same time. The area and power overhead of PROPHET are 2-4.5% in our experiments, and the temporal resolution can achieve 2 clock cycles with $R^2 > 0.9$ and NMAE $< 7\%$.

II. Related Works

A. On-chip Power Meters

Accuracy and overhead are two crucial factors when considering on-chip power meters. An effective OPM should not only provide accurate measurements but also have low overhead. Previous research has explored the use of performance counters to estimate power consumption during runtime for both CPU and GPU architectures [9]–[11], [13]. In these studies, micro-architecture events such as cache misses and retired instructions are counted within each power measurement window, which typically spans several thousand cycles. A regression model is then trained to estimate the average power consumption within each measurement window based on the event count. For example, Zhang et al. [13] constructed a runtime power model based on performance counters of important architectural registers to adjust hardware performance in image rendering, resulting in energy savings and high image quality. However, these models are typically used in coarse-grained management scenarios due to the long access time of event registers at the operating system level. Furthermore, they lack flexibility because users can't access part of architectural event registers.

In addition, several RTL-based runtime power models have been proposed for advanced power management techniques, with the aim of achieving higher temporal resolution and lower overhead [16], [20]–[22], [24], [25]. These models select the most power-correlated RTL signals, known as power proxies, as inputs to the power model. The works of [24], [25] consider all the selection registers and employ lasso regression as proxy selection and model training methods, achieving approximately 1K temporal resolution and 10% overhead. In particular, PowerProbe [20] is an automated framework that hierarchically selects the ports of each module. It can construct on-chip power meters with a temporal resolution of 100-1K clock cycles while maintaining a resource overhead of less than 8%. To achieve higher temporal resolution and lower overhead, some non-linear methods have been proposed. Lin et al. [26] apply decision tree regression to construct the OPM in FPGA, which can achieve 2.4-6 times lower overhead compared to the commonly used linear model. Furthermore, Simmani [21] introduces polynomial regression in model training, achieving per-cycle temporal resolution for the Rocket RISC-V microprocessor. However, due to the complexity of polynomial regressions, the overhead of OPM in Simmani is extremely large. It demonstrated that this RTL-based power model has the potential to achieve the temporal resolution of 1 clock cycle. Unfortunately, previous works are limited by the extra area overhead to improve resolution, significantly reducing the freedom of proxy selection and the underlying power model [16]. The state-of-the-art solutions, APOLLO [16] and DEEP [22], have made significant advances in constructing OPMs with a per cycle temporal resolution and less than 1% overhead. These solutions achieve this by employing intelligent proxy selection methods. However, it is worth noting that these methods are primarily designed for architectures with intricate control flow and may not be directly applicable to data-streaming-based DNN accelerators. Additionally, while the RTL-based OPMs in these solutions monitor the switch activities of selected proxies, their response time may still be insufficient for advanced power management techniques.

B. Voltage Management

In recent years, some techniques that improved the energy efficiency of DNN accelerators by reducing the supply voltage have been proposed. This type of method shows a large potential for energy efficiency enhancement [27]. In CPU architecture, proper mitigation of voltage emergency [14], [28], [29] can reduce the voltage margin over 20% with little delay overhead. The works of [19], [30] demonstrate that the worst-case voltage drop can be degraded up to 29% in GPU architecture. The basic concept to suppress voltage noise is to construct a power meter to obtain an accurate and fine-grained power trace, and then an actuator will detect the dangerous voltage drop and take actions like throttling or proactive clock gating. There is a feedback loop from the hardware system, OPM, actuator, and then back hardware system. [28] has demonstrated the unsuppressed voltage emergency will increase up to 50% when the feedback loop delay is over 2 clock cycles. Hence, the OPM should not only be accurate and fine-grained but also provide sufficient responding time for the downstream modules in modern voltage management techniques. The works of [28], [29] proposed to predict the voltage emergency based on the micro-architecture events and take actions proactively. Meanwhile, [19], [31] applied a fine-grained OPM to monitor the power fluctuation to smooth the voltage noise. However, due to the processing delay of OPM and feedback loop delay from OPM to the power management unit, the insufficient responding time left for the voltage management is still the bottleneck to be addressed.

C. Energy-Efficient DNN Accelerators

Numerous techniques have been proposed to improve the energy efficiency of DNN accelerators from various perspectives. Sparse DNN accelerators, such as SCNN [32] and Cambricon-X [33], optimize power consumption for sparse DNN models and data. However, these sparse DNN accelerators are mainly effective for sparse models and are commonly used in embedded systems. In most scenarios, general DNN accelerators that support both dense and sparse models are more popular. For FPGA-based DNN accelerators, offline dataflow analysis and scheduling techniques, as demonstrated in [7], can achieve energy savings of 10-30%. DVFS at the level of per PE, as shown in [8], can reduce the dynamic power consumption by more than 50% in DNN accelerators. In terms of supply voltage reduction, error-tolerant DNN accelerators have been developed to tolerate timing errors caused by low supply voltage, resulting in higher energy efficiency in specific applications [34]–[36]. However, due to limitations in error checking and recovery, timing errors can still occur and affect precision, making this approach suitable only for low-precision scenarios. Another approach to reduce supply voltage is the design of adaptive elastic clocks for low-supply-voltage DNN accelerators [37]. These elastic

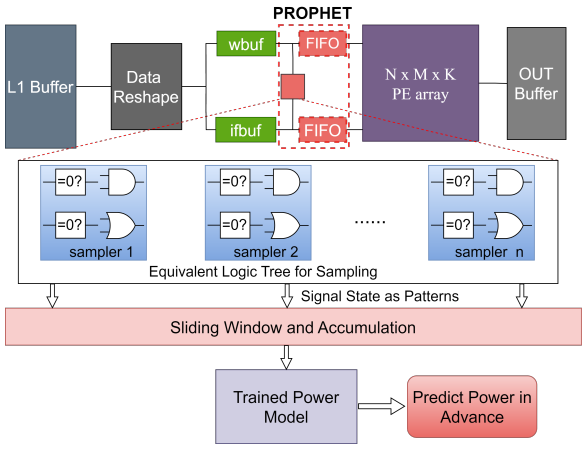

Fig. 2: PROPHET predicts the runtime power in DNN accelerators. FIFOs generate the time interval between weight buffer (wbuf), input feature buffer (ifbuf), and PE array, which makes PROPHET samples and then obtains runtime power in advance.

clocks can adjust the clock frequency adaptively when timing errors occur. However, this technology requires sophisticated clock circuit design, which can lead to long design periods, especially for large DNN accelerator designs. Therefore, we believe that predictive OPMs, which can predict accurate and fine-grained power traces dozens of clock cycles in advance, have significant potential for future fast power management techniques. Unfortunately, there is limited research exploring power prediction specifically for DNN accelerators.

To address the need for accurate and timely power traces in fast power management, we propose PROPHET, the first predictive and fine-grained on-chip power meter for DNN accelerators. Figure 2 illustrates how PROPHET can be implemented between the weight buffer (*wbuf*), input feature buffer (*ifbuf*), and PE array. The sampling logic trees, consisting of comparators and logic gates, can extract predefined data patterns during memory access. These patterns are accumulated using a small sliding power measurement window to construct the input vector, enabling high temporal resolution. Finally, the power consumption value is calculated using offline trained parameters and the sampled input vector. PROPHET includes FIFOs that generate time intervals between buffers and the PE array, allowing PROPHET to sample and predict power in advance. The number of clock cycles that PROPHET can predict inadvance is determined by the time interval generated by the FIFOs. Table I provides a comparison of recent power modeling methods to highlight the contributions of PROPHET. Compared to previous methods, PROPHET can accurately and finely predict the power waveform several dozen clock cycles in advance with low overhead.

Method	Type	Resolution	Cost	Predictive
[9]–[11], [13], [40], [41]	event counters	$\geq 1K$ cycles	low	✗
[20], [24], [25], [42]	signal proxies	≥ 100 cycles	$5 \sim 15\%$	✗
[21]		per-cycle	$> 100\%$	
[16], [22]		per-cycle	$< 1\%$	
OURS	data patterns	≥ 2 cycles	$2 \sim 4.5\%$	✓

TABLE I: Comparison among various power models

III. Background

A. Power consumption of hardware design

In general, the power consumption of the hardware design can be decomposed into static power [43] and dynamic power [44], which can be expressed by the following equations:

$$P_{total} = P_{static} + P_{dyn} = V_{dd} I_{leakage} + \sum_{i \in I} \alpha_i V_{dd}^2 C_i f \quad (1)$$

From Eq. 1, we can see that the power is the sum of products of signal switching activity α_i, capacitance C_i on the net $i \in I$, supply voltage V_{dd} and operating frequency f. The static power consumption is caused by reverse-bias leakage current $I_{leakage}$ between diffused regions and the substrates of transistors, irrespective of the workloads. In addition, the changes in process, voltage, and temperature (PVT) also affect the static power consumption. In contrast, dynamic power consumption is introduced by signal transitions which dissipate power by repeatedly charging and discharging the load capacitors. For DNN accelerators, the dynamic power will be much greater than its static power and its fluctuation can affect the system's reliability. Hence, in this work, we focus on dynamic power prediction.

B. General PE Array Construction

The PE array serves as the matrix unit and is a crucial component of DNN accelerators for performing matrix multiplications. In the DNN model, convolution layers and fully connected layers, which can be converted into a series of matrix multiplication operations, dominate the workload throughout the process. Therefore, the multiply-accumulate circuit plays a key role in DNN accelerators.

As shown in Figure 3a, the PE array in DNN accelerators typically consists of numerous homogeneous PEs. The DNN accelerators can be classified based on the different data flows. For example, in weight-stationary SAs like Google TPU [3], the feature data and partial sums are transferred between PEs. Similarly, in output-stationary SAs like Davinci [5], [39], the feature data and weight data are transferred between PEs, while the partial sums are reused within the PE. Another common dataflow in the PE array is broadcasting all feature/weight data to all PEs. This type of dataflow is typically found in smaller designs like NVDLA [6] and the Diannao family [38].

For these PE arrays in DNN accelerators, the PE can be categorized into two commonly used types of Multiply-Accumulate structures, as shown in Figure 3b and Figure 3c. In Figure 3b, each PE can perform MACC calculations using multiple multipliers and adders. Registers can be inserted into the data path of the MACC circuit to reduce the critical path delay. This type of MACC structure is typically implemented in output-stationary SAs and broadcasting-based PE arrays. In Figure 3c, each PE only includes one multiplier and adder, and the final partial sum is accumulated outside the PE array. This is the classical structure used in all weight-stationary SAs. Therefore, in this work, we explore our data-pattern-based OPM for these two types of MACC structures.

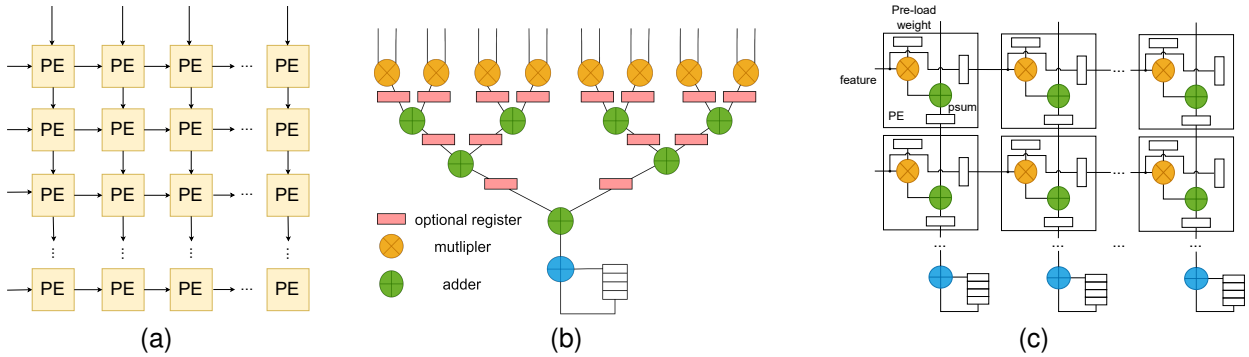

Fig. 3: PE array and two widely-used MACC structures in DNN accelerators. (a) PE array in DNN accelerators, the dataflow may vary in different DNN accelerators. (b) PE in NVDLA [6], DianNao [38], and Davinci [39], includes multiple multipliers and adders. (c) The classical PE structure in weight-stationary SAs.

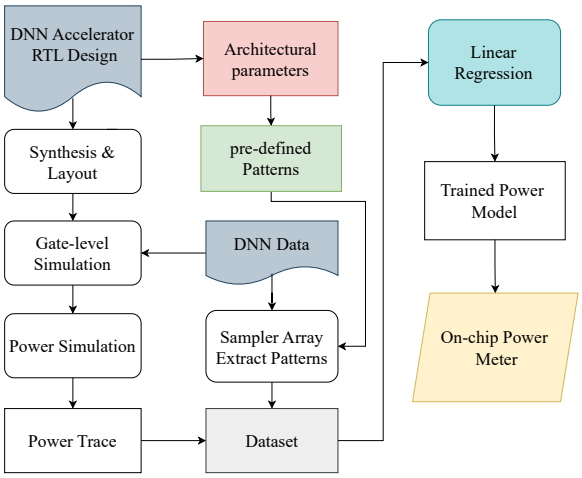

Fig. 4: The overall development framework of PROPHET.

C. Power Breakdown of DNN Accelerators

In DNN models, matrix multiplication plays a significant role compared to other functional units like pooling and activation. This is primarily due to the high data reuse and parallelism exhibited in matrix multiplication. Consequently, during the execution of convolution layers or matrix multiplications, the MACC units and on-chip buffers tend to contribute the most to the dynamic power consumption. A notable example is DNN Weaver, which was introduced in MICRO 2016 [45]. The power consumption of the PE array and on-chip buffers accounts for approximately 94% of the total power consumption. In contrast, the function unit (FU) responsible for ReLU activation and pooling only contributes around 5% to the power consumption. Given that DNN Weaver shares architectural similarities with well-known DNN accelerators like TPU [3], Diannao [1], and NVDLA [6], its power breakdown serves as a representative model for DNN accelerators. Thus, our research primarily focuses on analyzing the MACC units and on-chip buffers as the key factors influencing power consumption.

Symbol	Description
N	Rows of PE array(feature input)
M	Columns of PE array(weight input)
K	Input (feature, weight) pairs in each PE
L	Length of local PE pipeline
N_{sw}	The size of sliding window
R	Temporal resolution of power model

TABLE II: Description of frequently used symbols.

IV. Methodology

A. Overall Framework of PROPHET

Figure 4 shows the PROPHET development framework. It first conducts accurate power simulations with input data of DNN model to gather ground-truth power traces. Simultaneously, based on our definitions, the proposed data patterns are extracted from these input data in the sampler array. These patterns have a strong correlation with the power consumption of the specific SA architecture. The details of our data pattern definitions will be presented in Subsection IV-B. Next, the extracted data patterns are combined with the power traces to create the dataset for power model training. The details of this dataset extraction and sampling procedure will be discussed in Subsection IV-C. Finally, since PROPHET needs to sample the fine-grained data patterns for both weight and feature buffers, we implement 2D sampler array on the input ports of PE array for data pattern sampling. The input FIFOs is optional depending on whether the prediction is required, more detail will be discussed in Subsection IV-E.

B. Define Data Patterns

Data patterns refer to the various combinations of input data into logic gates that result in different dynamic power consumption. In DNN algorithms, the activation functions, particularly the commonly used ReLU function, tend to introduce sparse data into the feature map of intermediate layers by generating zero outputs. On the other hand, the compression DNN models also introduce a lot of zero data in the weight data, such as weight pruning, quantification, and so on. We have discovered that the zeros in the PE arrays' inputs significantly impact the dynamic power consumption of PE array. The work of [8] has demonstrated that the sparsity of

input feature maps can be a pattern to predict the workload for several milliseconds. However, such a sparsity level can only reflect the power consumption in coarse-grained temporal resolution. Furthermore, they only consider the dense DNN model whose weight data are all non-zero. It will be inaccurate when taking both feature and weight into consideration. Therefore, it is still necessary to identify more data patterns for fine-grained and accurate power modeling to construct the power model for timely voltage management.

As illustrated in Figure 5, most DNN accelerators comprise an $N \times M \times K$ array with homogeneous PEs. In each PE, K multipliers and K adders can perform multiple-accumulate operations for matrix multiplications. The weight-stationary systolic array is a special case with $k = 1$. The arithmetic units in the PE can be abstracted as a combination of combinational logic gates G, registers, and local controllers. The dynamic power consumption of the entire DNN accelerator during the time window T can be represented by Eq. 2. P_{dyn_m}, P_{dyn_a}, and P_{data} are the dynamic power of multipliers, adders, and data transformation in PE array, respectively. P_{ctrl} is the power consumption generated by local control logic, which is less affected by the input data. Thus, P_{ctrl} can be approximated as a constant. The P_{data} is the dynamic power generated by the data access in memory and registers. For multipliers and adders in PE, their two inputs are denoted as a and b. The average dynamic power consumption of multipliers and adders over the time window T can be formulated as Eq. 3 and Eq. 4, respectively. Here, $\alpha_g(a,b)$ represents the toggle rate of the combinational logic g with respect to the input data (a,b) of the multiplier or adder, and C_g denotes the average capacitance of the combinational logic g. The toggle rate $\alpha_g(a,b)$ follows a certain distribution $\alpha_g(a,b) \in \mathcal{D}_g(a,b)$.

$$P_{dyn} = P_{dyn\_m} + P_{dyn\_a} + P_{data} + P_{ctrl} \tag{2}$$

$$P_{dyn\_m} \propto \sum^{T}(\sum^{N}\sum^{M}\sum^{K}\sum_{g_1}^{G_1} V_{dd}^2 \alpha_{g1}(a,b) C_{g1})/T \tag{3}$$

$$P_{dyn\_a} \propto \sum^{T}(\sum^{N}\sum^{M}\sum^{K}\sum_{g_2}^{G_2} V_{dd}^2 \alpha_{g2}(a,b) C_{g2})/T \tag{4}$$

Premises: Since zero values in input have a deterministic impact on power consumption, we can categorize the data patterns for multipliers and adders as below. For multipliers, 1) When both $a \neq 0$ and $b \neq 0$, the toggle rate will be high, $\mathcal{D}_g(a,b) = \mathcal{D}_{m\_high}$, resulting in high power consumption. 2) When either a or b equals zero, the toggle rate will be low, and $\mathcal{D}_g(a,b) = \mathcal{D}_{m\_low}$. For adders, similarly, 1) When both $a \neq 0$ and $b \neq 0$, the toggle rate will be high, $\mathcal{D}_g(a,b) = \mathcal{D}_{a\_high}$. 2) When either a or b equals zero, $\alpha_g(a,b)$ will follow the other distribution $\mathcal{D}_{a\_medium}$ with a medium toggle rate, and the power consumption will be medium. 3) When both a and b equal zero, the average toggle rate will be low, and the power consumption will be low, $\mathcal{D}_g(a,b) = \mathcal{D}_{a\_low}$.

Applying the law of large numbers in statistics, if the $T \times N \times M \times K$ is sufficiently large, the average toggle rates of all combinational gates with the same structure in the PE array will converge to the expectations of the distribution. Therefore, the total dynamic power consumption of multipliers and adders in PE array with different input situations can be approximated as constants. For multipliers, we can statistically record two different input situations: $a, b \neq 0$ and any one of a or b is equal to zero, denoted as $m11$ and $m01$, respectively. Similarly, for adders in the PE array, we can record the following three input situations: $a, b \neq 0$, any one of a or b is equal to zero, and both a and b are zero, denoted as $a11$, $a10$, and $a00$, respectively. Meanwhile, for dynamic power of data transformation P_{data}, the Hamming distance between two consecutive data accessed by register and memory is the switch activities generated by these two data. For non-zero data, every bit keeps changing, and their hamming distance can be approximated as a constant. The zero data is still the main component affecting the switch activities in memory and register access. Hence, we can apply the sparsity of weight and feature to reflect the power of data transformation.

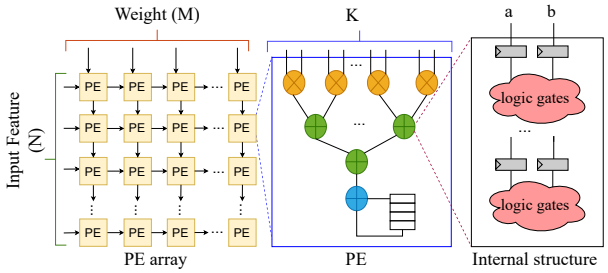

Fig. 5: The hierarchical structure of a typical PE array.

pattern	Description
α_{m11}	Both $a \neq 0$ & $b \neq 0$ in multiplier
α_{m01}	Any one of $a = 0$ or $b = 0$ in multiplier
α_{a11}	Both $a \neq 0$ & $b \neq 0$ in adder
α_{a01}	Any one of $a = 0$ or $b = 0$ in adder
α_{a00}	Both one of $a = 0$ & $b = 0$ in adder
β_w	sparsity of weight data
β_f	sparsity of feature data

TABLE III: Description of Data patterns in PROPHET.

In this way, we can formulate the data-driven power model as Eq. 5, where the learnable power model parameter I is the average dynamic power corresponding to each input data situation, α is the ratio of different input data patterns and β is the sparsity under the time window and can sample during memory access. We can construct the power model based on these data patterns and employ a regression model to fit these parameters in Eq. 5. The data patterns are summarized in Tab. III. There are only 7 data patterns for DNN accelerators compared to the RTL-proxies-based OPM, leading to the low area and power overhead.

$$\begin{aligned} P_{dyn} = P_{ctrl} + \alpha_{m11} I_{m11} + \alpha_{m01} I_{m01} + \alpha_{a11} I_{a11} \\ + \alpha_{a01} I_{a01} + \alpha_{a00} I_{a00} + \beta_w I_w + \beta_f I_f \end{aligned} \tag{5}$$

C. Data Pattern Extraction and Sampling

We have previously discussed data patterns related to zeros in the input data of adders/multipliers a and b. In this subsection, we will introduce how to extract and sample our predefined data patterns. In our conference paper [23], we only considered zero data in the feature map and assumed

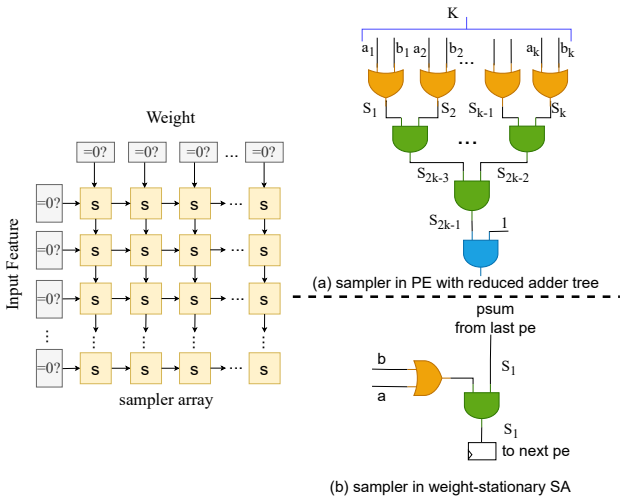

Fig. 6: Equivalent logic tree for data patterns sampling. The data transformation between samplers in the sampler array should be the same as PE array.

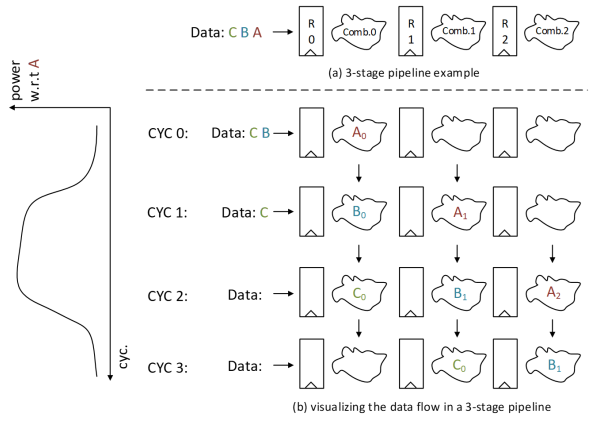

Fig. 7: An example of pipeline structures and their runtime power. Data will continue to affect power consumption within 3 cycles in this 3-stage pipeline structure.

that all weight data are non-zero. Under this assumption, PROPHET sampled data patterns only from the feature data. Then a sliding window was designed to account for the continuous effect of feature data on power consumption due to the inter-pipeline structure of the PE. However, sparse DNN models obtained through pruning have also gained popularity in mobile devices and embedded systems due to their higher energy efficiency during inference. Therefore, it is valuable to consider both weight and feature data.

Since the temporal resolution of PROPHET should be several clock cycles, we need to ensure that the data obtained by samplers is the same as that in the PE array, and the sampler should not affect the operation of the PE array. To achieve this, we implement PROPHET on the data input ports of the PE array. Meanwhile, the size and data flow of the sampler array should be the same as the PE array. Thus, we need a $N \times M$ sampler array for $N \times M$ PE array. For power prediction, we can insert FIFOs between the PE array and data buffers to allow PROPHET to sample input data in advance.

As shown in Figure 6, an $N \times M$ sampler array is implemented based on the data flow of the PE array. Each sampler corresponds to a PE in the DNN accelerator for data pattern sampling. For each valid input data, comparators are used to convert the multi-bit data into a 1-bit zero mask signal based on whether the data is zero. And then, the 1-bit zero mask will move the neighboring PE depending on the data flow of PE array. Thus, the designed 2D sampler array can obtain the same data order as the PE array enabling fine-grained power trace prediction. Otherwise, it will cause large error if the sampler array samples data patterns from mismatched feature and weight pairs. Within the sampler, an equivalent logic tree consisting of OR gates and AND gates is applied to obtain the data patterns. For example, by combining signals a_1 and b_1, if $a_1 == 0$ and $b_1 == 0$, we can identify the data pattern m_{11}. Similarly, if $s_1 == 0$ and $s_2 == 0$, the data pattern counter for a_{11} should be incremented. The other data patterns are collected in a similar manner by monitoring the input signals of the equivalent logic gates. These data patterns are then accumulated into counters for the power model.

For DNN accelerators with PEs consisting of multiple multipliers and adders, the sampler also includes multiple OR gates and AND gates. On the other hand, the structure of the sampler in weight-stationary systolic arrays is much simpler. Since the sampler only consists of 1-bit signals, flip-flops, and simple logic gates, the overhead of PROPHET is low.

Another important factor to consider is the pipeline length, which includes the pipeline within a PE's arithmetic unit and the pipeline between PEs in the PE array. As long as valid input data remains in the pipeline structure, it will continue to affect the total power consumption. Figure 7 provides an example of data in a 3-stage pipeline structure, where each data point affects the power consumption for 3 cycles. When the pipeline starts working, during the first two clock cycles, the power consumption may not reach its peak as only a portion of the stages are activated. Only when all 3 pipeline structures are activated does the power consumption increase to its highest point. Thus, we must consider the influence of the pipeline structure in the PE array. To achieve fine-grained temporal resolution and accurate OPM construction, a sliding window capable of recording all data patterns during a certain period is necessary. Since the sampler array can replicate the data transformation pipeline in the PE array, we only need to consider the pipeline within a PE's arithmetic unit. The size of the sliding window should match the length of the pipeline structure in the PE. At the same time, we accumulate and average each recorded data pattern within the sliding windows. The required temporal resolution of the on-chip power meter R is crucial and varies depending on different power management techniques. Therefore, we consider the temporal resolution R as the stride of the sliding window, which can be divided into two situations. When $R > L$, where L represents the pipeline length, there is no need to slide, and the sliding window size N_{sw} should be 1. On the other hand, in fine-grained scenarios where $R < L$, the data patterns in the sliding window should be updated every R clock cycles.

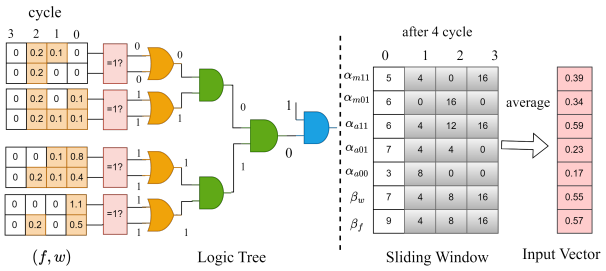

Fig. 8: the example of PROPHET extract data patterns from input data. Assuming that there is 1 PE which pipeline length is 16 cycles and the temporal resolution is 4 clock cycles.

Thus, the size of the sliding window is calculated as shown in Eq. 6.

$$N_{sw} = \left\lfloor \frac{L}{R} \right\rfloor \tag{6}$$

D. PROPHET Model Training

PROPHET training is off-chip, similar to most on-chip power meters. The ground-truth labels are obtained through power simulation. The input vector for linear regression is constructed based on the input weights and feature data of power simulations. Assuming a power meter with a temporal resolution of R, the following progress is made:

Firstly, per-cycle data patterns are extracted from the input feature and weight sequences. These patterns are then accumulated within R clock cycles. Subsequently, the input vectors for PROPHET's model training and inference are obtained by averaging the data patterns in the sliding window. The depth of the sliding window is determined by the pipeline length (L), while the step is determined by the temporal resolution (R).

To illustrate the sampling of pre-defined data patterns, Fig.8 is provided. Assuming $L = 16$ and $R = 4$, the signals in the logic tree continue to change as the feature and weight data are input. By combining the input signals of each OR and AND gate, the data patterns of this PE can be derived. As depicted in Fig.8, when the first input feature and weight pair is $(0.1, 0.1)$, the two input signals of the OR gate are $(1, 1)$. Consequently, we can record 1 data pattern for α_{m11}. These values are collected from all samplers in the array to obtain the total data patterns in that clock cycle. After 4 clock cycles, the newly recorded patterns are placed into the sliding window. Ultimately, the input vector for PROPHET is computed as the average of these 4 sub-windows. Once the dataset is constructed, linear regression is applied to train the parameters in PROPHET.

E. Hardware Implementation and Power Prediction

Figure 2 illustrates the general architecture of DNN accelerators, which includes multiple levels of memory. The L1 buffer is typically designed with a large capacity to store uniform data. The data reshape module converts the operations in convolution layers and fully connected layers into the general matrix multiplication format. The weight and feature data are stored in the weight buffer (*wbuf*) and input feature buffer (*ifbuf*), respectively, for PE array computing. To accurately trace power consumption at a fine-grained level, the sampling of data patterns must maintain the same order of input data flow as in the PE array. Therefore, when considering both weight and feature data, PROPHET should sample both simultaneously to ensure that the sampler array receives the same input data as the PE array.

However, the movement of input feature and weight data from the L1 buffer to *ifbuf* and *wbuf*, is usually independent due to different reshape operations and limited bandwidth. To address this, we implement our power model between the PE array, *wbuf*, and *ifbuf* to ensure that it captures the same data as the PE array. The FIFOs before the PE array are optional and depend on whether power prediction is required. For runtime power prediction, these FIFOs allow PROPHET to sample data before the PE array by generating a time interval between data readouts from buffers and PE array executions. The maximum number of clock cycles that PROPHET can predict in advance is determined by the depth of these FIFOs. Since the prediction requirement is usually not more than 100 clock cycles in most power management scenarios, the overhead is negligible compared to the weight and feature buffers, which have sizes in the hundreds of Kbs.

The hardware implementation of PROPHET show as Figure 9. First, for each valid input data, comparators convert the data into a single-bit mask based on whether the data is zero. These zero masks are then fed into the sampler array. The dataflow transformation in the sampler array mirrors that of the PE array. For example, in systolic array-based DNN accelerators, the input feature data flows along the column dimension. Similarly, the feature zero masks should also flow along the column dimension in the sampler array to ensure consistent data flow for sampling.Second, every clock cycle, the data patterns in the sampler array are collected and accumulated into data pattern registers. Once the temporal resolution of R clock cycles is reached, the sliding windows update and average all sub-windows to obtain the input vector for the pre-trained power model.Finally, multipliers and adders will calculate the predicted dynamic power for the PE array, with the trained parameters I. Since the input zero mask is a single-bit signal and the sampler array only consists of several logic gates, the overhead of PROPHET is very low.

V. Experiment and Discussion

A. Experiment Setup

The simulation platform for our experiments follows the architecture shown in Figure 2. We have implemented the L1 buffer, data reshape modules, weight buffers, and input feature buffer in software to allow for flexible simulation. The remaining modules have been synthesized, placed, and routed using the TSMC 40nm process. To obtain accurate ground-truth power waveform, we ran the post-layout power simulation in Synopsys PTPX [46].

In our experiments involving DNN accelerators, we have implemented PE arrays with two commonly used MACC

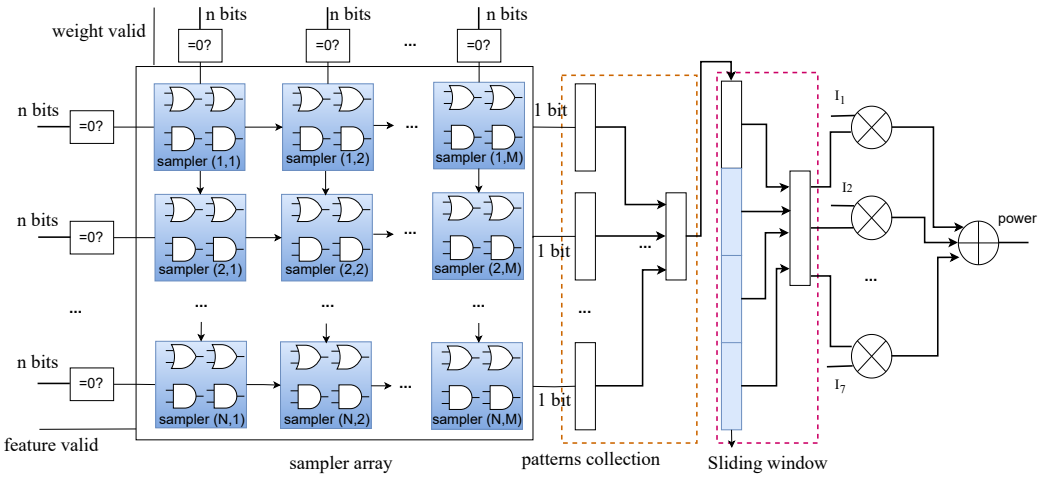

Fig. 9: Equivalent logic tree for data patterns sampling. The data transformation between samplers in the sampler array should be the same as PE array.

Parameter	OS-FP16	OS-INT16	WS-FP16	WS-INT16
M	4	4	16	16
N	4	4	16	16
K	16	16	1	1
L	8	4	3	3
$area(\text{mm}^2)$	0.57	0.46	0.49	0.53
$freq(\text{MHz})$	250	333	250	333

TABLE IV: Hardware Architecture Configurations

structures as depicted in Figure 3. Additionally, we have considered two popular data types in DNN accelerators: 16-bit floating point (FP16) and 16-bit integer (INT16). The 16-bit floating point data type is suitable for training and high-precision inference scenarios, while the integer data type is commonly employed for energy-efficient inference in embedded systems.

Regarding the architecture of DNN accelerators, we have chosen to implement the $4 \times 4 \times 16$ output-stationary systolic array, similar to the matrix unit CUBE in Ascend [39]. We have also implemented the 16×16 weight-stationary systolic array, akin to the architecture of Google TPU [3]. The details of our experiments are summarized in Tab. IV. All of these PE arrays consist of 256 multipliers and adders, which is comparable to the size of most DNN accelerators in embedded systems. The pipeline length of the PE in the output-stationary systolic array is deeper than that in the weight-stationary systolic array due to its longer data path. At the same time, the integer data type can achieve a higher operating frequency compared to the FP16 data type.

In power simulation, it is crucial to cover the range of dynamic power. According to the definitions of data patterns and previous works, zero data has a strong connection with the dynamic power consumption of the PE array. Therefore, we employ two types of data as stimuli for power simulation:

- **Feature and weight data from real-world DNN models:** We selected the pre-trained VGG19 and ResNet50 models from the PyTorch model hub as benchmarks for our study. The dataset construction involved two steps. Firstly, we partitioned the original input tensor into small data blocks based on the convolution dataflow. Then, we randomly chose several data blocks for power simulation. The number of selected data blocks was limited by the simulation clock cycles, approximately 10,000 cycles. For VGG19, the selected layer indexes were 2, 5, 10, 14, and 16. For ResNet50, the selected indexes were 1, 2, 5, 11, and 12. Since the weight sparsity in pruned DNN models is typically greater than 50% [47] leading to biased dataset, we randomly mask the weight data as zero to achieve sparsity levels ranging from 0 to 1. The total number of simulation clock cycles for this type of stimulus is around 100,000, which is comparable to the generated random data. Meanwhile, we also evaluate the power fluctuation of SA when executing the pre-trained BERT transformer. Because the data sparsity of matrix multiplication in BERT is almost 0, the power variation is flat leading to a biased dataset. Hence, we don't take BERT as the dataset and benchmark.
- **Generated random data**: In order to improve the generality of the training dataset, we traverse the sparsity of all weight and feature data in steps of 10%. As a result, we obtained 100 groups of simulation data, with each group supporting simulation for 1024 clock cycles. For the floating-point data type, the data range is $[0, 1]$. For the 16-bit integer data type, the data range covers all the bits, ranging from $[-32768, 32767]$.

We evaluate the accuracy of the proposed data-pattern-based power model in four different DNN accelerators, as shown in Table IV. The per-cycle pattern trace is generated by analyzing the input feature data based on the defined data patterns. Power and data patterns with different temporal resolutions and sliding window sizes are then constructed by averaging the per-cycle waveform during a time window for experiments with different resolutions. We evaluate our power model in terms of accuracy and the overhead in power and area aspects.

For accuracy evaluation, we utilize two metrics: the normalized mean absolute error (NMAE) and the coefficient of determination R^2. The NMAE, as defined in Eq.7, is an important metric that reflects the error of power models.

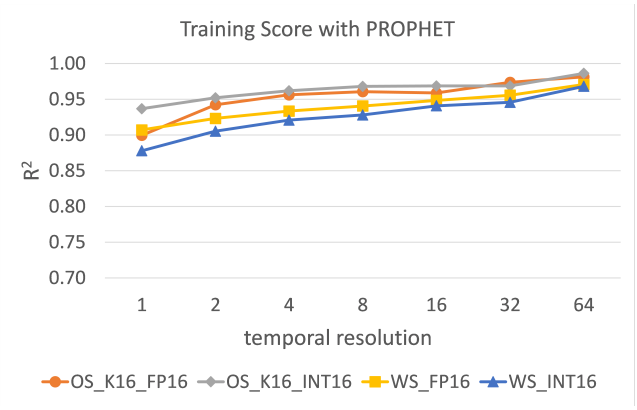

Fig. 10: The training score R^2 in four hardware platforms with PROPHET in linear regression.

In the equation, y_i represents the ground-truth power, and $\hat{y}_i$ represents the predicted power by power models. The coefficient of determination R^2, as defined in Eq.8, is a widely used metric in regression analysis. In this equation, $\overline{y}_i$ represents the average value of the ground-truth power from power simulation. A higher R^2 value indicates a better fit of the regression model to the data.

$$\text{NMAE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{7}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2} \tag{8}$$

B. Model Training Score

During model training phase, the training score, represented by the coefficient of determination R^2, is a crucial metric that indicates the quality of the trained model. A higher training score signifies better model quality. For model training, we randomly select half of the data from both the generated dataset and the real-world application dataset. The remaining half of the data is used for verification. Figure 10 displays the training scores of four hardware platforms during our model training phase. The training scores R^2 at a per-cycle temporal resolution can exceed 0.85, and they reach 0.9 when temporal resolution is larger than 2 clock cycles. Moreover, as the temporal resolution increases, the training scores surpass 0.95. These results demonstrate that our proposed data patterns have a strong correlation with the dynamic power of the PE array, making them suitable proxies for constructing an accurate power model for DNN accelerators.

C. PROPHET Performance Evaluation

We apply the NMAE in the verified dataset to reflect the accuracy of our data-pattern-based power model. We compare the NMAE of PROPHET with four baselines: (1) SPA: taking only the sparsity of weight and input feature map as data patterns and applying linear regression to construct power model. The sparsity of input feature is a popular pattern to predict the workload of DNN accelerator [8] and also can be sampled during memory access. (2) Lasso-regression-based OPM predicts dynamic power 64 clock cycles in advance. The lasso regression is the most widely-used sparsity-inducing method, for proxy selection and model construction, to construct the low-overhead OPM in work of [25]. We take all the RTL signals in behavior simulations as input for lasso regression and train a power model. Then, in the verified dataset, we apply this trained power model to predict the power of 64 clock cycles in advance. (3) MCP-regression-based OPM: the MCP technique adopted in APOLLO [16] and DEEP [22] is considered one of the most advanced methods for selecting the minimum proxies from the RTL signals to construct accurate, per-cycle, and low-cost OPM. For a fair comparison, we also take all the RTL signals as input and train the power model with MCP regression. Then, we obtain the NMAE when the MCP-based OPM predicts the dynamic power 64 clock cycles in advance in the verified dataset. (4) PE1 $\times$ 1: training the power model with our proposed data patterns, but the hardware platforms scale into only 1 PE. In this way, the size of a single PE is the $1/16$ of output-stationary SAs and $1/256$ of weight-stationary SAs to verify the premises we mentioned in section IV-B.

Figures 11 and 12 depict the comparisons between PROPHET and the baselines on four experimental hardware platforms. PROPHET demonstrates accurate dynamic power prediction with high temporal resolution. For both integer and floating-point data types, PROPHET achieves an NMAE below 7%, even when considering per-cycle temporal resolution requirements. As the temporal resolution increases, the error decreases to less than 5%.

In contrast, power models based on sparsity perform poorly. The error remains consistently above 15% for output-stationary SAs with integer data type and weight-stationary SAs, even with increasing temporal resolution. The error of the FP16 output-stationary SA in baseline SPA is slightly lower compared to other platforms due to its deeper pipeline structure, resulting in a higher proportion of internally powered components unaffected by data. However, the error of SPA does not decrease with increasing temporal resolution. Even with a temporal resolution of 64 clock cycles, the NMAE remains above 15%.

Furthermore, two RTL-signal-based power models, trained using classical and novel regression methods, exhibit significant errors ($> 30\%$) when predicting dynamic power. This inaccuracy arises because the switching activities of selected signals rely only on current and past data, lacking consideration of future input data. However, a slight improvement in error can be observed as the temporal resolution increases. The possible reason is there are some overlapped data in the convolution data flow between consecutive convolution kernels. These overlapped data will lead to similarities in power between two consecutive periods.

When evaluating PROPHET's accuracy in a systolic array with only 1 PE, the error increases to 20% compared to the original SAs in pre-cycle temporal resolution for output-stationary SAs. However, as the temporal resolution increases, the error decreases. For weight-stationary SAs, the error ex-

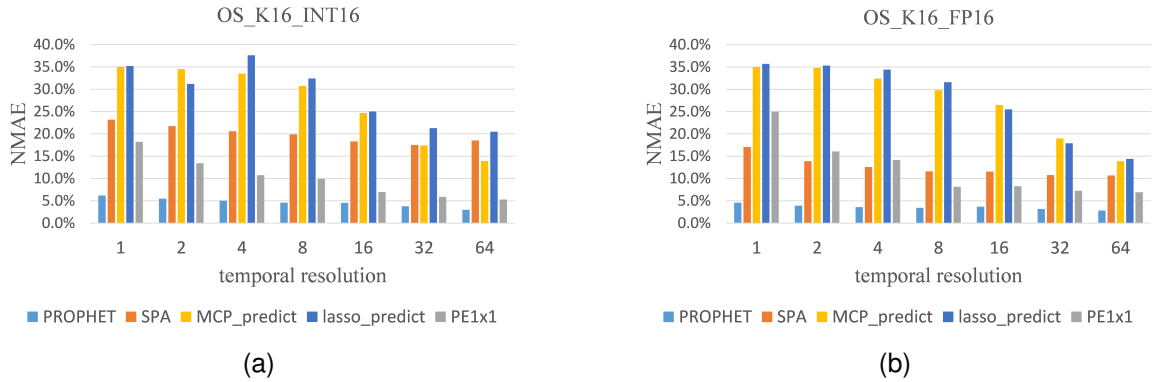

Fig. 11: The NMAE of trained power model in the verified dataset for two output-stationary SAs. (a) Output-stationary SA with $K == 16$ and 16-bit integer data types.(b) Output-stationary SA with $K == 16$ and 16-bit floating point data type.

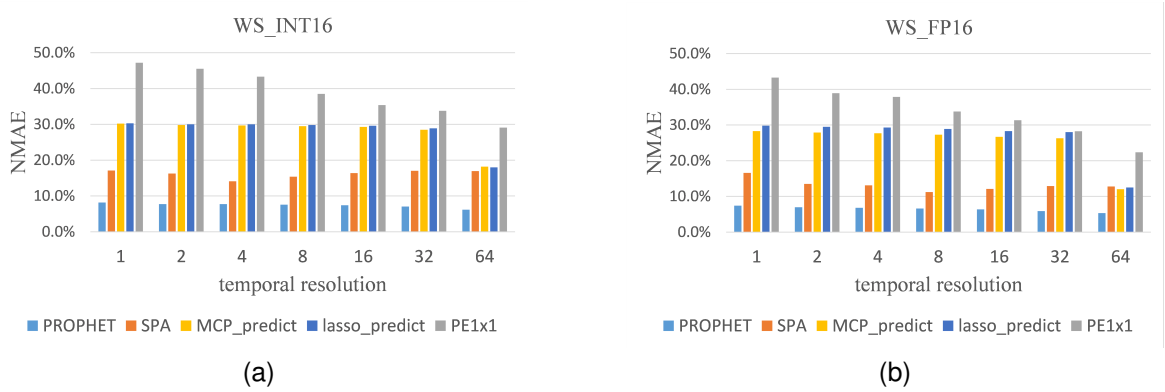

Fig. 12: The NMAE of trained power model in verify dataset for two weight-stationary SAs. (a) weight-stationary SA with 16-bit integer data type. (b) weight-stationary SA with 16-bit floating point data type.

ceeds 40% due to their PEs consisting of only 1 multiplier and adders, which are much smaller compared to the PEs of output-stationary SAs. These findings confirm that as the size of the $T \times N \times M \times K$ operation increases, the average toggle rates of all multipliers and adders in the proposed data patterns tend to become constants, resulting in low errors during power model construction.

The area and power overhead are crucial factors in the implementation of OPM. It is essential to minimize the area and power of OPMs to ensure minimal impact on the overall system. In our experiments, we implemented a 4×4 sampler array for output-stationary SAs and a 16×16 sampler array for weight-stationary SAs. To handle feature and weight sparsity sampling, we incorporated two counters on the *ifbuf* and *wbuf* respectively, in the SPA. The implementation of MCP-regression-based OPM and lasso-regression-based OPM resembles that of APOLLO [16], providing pre-cycle temporal resolution. Consequently, to estimate power consumption, multipliers are replaced with AND gates to reduce overhead. The number of proxies in lasso regression is 1097, 2026, 2239, and 1768 for *OS_INT16*, *OS_FP16*, *WS_INT16*, and *WS_FP16*, respectively. In MCP regression, proxies are selected in a more compressed and accurate manner, with 157, 208, 236, and 149 proxies chosen for the four hardware platforms. This approach achieves similar accuracy compared to lasso regression in power model construction. To ensure a fair comparison, all trained parameters in OPMs are 16-bit integers. Table V and Table VI present the area and power overhead of PROPHET and other OPMs. The sparsity-based OPM exhibits minimal overhead due to the ease of counting feature and weight sparsity. Additionally, the advanced MCP-based OPM incurs only 1-2% overhead, which is one-tenth of that incurred by lasso-based OPMs. In the case of our proposed PROPHET, the area and power overhead account for approximately 2-4.5% of the four different systolic array designs, even with the inclusion of a sampler array for data pattern extraction. While the sparsity-based and MCP-based OPMs can achieve lower overhead than PROPHET, they lack accuracy in predicting dynamic power consumption for DNN accelerators.

Additionally, we evaluated the timing impact of PROPHET. The experiments revealed that PROPHET does not impact the critical path of the original SA designs. In the case of samplers, the logic tree consists of only 1-bit AND and OR gates. Moreover, we strategically placed registers between the samplers and power calculation units to minimize the data path length of PROPHET.

D. Reliability Analysis of Proposed Data Patterns

We employ the most advanced technique, MCP regression, for proxy selection in RTL-based OPM construction to assess the reliability of our defined data patterns. MCP regression

	OS-FP16	OS-INT16	WS-FP16	WS-int16
lasso	21.0	32.0	38.0	32.1
MCP	3.1	3.3	4.0	2.7
SPA	1.9	1.5	1.6	1.7
PROPHET	3.8	2.9	4.0	3.8

TABLE V: Area Overhead Comparison(unit:%)

	OS-FP16	OS-INT16	WS-FP16	WS-int16
lasso	12.5	24.1	26.9	19.3
MCP	1.8	2.5	1.9	1.6
SPA	0.4	0.3	0.3	0.4
PROPHET	3.5	2.6	4.5	2.7

TABLE VI: Power Overhead Comparison(unit:%)

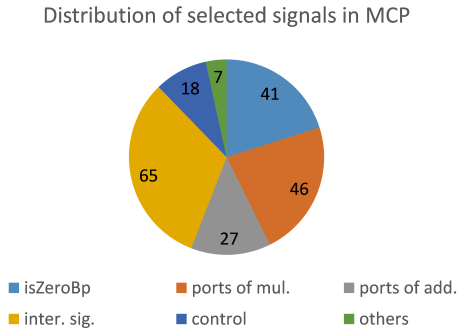

Fig. 13: Selected Proxies by MCP regression in output-stationary SA with floating point data type.

utilizes a strong and relaxed penalty term during signal selection. We compare the selected signals in MCP regression with our proposed data patterns to evaluate their reliability.

Figure 13 illustrates the distribution of selected RTL signals using MCP regression on the hardware platform of FP16 output-stationary SA. Approximately 20% of the proxies is the *isZeroBp* signal, which represents the zero bypassing mechanism in multipliers. When either input data a or b is zero, *isZeroBp* is set to 1, bypassing the multiplication and resulting in an output data c of zero. These *isZeroBp* signals are equivalent to our data patterns m_{01} and m_{11}. Moreover, we observe that over 60% of the selected proxies using MCP regression are the ports of multipliers and adders, indicating a strong connection between the input data of these components and the dynamic power of DNN accelerators. Similarly, our proposed data patterns also reflect the dynamic power by examining the input data ports. Additionally, approximately 30% of the proxies identified by MCP regression correspond to the internal signals of multipliers and adders. This explains why MCP-based OPMs achieve lower errors when estimating runtime power.

The experimental results demonstrate that our proposed data patterns in PROPHET are reasonable and similar to the MCP proxies. The advantage of PROPHET lies in the ease of obtaining our proposed data patterns during memory access, as opposed to monitoring the switching activities of selected signals required by MCP regression.

VI. Case Study

A. Power Peak Mitigation with PROPHET Prediction

The motivation behind PROPHET is to detect and address dangerous power peaks that can lead to risky dynamic voltage

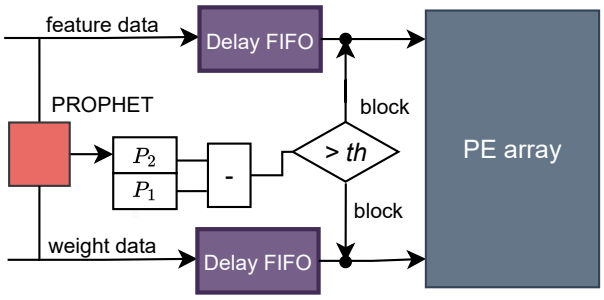

Fig. 14: Diagram of PROPHET for power peak mitigation.

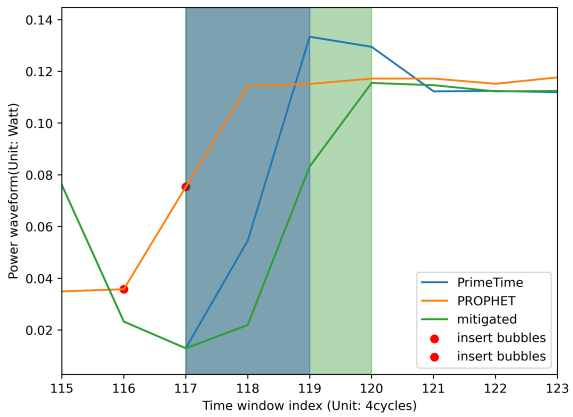

Fig. 15: PROPHET predicts power waveform 4 cycles in advance and mitigates the power surge.

drops. This dynamic voltage drop can be simplified as LdI/dt, where L represents the equivalent inductor of the power delivery network, and dI/dt denotes the rate of current change over a specific period.

In Figure 14, we present an example showcasing the effectiveness of PROPHET in mitigating power peaks. Our experiments have revealed that power peaks often occur during the startup phase and in data blocks located at the edges of input feature maps due to zero padding. To allow sufficient time for actuators to respond, we have incorporated two 4-clock-cycle delay FIFOs, enabling PROPHET to predict power consumption four clock cycles in advance. In this setup, P_2 represents the anticipated power consumption of the PE array. When the power change trend (ΔP) exceeds a certain threshold (th), the actuator blocks the input data for two clock cycles, thereby reducing the rate of change in current (dI/dt). Further details on power peak mitigation during the execution of VGG layer 2 with output stationary INT16 SA are provided in Figure 15. Notably, at $X = 117$ and 118, a significant power surge is observed, and the actuator inserts two clock cycle "bubbles" to mitigate this surge. With the predictive capabilities of PROPHET, the actuator has ample time to respond. Moreover, by introducing a mere 4-clock-cycle delay, the rate of change in current (dI/dt) is reduced by 36% during this power surge. These results demonstrate PROPHET's significant potential in power peak mitigation scenarios. However, the power fluctuations in real-world DNN accelerators are more complex, and we plan to conduct a comprehensive exploration of this topic in our future work.

VII. Conclusion

Power consumption is a crucial consideration in the design of DNN accelerators. In this study, we introduced PROPHET, a fine-grained predictive on-chip power meter for DNN accelerators that can predict the power waveform dozens of clock cycles in advance. We proposed data patterns that accurately reflect the dynamic power of DNN accelerators. These patterns can be sampled during memory access between buffers and the PE array within a low overhead. Additionally, we design a sliding window to record data patterns in fine-grained temporal resolution scenarios, considering the pipeline in the PE. For the PROPHET hardware prototype, we implemented an equivalent logic sampler array that captures the data flow of the PE array to sample these patterns. Since PROPHET utilizes only seven data patterns and the sampler consists of 1-bit AND gates, OR gates, and Flip-Flops, the overhead of PROPHET is minimal.

In our experiments, we first presented the results of training our data-pattern-based power model for four PE arrays with different dataflows and data types. PROPHET achieved an $R^2 > 0.85$ in pre-cycle temporal resolution for all hardware platforms, and the R^2 value exceeded 0.9 as the temporal resolution became coarser. Furthermore, we compared PROPHET with classical data patterns in DNN workload prediction and two advanced RTL-signal-based OPMs. PROPHET achieved an NMAE of less than 7% in the verified dataset with an area and power overhead of less than 5%. In contrast, sparsity-based OPMs, commonly used in workload prediction, exhibited inaccuracies and yielded errors of approximately 15-25% in fine-grained scenarios. RTL-signal-based OPMs did not provide complete runtime power prediction.

We also analyzed the reliability of our proposed data patterns by comparing PROPHET in a small single PE scenario with proxies selected through MCP regression. When only one PE was considered, PROPHET exhibited errors greater than 25%. However, as the value of $T \times N \times M \times K$ increased, the error decreased, substantiating our premises in section IV-B. Furthermore, over 60% of the selected proxies in MCP regression demonstrated a strong connection with our proposed patterns.

In summary, our data-pattern-based OPM for DNN accelerators can predict dynamic power dozens of clock cycles in advance and achieve an error of less than 7% with a maximum overhead of 4.5%, when the temporal resolution is greater than 2 clock cycles. Moreover, the experiments confirmed the reliability of our proposed data patterns, which can be applied to other DNN accelerators with regular PE arrays and multiply-accumulate structures.

VIII. Future Work

In state-of-the-art transformer accelerators, the softmax operation exhibits higher power dissipation compared to ReLU and pooling in CNN. In this recent work [48], the softmax operation accounts for approximately 15% of the dynamic power consumption in transformer accelerators. As the softmax operation is an independent operator, our future work will focus on exploring a power meter specifically for the softmax unit.

Furthermore, the power consumption of DNN accelerators is influenced by system-level events, such as start working and data reuse configurations. There is still ample room for exploration in terms of leveraging PROPHET for energy-efficient power management in a complete DNN accelerator. In our future work, we plan to conduct a comprehensive exploration and evaluation in this area.

References

[1] T. Chen *et al.*, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 269–284, 2014.

[2] D. Liu *et al.*, "Pudiannao: A polyvalent machine learning accelerator," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1, pp. 369–381, 2015.

[3] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*, 2017, pp. 1–12.

[4] T. Norrie *et al.*, "The design process for google's training chips: Tpuv2 and tpuv3," *IEEE Micro*, vol. 41, no. 2, pp. 56–63, 2021.

[5] H. Liao *et al.*, "Davinci: A scalable architecture for neural network computing." in *Hot Chips Symposium*, 2019, pp. 1–44.

[6] NVIDIA. Nvidia® deep learning accelerator. [Online]. Available: http://nvdla.org/primer.html

[7] Q. Sun *et al.*, "Power-driven dnn dataflow optimization on fpga," in *ICCAD*, 2019, pp. 1–7.

[8] S. Liu *et al.*, "Dynamic voltage and frequency scaling to improve energy-efficiency of hardware accelerators," in *HiPC*. IEEE, 2021.

[9] C. Gilberto *et al.*, "Power prediction for intel xscale processors using performance monitoring unit events power prediction for intel xscale processors using performance monitoring unit events," in *ISLPED*, 2005.

[10] F. Oboril *et al.*, "High-resolution online power monitoring for modern microprocessors," in *DATE*. IEEE, 2015.

[11] M. Sagi *et al.*, "A lightweight nonlinear methodology to accurately model multicore processor power," *TCAD*, 2020.

[12] R. Bertran *et al.*, "Decomposable and responsive power models for multicore processors using performance counters," in *ICS*, 2010, pp. 147–158.

[13] Y. Zhang *et al.*, "On-the-fly power-aware rendering," in *Computer Graphics Forum*. Wiley Online Library, 2018.

[14] M. S. Gupta *et al.*, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *DATE*. IEEE, 2007.

[15] X. Zhang *et al.*, "Characterizing and evaluating voltage noise in multi-core near-threshold processors," in *ISLPED*. IEEE, 2013, pp. 82–87.

[16] Z. Xie *et al.*, "APOLLO: An automated power modeling framework for runtime power introspection in high-volume commercial microprocessors," in *MICRO*, 2021, pp. 1–14.

[17] S. Das *et al.*, "Modeling and characterization of the system-level power delivery network for a dual-core arm cortex-a57 cluster in 28nm cmos," in *ISLPED*. IEEE, 2015, pp. 146–151.

[18] F. Ye *et al.*, "On-chip droop-induced circuit delay prediction based on support-vector machines," *TCAD*, vol. 35, no. 4, pp. 665–678, 2015.

[19] J. Leng *et al.*, "GPU voltage noise: Characterization and hierarchical smoothing of spatial and temporal voltage noise interference in gpu architectures," in *HPCA*. IEEE, 2015.

[20] D. Zoni *et al.*, "Powerprobe: Run-time power modeling through automatic rtl instrumentation," in *DATE*. IEEE, 2018.

[21] D. Kim *et al.*, "Simmani: Runtime power modeling for arbitrary rtl with automatic signal selection," in *MICRO*, 2019.

[22] Z. Xie *et al.*, "DEEP: Developing extremely efficient runtime on-chip power meters," in *ICCAD*, 2022.

[23] J. Peng *et al.*, "PROPHET: Predictive on-chip power meter in hardware accelerator for dnn," in *DAC*, 2023, pp. 1–6.

[24] M. Najem *et al.*, "A design-time method for building cost-effective run-time power monitoring," *TCAD*, 2016.

[25] D. J. Pagliari *et al.*, "All-digital embedded meters for on-line power estimation," in *DATE*. IEEE, 2018.

[26] Z. Lin *et al.*, "An ensemble learning approach for in-situ monitoring of fpga dynamic power," *TCAD*, vol. 38, no. 9, pp. 1661–1674, 2018.

[27] D. Gizopoulos *et al.*, "Modern hardware margins: Cpus, gpus, fpgas recent system-level studies," in *IOLTS*. IEEE, 2019, pp. 129–134.

[28] V. J. Reddi *et al.*, "Voltage emergency prediction: Using signatures to reduce operating margins," in *HPCA*. IEEE, 2009.

[29] G. Papadimitriou *et al.*, "Harnessing voltage margins for energy efficiency in multicore cpus," in *MICRO*, 2017.

[30] J. Leng *et al.*, "Gpuvolt: Modeling and characterizing voltage noise in gpu architectures," in *ISLPED*, 2014, pp. 141–146.

[31] V. K. Kalyanam *et al.*, "A proactive voltage-droop-mitigation system in a 7nm hexagon™ processor," in *VLSI*. IEEE, 2020.

[32] A. Parashar *et al.*, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 45, no. 2, pp. 27–40, 2017.

[33] S. Zhang *et al.*, "Cambricon-x: An accelerator for sparse neural networks," in *MICRO*. IEEE, 2016, pp. 1–12.

[34] J. Zhang *et al.*, "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *DAC*, 2018, pp. 1–6.

[35] N. Chandramoorthy *et al.*, "Resilient low voltage accelerators for high energy efficiency," in *HPCA*. IEEE, 2019, pp. 147–158.

[36] N. D. Gundi *et al.*, "Strive: Enabling choke point detection and timing error resilience in a low-power tensor processing unit," in *DAC*. IEEE, 2023, pp. 1–6.

[37] T. Jia *et al.*, "A dynamic timing enhanced dnn accelerator with compute-adaptive elastic clock chain technique," *JSSC*, vol. 56, no. 1, pp. 55–65, 2020.

[38] Y. Chen *et al.*, "Diannao family: energy-efficient hardware accelerators for machine learning," *Communications of the ACM*, vol. 59, no. 11, pp. 105–112, 2016.

[39] H. Liao *et al.*, "Ascend: a scalable and unified architecture for ubiquitous deep neural network computing: Industry track paper," in *HPCA*. IEEE, 2021.

[40] M. Pricopi *et al.*, "Power-performance modeling on asymmetric multicores," in *CASES*. IEEE, 2013, pp. 1–10.

[41] M. J. Walker *et al.*, "Accurate and stable run-time power modeling for mobile and embedded cpus," *TCAD*, vol. 36, no. 1, pp. 106–119, 2016.

[42] M. Najem *et al.*, "Method for dynamic power monitoring on fpgas," in *FPL*. IEEE, 2014.

[43] H. J. Veendrick, "Short-circuit dissipation of static cmos circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 4, pp. 468–473, 1984.

[44] A. P. Chandrakasan *et al.*, "Low-power cmos digital design," *IEICE Transactions on Electronics*, vol. 75, no. 4, pp. 371–382, 1992.

[45] H. Sharma *et al.*, "From high-level deep neural models to fpgas," in *MICRO*. IEEE, 2016, pp. 1–12.

[46] synopsysPTPX. Synopsys® primetime px. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html

[47] S. Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, 2016.

[48] A. Marchisio *et al.*, "Swifttron: An efficient hardware accelerator for quantized transformers," in *IJCNN*. IEEE, 2023, pp. 1–9.

Jian Peng received his B.S. degree from University of Electronic Science and Technology of China, Chengdu, China, in 2019, and the MPhil degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, in 2022. He is currently a PhD student in the same department and university. His current research interests include power modeling and power management for DNN accelerator.

Tingyuan Liang received his B.S. degree in Electrical and Information Engineering from Zhejiang University in 2017. After completing his undergraduate studies, he pursued advanced degrees and earned his MPhil and PhD degrees in the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology in 2019 and 2023, respectively. His current research interests include EDA algorithms and computer architecture.

Jingbo Jiang received his B.S. degree from Hefei University of Technology, Hefei, China, in 2016, and his M.S. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2020. He is currently working towards his Ph.D. degree at Hong Kong University of Science and Technology, Hong Kong, China. His research interests are in circuit design for implementing deep learning algorithms.

Yipu Zhang received the B.S. degree from the School of Integrated Circuit Science and Engineering, University of Electronic and Science of China, Chengdu, China, in 2023. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong. His current research interests include hardware-software co-design and accelerator design.

Zhe Lin (Member, IEEE) received the B.S. degree from the School of Electronic Science and Engineering, Southeast University, Nanjing, China, in 2014, and the Ph.D. degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, in 2019. He is an Assistant Professor with the School of Integrated Circuits, Sun Yat-sen University, Shenzhen, China. Previously, he was an Associate Research Fellow with Peng Cheng Laboratory, Shenzhen. Dr. Lin was a recipient of the Best Paper Award Nominations at DATE 2022 and FCCM 2019.

Zhiyao Xie is an Assistant Professor of the Department of Electronic and Computer Engineering (ECE) at the Hong Kong University of Science and Technology (HKUST). Zhiyao received his Ph.D. degree from Duke University in 2022 and B.Eng. from City University of Hong Kong in 2017. His research interests include machine learning algorithms for EDA and VLSI design. He has received multiple prestigious awards, including the IEEE/ACM MICRO 2021 Best Paper Award, ACM SIGDA SRF Best Research Poster Award 2022, ASP-DAC 2023 Best Paper Award, ACM Outstanding Dissertation Award in EDA 2023, EDAA Outstanding Dissertation Award 2023, and the 2023 Early Career Award from Hong Kong Research Grants Council (RGC).

Wei Zhang received her PhD degree in Electrical Engineering from Princeton University with Wu Prize for research excellence. She joined Hong Kong University of Science and Technology in 2013 and established the Reconfigurable System Lab, and is currently a professor in the same department. She was an Assistant Professor at the School of Computer Engineering at Nanyang Technological University, Singapore, from 2010 to 2013. She has authored and co-authored more than 150 papers in peer-reviewed journals and international conferences, and won the best paper awards in ISVLSI 2009, ICCAD 2017 and ICCAD 2022. Her current research interests include reconfigurable system and FPGA-based design, electronic design automation, computer architecture, and embedded system security. Prof. Zhang has served and currently serves in many editorial boards as Associate Editor including ACM TRETS, IEEE TVLSI, IEEE TCAD, ACM TECS, and ACM JETC, IEEE ESL, IEEE TCASII, etc. She served as General Chair or Vice-Chair of ISVLSI 2018, FPT 2022, and ASAP 2024. She also serves on many organization committees and technical program committees, such as DAC, ICCAD, DATE, FPGA, etc.