PROPHET: Predictive On-Chip Power Meter in Hardware Accelerator for DNN

Jian Peng
Hong Kong University of Science and Technology
jpengai@connect.ust.hk

Tingyuan Liang
Hong Kong University of Science and Technology
tliang@connect.ust.hk

Zhiyao Xie
Hong Kong University of Science and Technology
eezhiyao@ust.hk

Wei Zhang
Hong Kong University of Science and Technology
wei.zhang@ust.hk

***Abstract*—On-chip power meters play a critical role in power management by generating timely and accurate power traces at runtime. However, both performance-counter-based and existing RTL-based on-chip power meters have difficulty in providing sufficient response time for fast power and voltage management scenarios. Additionally, they can be costly to implement for large-scale DNN accelerators with many homogeneous process elements. To address these limitations, this paper proposes PROPHET, a data-pattern-based predictive on-chip power meter targeting multiply-accumulate-based DNN accelerators. By sampling pre-defined data patterns during memory access, PROPHET can predict power consumption before it actually happens. In our experiments, PROPHET predicts power consumption dozens of clock cycles in advance, with a temporal resolution of 4 clock cycles and NMAE $< 7\%$ and area overhead $< 2\%$ for various systolic-array-based DNN accelerators. PROPHET has the potential to enable fine-grained power management and optimization for large-scale DNN accelerators, improving their energy efficiency.**

***Index Terms*—pattern-based, power prediction, on-chip power meter, DNN accelerator**

I. Introduction

Efficient power management is indispensable for attaining high energy efficiency and guaranteeing the system stability of modern hardware design. In power management, on-chip power meters (OPMs) play a critical role by providing accurate and timely power traces. However, the specific requirements for OPMs may vary, depending on the intended application. For example, dynamic voltage and frequency scaling (DVFS), which is managed by the system firmware and/or operating system (OS), only requires coarse-grained temporal resolution in power tracing. In contrast, techniques for fast power management, voltage boosting, and voltage drop mitigation require fine-grained temporal resolution and short response time. For example, dynamic LdI/dt voltage noise effects can develop within 20 nanoseconds in modern computing architectures [1].

Previous studies have presented power models using performance counters for coarse-grained power tracing, targeting DVFS and temperature-suitability management. Recently, some automatic frameworks have been proposed to construct RTL-based OPMs with low overhead and fine-grained temporal resolution. For instance, the PowerProbe proposed in [2] can achieve temporal resolution at the level of tens of cycles. Simmani [3] and APOLLO [1] can further achieve per-cycle resolution for microprocessors. However, these RTL-based OPMs can be costly to implement on DNN accelerators, as they will select proxies from all the homogeneous processing elements, some of which are repeated.

Moreover, even with per-cycle temporal resolution, OPMs may still not leave sufficient response time for timely power management in some scenarios. For example, for voltage emergency mitigation in single-core CPUs, unsuppressed voltage emergencies dramatically increase when the feedback loop delay is greater than one cycle [4]. Therefore, proactive voltage management based on architectural events has been proposed for voltage noise smoothing and voltage emergency mitigation [4], [5]. However, these prediction models can only predict whether an emergency will occur, rather than providing an accurate power trace that enables more detailed guidance to proactive management strategies.

To bridge this gap, this paper proposes a novel data-pattern-based OPM for float-point DNN accelerators named PROPHET. We identified some patterns with strong correlation with power consumption. By sampling them during memory access, we successfully predict upcoming power consumption dozens of clock cycles in advance. 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. Experiments on systolic-array-based DNN accelerators have demonstrated our model's accuracy and fine-grained prediction capabilities. Our contributions are summarized below.

- 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.
- The target data patterns captured by PROPHET can accurately reflect the power consumption of multiply-accumulate-based DNN accelerators. This observation also applies to other DNN accelerators with a regular PE array and the multiply-accumulate (MAC) structure.
- PROPHET achieves low-overhead and high-performance at the same time. The area and power overhead of PROPHET are lower than 2% in our experiments, and the temporal resolution can achieve 4 clock cycles with MAE $< 7\%$.

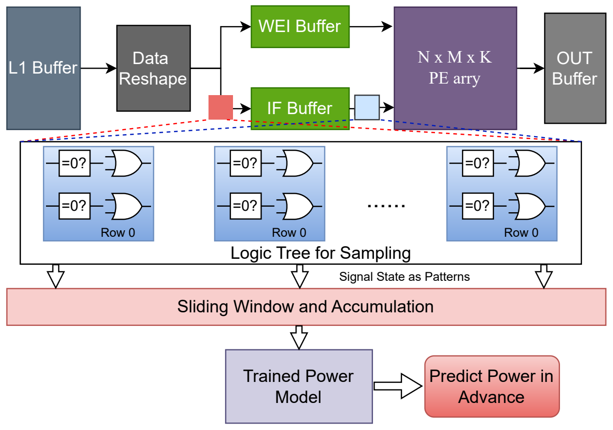


Fig. 1: PROPHET predicts the runtime power in DNN accelerators. The red block is PROPHET on write ports of the input-feature (IF) buffer. And the blue block is PROPHET on read ports. Both power models have the same structure.

II. Related Works

A. *On-chip Power Meters*

Prior research has investigated the use of performance counters to estimate the power consumption during runtime for both CPU and GPU architectures [6]–[9]. In these studies, micro-architecture events such as cache misses and the number of 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. Nonetheless, these models are typically utilized in coarse-grained management scenarios and lack flexibility due to limited access to certain event counters.

In recent years, some RTL-based runtime power models have been proposed for fast power managements by achieving high temporal resolution [1]–[3], [10]–[12]. They select the most power-correlated RTL signals, named power proxies, as the power model input. Notably, PowerProbe [2] can construct the on-chip power meters with a temporal resolution of 100-1K clock cycles, while maintaining the resource overhead of less than 8%. Simmani [3] achieves per-cycle temporal resolution for the Rocket RISC-V microprocessor but utilizes over 500 proxies, resulting in significant overhead. State-of-the-art solutions like APOLLO [1] can construct the OPM with per-cycle temporal resolution and less than 1% overhead. However, these methods are primarily designed for architectures with complex control flow and may not be suitable for data-streaming-based DNN accelerators. Moreover, their response time may still be insufficient for advanced power management techniques.

B. *Voltage Management*

Over the past two decades, many techniques have been proposed for managing supply voltage to maintain the system stability, including voltage emergency mitigation [4], [13] and voltage noise smoothing [5], [14]. There is a strong connection between power fluctuations and inductive voltage fluctuations. The works of [4], [15] proposed to predict the voltage emergency based on the micro-architecture events and take actions proactively. The works of [5], [14] 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.

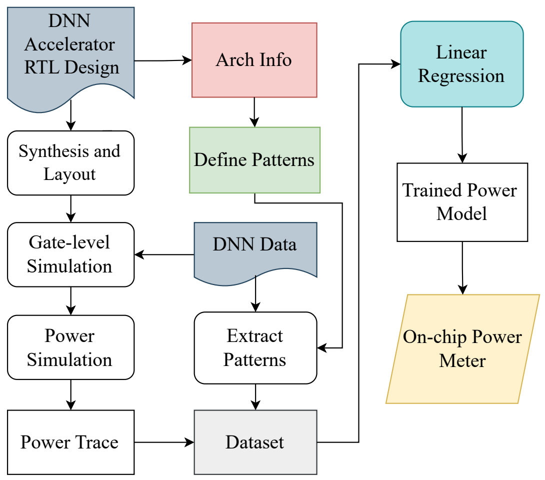


Fig. 2: The overall development framework of PROPHET.

To provide accurate and timely power trace for fast power management, we propose PROPHET, the first predictive and fine-grained on-chip power meter for DNN accelerators. As Fig. 1 shows, PROPHET can be implemented either on the feature buffers' write ports or read ports, where the sampling logic trees consisting of comparators and AND gates can extract pre-defined data patterns during memory access. These data patterns are then accumulated using a small sliding power measurement window to construct the input vector, enabling high temporal resolution. Finally, the power consumption value can be calculated using the off-line trained parameters and the sampled input vector. Since these sampling and calculation processes require only a few clock cycles, predicting power consumption in advance is possible by observing memory access patterns. Table I summarizes recent power modeling methods. Compared to previous power modeling methods, PROPHET can predict the power waveform dozens of clock cycles before it happens in an accurate, fine-grained, and low-cost way.

Method	Type	Resolution	Cost	Predictive
[6]–[9]	event counters	$\geq 1K$ cycles	low	✗
[2], [10], [11]	signal proxies	≥ 100 cycles	$5 \sim 15\%$	✗
[3]		per-cycle	$> 100\%$	
[1], [12]		per-cycle	$< 1\%$	
PROTHET	data patterns	≥ 4 cycles	$1 \sim 2\%$	✓

TABLE I: Comparison among various power models

III. Methodology

A. *Overall Framework of PROPHET*

Fig. 2 shows the PROPHET development framework. It first conducts accurate power simulations with input data of DNN model to gather ground-truth power traces. Simultaneously, our proposed data patterns are extracted from these input data. These patterns have a strong correlation with the power

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
G	The number of combinational logics in multiplier or adder
W	Number of cycles in sliding window
N_{sw}	The number of sub-windows in sliding window
R	Temporal resolution of power model

TABLE II: Description of frequently used symbols.

consumption of the specific systolic array architecture. The details of our data pattern definitions will be presented in Subsection III-B. Next, the extracted data patterns are combined with the power traces to create the dataset for the subsequent stage of power model training. The details of this dataset extraction and sampling procedure will be discussed in Subsection III-C. Finally, since our on-chip power meter uses data patterns as input, it can be implemented at the write ports of the input buffer to predict the power consumption of the DNN accelerator dozens of cycles in advance. Alternatively, the power meter at the read ports of the input buffer can estimate the run-time power. The difference between the two designs is discussed in detail in Subsection III-D.

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. We have discovered that the zeros in the PE arrays' inputs significantly impact the entire system's dynamic power consumption. The work of [16] has demonstrated that the sparsity of input feature maps can be a pattern to predict the workload for several milliseconds. However, such sparsity level can only reflect the power consumption in coarse-grained temporal resolution. Therefore, it is still necessary to identify more data patterns for fine-grained power modeling to construct the power model for timely voltage management.

As illustrated in Fig. 3, systolic array-based 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 arithmetic units in the PE can be abstracted as a combination of combinational logic gates G and registers. The dynamic power consumption of the entire DNN accelerator during the time window T can be represented by Eq. 1. P_{dyn_m} and P_{dyn_a} are the dynamic power of multipliers and adders, respectively. P_{other} is the power consumption generated by other components, such as memory, register, and local control logic, which are less affected by the input data. Thus, P_{other} can be approximated as a constant. The two input data of adders/multipliers 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. 2 and Eq. 3, respectively. Here, $\alpha_g(a,b)$ represents the toggle rate of the

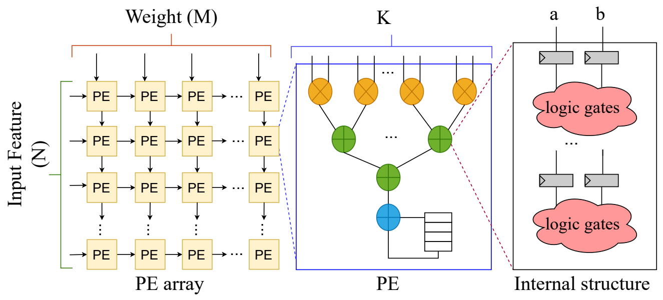


Fig. 3: The hierarchical structure of a typical PE array.

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 $\alpha_g(a,b) \in \mathcal{D}_g(a,b)$.

$$P_{dyn} = P_{dyn\_m} + P_{dyn\_a} + P_{other} \tag{1}$$

$$P_{dyn\_m} \propto \sum^{T}(\sum^{N}\sum^{M}\sum^{K}\sum_{g_1}^{G_1}\frac{1}{2}V^2\alpha_{g1}(a,b)C_{g1})/T \tag{2}$$

$$P_{dyn\_a} \propto \sum^{T}(\sum^{N}\sum^{M}\sum^{K}\sum_{g_2}^{G_2}\frac{1}{2}V^2\alpha_{g2}(a,b)C_{g2})/T \tag{3}$$

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. In this way, we can formulate the data-driven power model as Eq. 4, where the learnable power model parameter I is the average dynamic power corresponding to each input data situation, and α is the ratio of different input 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. 4.

$$P_{dyn} = P_{other} + \sum^{N}\sum^{M}\sum^{K}(\alpha_{m11}I_{m11} + \alpha_{m01}I_{m01}) + \sum^{N}\sum^{M}\sum^{K}(\alpha_{a11}I_{a11} + \alpha_{a01}I_{a01} + \alpha_{a00}I_{a00}) \tag{4}$$

C. Extraction and Sampling Data Patterns

We have discussed the data patterns related to zeros in input data of adders/multipliers a and b. For the multipliers, as shown in Fig. 3, their two inputs a and b correspond to input features and weights, respectively. Currently, due to the irregular memory access enabled for DNN algorithms after weight pruning, dense DNN algorithms are still the predominant workload for most general DNN accelerators. Executing a dense DNN algorithm on an accelerator generates a considerable amount of dynamic power consumption compared to the sparse DNN algorithm since the sparsity of weight data in sparse DNN can be more than 80% [17]. Therefore, our data-pattern-based power modeling focuses on the DNN accelerator when running dense DNN applications. Based on this, our data patterns can be extracted directly from the feature data (i.e., input a of multipliers), and all weights (i.e., input b of multipliers) are assumed to be non-zero by default. To achieve fine-grained power prediction in advance, we need to construct the dataset for power model training based on our predefined data patterns.

For DNN accelerators, the PE typically comprises multiple multipliers and adders. As Fig. 4 shows, since different zeros in a and b serve as data patterns for multipliers and adders, we can directly count these patterns on the PE input through the logic tree. The logic tree is constructed by abstracting the multiplier as a comparator and the adder as an AND gate. The states of signals $S \in \{S_0, S_1, \ldots, S_{2k-2}\}$ in the logic tree can reflect the data patterns of multipliers and adders. Therefore, we can record the states of signals in the logic tree as the data patterns of PEs to construct the input vector of the power model.

Another crucial factor is the pipeline length, which includes the arithmetic unit's pipeline within a PE and the pipeline between PEs in the PE array. As long as the valid input data remains in the pipeline structure, it will continue to affect the total power consumption. Thus, a sliding window that can record all data patterns during a certain period is necessary for fine-grained temporal resolution. Considering that the input feature data can transfer along the column dimension M of the PE array, the number of clock cycles in the sliding window W depends on the average pipeline length of the PE internal pipeline L and data transfer pipeline M, as shown in Eq. 5.

Regarding the on-chip power meter, considering the temporal resolution R as the stride of the sliding window, when $R < W$, there will be multiple sub-windows to record the patterns from each sampling. The number of sub-windows

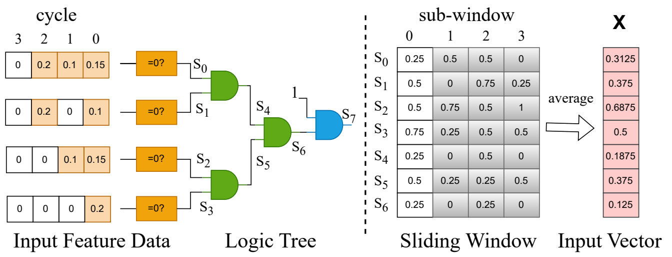


Fig. 4: The example of input vector generation for input feature data is when $K = 4$, $N_{sw} = 4$ and $R = 4$ (S_7 can be pruned as it is same as S_6). The logic tree corresponds to the PE structure in Fig. 3, with comparators and AND gates corresponding to multipliers and adders, respectively. When assuming weights are non-zero, compactors only need to check input features.

N_{sw} in the sliding window can be calculated as shown in Eq. 6.

$$W = L + [\frac{M-1}{2}] \tag{5}$$

$$N_{sw} = [W/R] \tag{6}$$

Assuming an $8 \times 8 \times 4$ systolic array, feature data is transferred between PEs along the M dimension and the multiply-accumulate (MAC) pipeline length structure in each PE is 12. Therefore, based on Eq. 5, the sliding window size W is 16. Fig. 4 shows an example of input vector generation when $K = 4$, $N_{sw} = 4$, and $R = 4$. The sampling logic tree consists of 4 comparators and 4 AND gates corresponding to 4 multipliers and 4 adders in the PE. The signal states (s) in the logic tree are recorded. Every cycle, four input feature data are input to the PE array. After 4 cycles, the sliding window shifts, and the signal states S during the latest 4 cycles are updated to sub-window 0 in Fig. 4. The values in the sliding window are then averaged to generate the new input vector.

To train the power model, we can construct a dataset based on the defined data patterns, sliding window, and stimulated power traces. The per-cycle pattern trace can be obtained by extracting the stimulation data using the equivalent logic tree. For power model training, a linear regression model can be employed. As shown in Eq. 7, the input vector $\mathbf{X}$, which corresponds to each $S \in \{S_0, S_1, \ldots, S_{2k-2}\}$, is constructed according to the resolution R, and P is the power consumption obtained from gate-level power simulation. After constructing the dataset, the data-pattern-based power model can be trained with a regression model. The trained parameter $\mathbf{W}$ and bias item b can then be implemented into the hardware system.

$$P = \sum_{i=0}^{2K-2}(X_i \times w_i) + b \tag{7}$$

D. Hardware Implementation and Power Prediction

Since the proposed power model is based on the data patterns of input feature data, after the hardware implementation, it samples data patterns during memory access. Fig. 1 has shown the general architecture of DNN accelerators, which comprises multi-level memories. The L1 buffer typically has a large capacity to store uniform data. The data reshape module

SA1		SA2	
symbols	value	symbols	value
M	8	M	4
N	4	N	4
K	4	K	8
L	22	L	28
W	32	W	32

TABLE III: Hardware Architecture Info

can convert 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 (WEI) buffer and input-feature (IF) buffer, respectively, for PE array computing. For fine-grained tracing of power consumption, the sampling of data patterns should ensure the same order of input data flow as in the PE array. Therefore, the feasible locations for integrating our proposed power meter are at the write or read ports of the feature buffer.

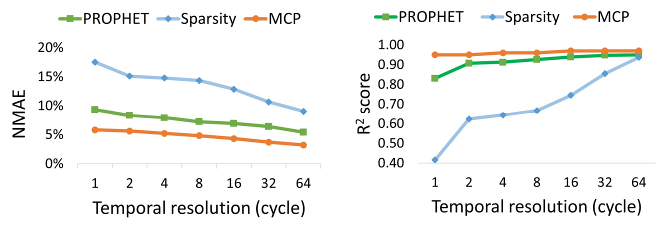
1) Power Estimation on Read Ports: The power meter at the read ports of the feature buffer can estimate the run-time power of the DNN accelerators since the data input flow sampled by the power meter is the same flow as that to the DNN accelerators. Data pattern samplers will monitor the signals' state in the sampling logic tree every clock cycle and then construct the input vector of the power model. The power consumption will be calculated based on the trained model. However, the power meter at the read ports of the feature buffer can only achieve run-time estimation because the response time left will be too short for any proactive power mitigation. Still, it will not be influenced by the control flow of the system.

2) Power Prediction on Write Ports: The power meter at the write ports of the feature buffer can predict the power consumption for a batch of data written in the buffer. The number of clock cycles predicted in advance is the time interval between the read and write operations of this batch of data. Unlike the samplers on the read ports, the controller can affect the data flow of the DNN accelerators. In our current experiments, we assume that the sampled data still follow the same flow as the read data, and these data will be read out to the accelerator without interruption. To further take a more complex control flow into consideration, our solution could be extended to address this impact by developing an additional sub-model for the controller, which will be studied in our future work.

IV. Experiment and Discussion

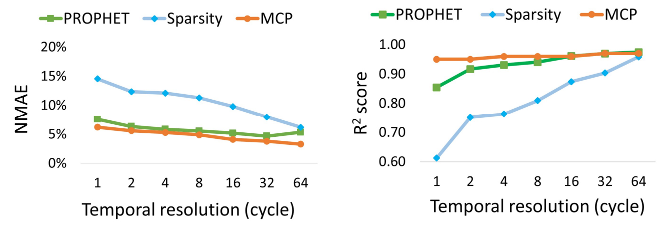
A. Experiment Setup

The simulation platform for experiments is implemented according to Fig. 1. The L1 buffer and data reshape modules are implemented in software for flexible simulation, while the rest of the modules are synthesized, placed, and routed with TSMC 40nm process. The ground-truth power simulation is performed with Synopsys PTPX. The DNN accelerators are implemented based on the widely-adopted output-stationary systolic array (SA) with various shapes and PE structures, as summarized in Table III. The data type used is floating-point numbers, which generate a large amount of dynamic power.



(a) NMAE (b) R^2 score

Fig. 5: Accuracy vs. temporal resolution for SA1.(sliding window W: 32 cycles)



(a) NMAE (b) R^2 score

Fig. 6: Accuracy vs. temporal resolution for SA2. (Sliding window W: 32 cycles)

Two systolic array accelerators are constructed using the same multiplier, adder, and accumulator, with pipeline lengths of 6, 6, and 4, respectively. The sliding window size is fixed to 32 clock cycles, which enables the shift operation to replace the divider and reduce overhead. The power modeling dataset comprises a 100,000-cycle power waveform obtained from gate-level power simulation. The stimuli are data from 10 convolution layers in VGG and ResNet.

We evaluate the performance of the proposed data-pattern-based power model with two different DNN accelerators in Table III. 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 using the normalized mean absolute error (NMAE), coefficient of determination R^2, and the overhead of power and area. We compare our model with two baselines: the one using sparsity as the data pattern [16] and minimax concave penalty (MCP) regression [1]. The method in [16] uses sparsity as the common data pattern for workload prediction for DNN accelerators. Meanwhile, the MCP technique adopted in APOLLO [1] 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.

B. Evaluate Data-Pattern-Based Power Model

We evaluated the accuracy, temporal resolution, and overhead of the three models. The MCP baseline selected about 200 proxies from RTL signals to construct the power model for SA1 and SA2, respectively. Fig. 5 and Fig. 6 show the R^2 score and the NMAE of power models under different temporal resolutions. Our data-pattern-based power model can

	SA1			SA2		
	SPA	MCP	OURS	SPA	MCP	OURS
Area (%)	0.21	1.15	0.96	0.31	1.17	1.96
Power (%)	0.18	0.90	0.91	0.33	0.96	1.96

TABLE IV: Hardware overhead of our power model. (SPA: only sparsity of input data as model input)

compete with the state-of-the-art MCP-based power model when the resolution is more than 4 clock cycles, while providing several additional benefits. First, PROPHET can predict power waveform dozens of clock cycles in advance during memory access. Second, PROPHET only requires 7 and 15 input patterns in the power model, while the MCP-based OPM requries 100-200 input proxies. For the MCP-based OPM, transitions of each proxy should be collected every clock cycle from various locations of the entire chip, which may increase the difficulty of layout for large-scale systems. Instead, PROPHET samples data patterns only on the input buffers, and the layout of the power model is more concentrated, reducing the overhead of the power meter.

In our experiment, we implemented four samplers for four input columns of the PE array, and the bit-width of parameters in the power model is 16 bits. Table IV shows the area and power overhead. For SA1 and SA2, the overhead of PROPHET is lower than 2%. And for the MCP-based model, the overhead is only around 1% beacuse multi-bit multipliers are replaced with AND gates. However, as the size of the DNN accelerator increases in M, N, and K dimensions, the area and power overhead of PROPHET will be lower than 1%, since the overhead and power are only linear with M and K. Although MCP and SPA can achieve lower overhead than our method, as seen in Fig. 5 and Fig. 6, SPA incurs a much larger power estimation error. As for MCP, since it is based on runtime counting of the signal switching activities inside the SA, it cannot be used for predictive power prediction.

C. Predict Power in Advance

PROPHET at the write ports of the input feature buffer is implemented to demonstrate the feasibility of the power prediction. The input buffer is a ping-pong FIFO that enables one batch of data to be written into the buffer first and then starts the computing pipeline. The depth of the ping-pong FIFO is 64, leading to a 64-cycle time interval between write and read. When the input feature buffer is full, the DNN accelerator will start reading data for computing, and the next batch of data will be put into the input buffer at the same time. Fig. 7 illustrates the predictive and ground-truth power waveform when the DNN accelerator is working. PROPHET can achieve power prediction of 56 clock cycles in advance for this buffer setting.

V. Conclusion and Future Work

In this paper, we proposed PROPHET, a fine-grained predictive on-chip power meter for DNN accelerators that can predict the power waveform dozens of cycles in advance. Our analysis and experimental results showed that PROPHET can achieve a temporal resolution of 4 clock cycles with $R^2 > 0.92$ and NMAE $< 7\%$ within 2% area/power overhead for two different systolic array accelerators. Moreover, we demonstrated that PROPHET can predict the power waveform 56 cycles in advance when the input buffer is a 64-depth ping-pong FIFO.

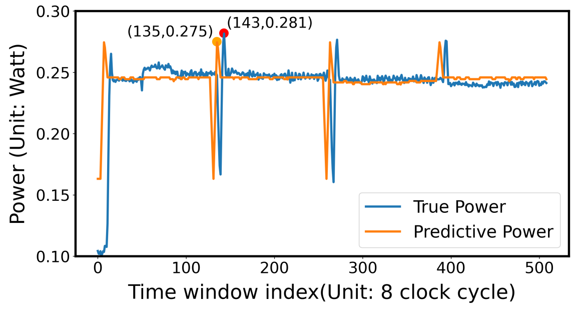


Fig. 7: PROPHET predicts the power waveform.

Our future research will focus on two directions. First, we will conduct a comprehensive evaluation and exploration of our data patterns using different architectures and data types. Second, we will explore high-performance fast power management techniques based on our power prediction model.

VI. Acknowledgments

This work is partially funded by Hong Kong RGC GRF 16213521 and Huawei Hong Kong Research Center (HKRC). We thank Chun Hang Lee and Jingbo Jiang in HKRC for their excellent critique and feedback.

References

[1] Z. Xie *et al.*, "APOLLO: An automated power modeling framework for runtime power introspection in high-volume commercial microprocessors," in *MICRO*, 2021.
[2] D. Zoni *et al.*, "Powerprobe: Run-time power modeling through automatic rtl instrumentation," in *DATE*. IEEE, 2018.
[3] D. Kim *et al.*, "Simmani: Runtime power modeling for arbitrary rtl with automatic signal selection," in *MICRO*, 2019.
[4] V. J. Reddi *et al.*, "Voltage emergency prediction: Using signatures to reduce operating margins," in *HPCA*. IEEE, 2009.
[5] V. K. Kalyanam *et al.*, "A proactive voltage-droop-mitigation system in a 7nm hexagon™ processor," in *VLSI*. IEEE, 2020.
[6] 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.
[7] F. Oboril *et al.*, "High-resolution online power monitoring for modern microprocessors," in *DATE*. IEEE, 2015.
[8] Y. Zhang *et al.*, "On-the-fly power-aware rendering," in *Computer Graphics Forum*. Wiley Online Library, 2018.
[9] M. Sagi *et al.*, "A lightweight nonlinear methodology to accurately model multicore processor power," *TCAD*, 2020.
[10] M. Najem *et al.*, "A design-time method for building cost-effective run-time power monitoring," *TCAD*, 2016.
[11] D. J. Pagliari *et al.*, "All-digital embedded meters for on-line power estimation," in *DATE*. IEEE, 2018.
[12] Z. Xie *et al.*, "DEEP: Developing extremely efficient runtime on-chip power meters," in *ICCAD*, 2022.
[13] M. S. Gupta *et al.*, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *DATE*. IEEE, 2007.
[14] 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.
[15] G. Papadimitriou *et al.*, "Harnessing voltage margins for energy efficiency in multicore cpus," in *MICRO*, 2017.
[16] S. Liu *et al.*, "Dynamic voltage and frequency scaling to improve energy-efficiency of hardware accelerators," in *HiPC*. IEEE, 2021.
[17] S. Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, 2016.