A Fast Cross-Layer Dynamic Power Estimation Method by Tracking Cycle-Accurate Activity Factors With Spark Streaming

Leilei Jin, Wenjie Fu, Ming Ling, *Member, IEEE*, and Longxing Shi, *Senior Member, IEEE*

***Abstract*— The advent of autonomous power-limited systems poses a new challenge for early design space exploration. The existing architecture-level power evaluation tools lose accuracy due to ignoring features of circuit-level behaviors and influences of process, voltage, and temperature variations. Although power estimations based on SPICE or PrimeTime PX (PTPX) are accurate enough, they come at the cost of long simulation time and are available only in very late phases of design flow. In this article, a fast and accurate dynamic power evaluation method is proposed, which estimates activity factors at the circuit level. The impact of process variation at the gate level is considered through the proposed effective capacitance model. Activity factors are then estimated by the model and input vectors of the circuit. Input vectors are generated by architecture-level simulations in the form of streaming. For real-time and high-speed power evaluation, a data streaming framework is proposed for massive parallelism. The cross-layer estimation is verified based on the functional units of PULPino processor running SPEC CPU2006 benchmarks. Compared with the SPICE results using SMIC 28-nm PDK, our cycle-by-cycle dynamic power analysis shows an average error of 5.4%. Meanwhile, our approach realizes 65.2% faster than the traditional PTPX simulation and 48.8% faster compared with the state-of-art cross-level evaluation method.**

***Index Terms*— Activity factor, dynamic power, effective capacitance estimation, process variations, worse case power estimation.**

I. Introduction

THE growing demand for energy-constrained systems is driven by the increasing prevalence of battery-powered devices and artificial Intelligence of Things (AIoT). To support architectural design space exploration, it is important to quickly and accurately quantify the power consumption of processors early in the design flow [1]. Furthermore, for different workloads, detailed real-time results are often used to support architecture and software optimizations [2]. Unfortunately, time-related dynamic power as a primary factor in power dissipation is hard to estimate since it is highly workload-dependent [28]. Such real-time estimation is only available through extremely detailed circuit-level simulations for obtaining circuit dynamic information at each cycle or with each input vector [29]. It is usually of high accuracy but too time-consuming and is unacceptable for shortening the design space exploration time. Therefore, a real-time and fast estimation is necessary to calculate the architecture-level dynamic power using the circuit-level characteristics, which are also named bottom-up estimations. Besides, the massive amount of real-time power information brings the issues of storing and processing, which appeals to big-data processing tools [3].

Various bottom-up power estimation tools have been developed [4], such as CACTI [5] and McPAT [6]. These tools are used to simulate the power consumption of CPU components, e.g., functional units and caches. In particular, McPAT is a popular power and timing evaluation tool that calculates dynamic power based on a constant empirical activity factor for each component, ignoring the feature of workloads. As demonstrated in several pieces of researches [7], [8], McPAT has average power errors of over 20% for most of the tested workloads when comparing McPAT with a physical ARM Cortex-A9 device. In [9], the power consumption variability of functional units is considered to follow near-Gaussian distributions due to process variations. Under this circumstance, the 99th percentile of the distribution is taken to represent the result of worst case power consumption estimation [9], [10]. However, these works still give a constant activity factor for dynamic power estimation. The accurate activity factor of each workload needs to be collected from extremely detailed circuit-level simulations. Commercial dynamic power analysis tools, such as Synopsys PrimeTime PX (PTPX), give cycle-accurate power dissipation, which needs value change dump (VCD) files to store the circuit switching activity of each cycle [11]. It requires huge time and storage when analyzing processors with these files.

In this work, we first construct a gate-level capacitance model to predict the total capacitance of the switched nodes in the target circuit. Based on SMIC 28-nm PDK, the process variation effect is quantified in the model and the effective capacitance is predicted using polynomial fitting. In each cycle, the model takes input vectors collected from an architecture-level simulator to identify whether the state of each node is toggled or not. The activity factor of each cycle can then be calculated by dividing the effective capacitance by the total capacitance of the circuit. The modified McPAT inte-

Manuscript received May 19, 2021; revised August 3, 2021 and August 22, 2021; accepted September 5, 2021. Date of publication September 15, 2021; date of current version March 22, 2022. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61874152 and Grant 61974024. *(Corresponding author: Ming Ling.)*

The authors are with the National ASIC System Engineering Technology Research Center, Southeast University, Nanjing 210096, China (e-mail: leileijin@seu.edu.cn; trio@seu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3111000.

Digital Object Identifier 10.1109/TVLSI.2021.3111000

1063-8210 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

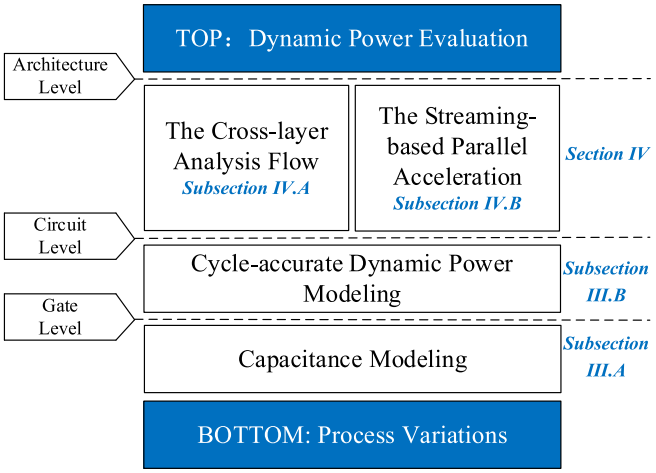


Fig. 1. Overview diagram of the proposed cross-layer dynamic power estimation method.

grated with the gate-level models can give accurate dynamic power analysis results with these activity factors. For a real-time and high-speed power evaluation, a streaming-based flow is proposed to catch the input vectors and feed them into a data streaming engine that calculates activity factors at the same time. Finally, we implement our method on each functional unit of PULPino [12] to find out the precise power behavior under different workloads that run in the near-threshold voltage region.

The main contributions of this article can be summarized as follows.

1) Proposing a gate-level capacitance model to quantify the capacitance distribution of each node of the circuit under process variation effect, through a Gaussian variable and polynomial fitting. The activity factor is predicted by the model and the state of each node. Resulting from underlying process variations, the power variability and its worst case consumption in each cycle are quantified by the capacitance model.
2) Proposing a high-speed cross-layer framework based on data streaming for real-time dynamic power evaluation. It automatically obtains input vectors of the specific workload from the architecture-level simulator and performs the modified McPAT to provide cycle-by-cycle dynamic power distributions.

As shown in Fig. 1, the bottom-up cross-layer dynamic power analysis flow consists of three main steps: gate-level capacitance modeling, circuit-level dynamic power modeling, and architecture-level evaluation. The rest of this article is structured as follows: Section II presents related works. Section III describes the method of dynamic power modeling. Section IV gives details of the architecture-level power evaluation flow. The experimental setup and the accuracy of the method are discussed in Section V. Finally, Section VI concludes this article.

II. Related Works

A. Power Estimation Method

Power models can be broadly split into two key categories: bottom-up power models, e.g., McPAT, CACTI, and top-down power models, e.g., performance monitoring counter (PMC)-based models [14], [15]. Top-down power models are built from PMC data of a specific CPU for power consumption measurement. Many CPUs have PMCs that count certain architectural events to record CPU behaviors, such as the number of L2 cache misses and the number of execution cycles. However, collecting PMCs from hardware is not straightforward for power evaluation; instead, the real-time power is usually quantified by regressions. Top-down CPU power models are more accurate but only suitable for post-silicon evaluations.

Taking advantage of the available circuit information (e.g., gate-level netlist, extracted parasitic, and layout), the bottom-up power models use information of each component (e.g., cache size and the length and width of the pipeline) for architecture-level evaluation. As it is feasible even during early design phases, bottom-up tools are widely used, such as McPAT [6], CACTI [5]. CACTI is a tool for estimating the timing, area, and power of caches and memories using gate-level empirical models based on scaling factors from the ITRS roadmap. McPAT is developed based on CACTI and widely used for whole processor evaluation. The dynamic power evaluation using McPAT captures architecture-level behaviors through the cycle-accurate simulator, gem5 [16]. The work [17] uses analytical models to capture the inputs of McPAT with significant acceleration of time but losses real-time behaviors. Furthermore, McPAT estimates the dynamic power of logical components by empirical expressions, ignoring the detailed switching activities during the runtime.

The dynamic power is affected by the variations in several parameters, such as frequency and temperature. In [35], the impact of frequency is mainly quantified through circuit delay modeling. In addition, McPAT-PVT considers the process variation effect on the average power and frequency, other than the detailed activities of each cycle, let alone real-time evaluations [13]. So, the existing bottom-up power models merely provide average results of different workloads, other than real-time evaluations from the top-down methods.

Some works estimate the real-time dynamic power by tracking activity factors with circuit switching information using machine learning [18], [19]. At register transfer level (RTL) level, PRIMAL predicts activity factors by training input vectors [18]. However, the establishment of a neural network is time-consuming due to sampling and network training and cannot be used for power evaluation of a given application directly.

Commercial dynamic power analysis tools, such as Synopsys PTPX, generate accurate cycle-by-cycle power analysis results [11]. They rely on VCD files that store circuit switching activities. The analysis is too time-consuming for practical estimation of large-scale applications in real-time. Compared with the above researches, a bottom-up method is proposed in this work, from the gate level to the architecture level. The

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

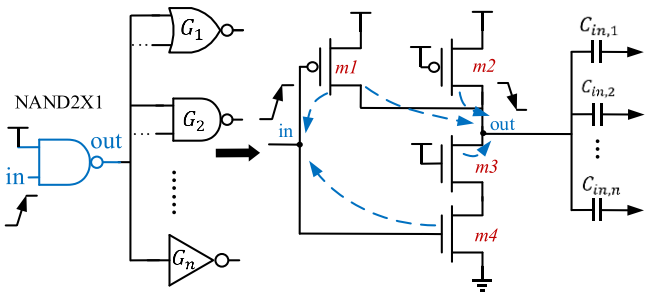


Fig. 2. NAND2X1 gate with different fan-out gates.

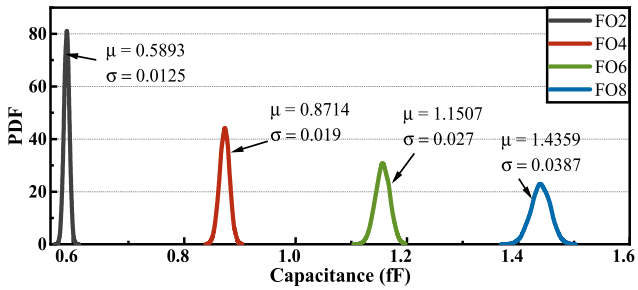


Fig. 3. Distributions of load capacitance of an INVX1 with different number of fan-out gates (SMIC 28 nm).

gate-level capacitance model is built to get activity factors under each input vector. The vector is generated from the architecture-level simulator, gem5.

B. Parallel Accelerating Method

Algorithms that highlight multithreaded and massively parallel accelerations are strongly encouraged for speeding up dynamic power analysis [1]. However, current literature is still short of novel ideas for fast parallel dynamic power analysis. The development of platforms for analytical data processing has become a popular direction [3]. The best known among such platforms is Apache Hadoop [20]. Hadoop is a set of software utilities, the core of which is a distributed file system, and a data processor that implements the MapReduce processing model [21]. In the ASIC physical design flow, MapReduce is used for path-based timing analysis [22], [23]. It is still time-consuming due to the slow hard disk reading and writing.

Differently, Spark is hundreds of times faster than MapReduce as it is a memory-based engine [24]. The stream processing function of Spark makes it appropriate for problems that contain several steps with inputs and outputs. In this work, a cross-layer power evaluation framework is proposed considering gate-level capacitance variations, circuit-level activities, and architecture-level workload features. Throughout each level, Spark is used to capture the output stream from the architecture level, convert them into input vectors, estimate the activity factor of each vector at the circuit level, and pipe them into the modified McPAT with gate-level models.

III. Dynamic Power Modeling

A. Gate-Level Capacitance Modeling

The dynamic power is mainly determined by the load capacitance of each gate [25], [33]. Owing to the process variation effect, the value of load capacitance is in the form of a distribution, which leads to the variability of dynamic power. Hence, capacitance distribution modeling is the key to accurate power estimation. As shown in Fig. 2, given a NAND2X1 gate connected with several fan-out gates $(G_1, G_2, \ldots, G_n)$, the load capacitance (C_{L}) can be quantified by the capacitance of its output node and each input node capacitance of its fan-out gates. It can be calculated by (1), where C_{out} represents the capacitance of its output node and $C_{\mathrm{in},i} (i = 1, 2, \ldots, n)$ represents the input of each gate

$$C_{\mathrm{L}} = C_{\mathrm{out}} + \sum_{1 \le i \le n} C_{\mathrm{in},i}. \tag{1}$$

The capacitance of the input node is obtained by summating the gate capacitances and overlap capacitances from gate to source and drain of MOSFET m1 and m4. Also, the capacitance of the output node can be calculated by the summation of the gate–source overlap capacitances from the output node and the drain-body capacitances of MOSFET m1–m3. The capacitance model is based on the transistor-level simulations in SPICE [36] which is sufficiently precise with process geometries shrinking.

For each gate, the capacitance of each input–output node is fixed when the input transition state (rise/fall) and the driven pin (A1/A2) are given. Hence, the node capacitance is modeled directly under every driven state, instead of deduction from device-level capacitances. Defined from the manufacturing foundry, a MOSFET contains global process variations and local process variations, representing die-to-die variations and with-in-die variations, respectively. The value of gate capacitance is in the form of a distribution because of the non-negligible influence of those process variations. The capacitance distribution is represented by a random variable x with the density function in the form of (2), where μ and σ stand for the mean and standard deviation, respectively

$$f(x) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{2}$$

Therefore, the calculation of load capacitance (C_{L}) turns into a statistical summation between C_{out} and $C_{\mathrm{in},i}$. Fig. 3 plots the probability density function (pdf) of the load capacitance distribution of an INVX1 with a different number of fan-out gates using SPICE Monte Carlo simulations. FOn states for the inverter with n fan-out inverters of the same type. The values of the mean and standard deviation of load capacitance distribution are proportional to the number of fan-out gates and the distributions maintain Gaussian-like shapes.

Supposing gate A contains n fan-out gates, its load capacitance also follows Gaussian distribution, represented by $C_{\mathrm{L}} \sim N(\mu_{\mathrm{L}}, \sigma_{\mathrm{L}}^2)$. Based on Gaussian summation theory, the given output node capacitance follows $N(\mu_{\mathrm{out}}, \sigma_{\mathrm{out}}^2)$ and the input node capacitance of the ith fan-out gate follows $N(\mu_{C_{\mathrm{in},i}}, \sigma_{C_{\mathrm{in},i}}^2)$, and the mean of load capacitance distribution

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

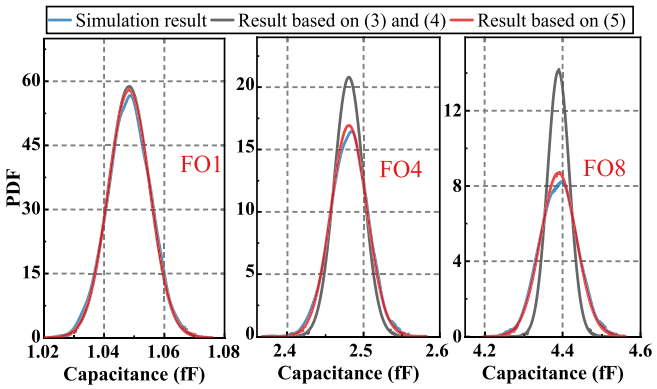


Fig. 4. Comparison of load capacitance of NAND2X1 using Gaussian theory and polynomial fitting.

μ_{L} can be calculated through the following equation:

$$\mu_{\mathrm{L}} = \mu_{\mathrm{out}} + \sum_{1}^{n} \mu_{\mathrm{in},i}. \tag{3}$$

Similarly, the standard deviation σ_{L} can be obtained by the deviation of each random variable. Because of the with-in-die process variation, C_{out} and $C_{\mathrm{in},i}$ are assumed as jointly normally distributed variables. To calculate σ_{L}, it needs to quantify the correlation between input and output capacitances through the covariance $\mathrm{Cov}(C_{\mathrm{out}}, C_{\mathrm{in},i})$, and the correlation between each input using $\mathrm{Cov}(C_{\mathrm{in},i}, C_{\mathrm{in},j})$

$$\sigma_{\mathrm{L}}^2 = \sigma_{\mathrm{out}}^2 + \sum_{1}^{n} \sigma_{\mathrm{in},i}^2 + 2\sum_{1}^{n} \mathrm{Cov}\left(C_{\mathrm{out}}, C_{\mathrm{in},i}\right) + 2 \sum_{\substack{i,j=1,2,\ldots,n \\ i\neq j}} \mathrm{Cov}\left(C_{\mathrm{in},i}, C_{\mathrm{in},j}\right). \tag{4}$$

However, the deduction of (3) and (4) is not accurate enough based on Gaussian summation theory, because the variables $(C_{\mathrm{out}}, C_{\mathrm{in},i})$ in (1) are affected by several process variations through multiplying and summating with each other. Considering the uncertainty of the process variation effect, we introduce another polynomial model to estimate the mean and standard deviation, as shown in (5). The importance of C_{out} and $C_{\mathrm{in},i}$ to C_{L} is quantified by the coefficients a_1, a_2 and b_1 to b_4 and can be quantified through (6)–(12)

$$\begin{cases} \mu_{\mathrm{L}} = a_1 \mu_{\mathrm{out}} + a_2 \sum_1^n \mu_{\mathrm{in},i} \\ \sigma_{\mathrm{L}}^2 = b_1 \sigma_{\mathrm{out}}^2 + b_2 \sum_1^n \sigma_{\mathrm{in},i}^2 + b_3 \sum_1^n \sigma_{\mathrm{out}} \sigma_{\mathrm{in},i.} \\ \qquad + b_4 \sum_{\substack{i,j=1,\ldots,n \\ i\neq j}} \sigma_{\mathrm{in},i} \sigma_{\mathrm{in},j} \end{cases} \tag{5}$$

The density function of load capacitance is complicated and not in the form of Gaussian function. As shown in Fig. 3, despite the shape of capacitance distribution being similar to that of Gaussian distribution, the summation of capacitance cannot be simply calculated by the Gaussian summation theory. Fig. 4 compares the simulated load capacitances of FO1, FO4, and FO8 with the estimated values using Gaussian theory and our proposed polynormal fitting method. The accuracy of estimating the load capacitance using Gaussian theory gets worse with the increasing number of fan-out gates. Compared with (4), the recalculated σ_{L}^2 in (5) gets larger and is closer to the simulation result.

The load capacitance expressed by a polynomial in (5) is completely determined by its coefficients. The multi-variable polynomial fitting needs to find suitable coefficients to fit the mean and standard deviation of load capacitance. This can be solved by setting up a system of equations based on samples obtained from SPICE simulations. Given a dataset $\{s_i \in S | i = 1, \ldots, N\}$ of N samples, the polynomial fitting model of the mean μ_{L} is written in the vector form as (6), where $\varepsilon_\mu \in \mathbb{R}^m$ is the residual. The vectors $\mathbf{X}$ and $\mathbf{A}$ are represented by (7)

$$\mu_{\mathrm{L}} = \mathbf{X}\mathbf{A} + \varepsilon_\mu \tag{6}$$

$$\mathbf{X} = \left[\mu_{\mathrm{out}}, \sum \mu_{\mathrm{in},i}\right], \quad \mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}. \tag{7}$$

Similarly, the standard deviation is expressed as (8) with its residual $\varepsilon_\sigma \in \mathbb{R}^m$. The vectors $\mathbf{Y}$ and $\mathbf{B}$ are given by (9) and (10)

$$\sigma_{\mathrm{L}}^2 = \mathbf{Y}\mathbf{B} + \varepsilon_\sigma \tag{8}$$

$$\mathbf{Y} = \left[\sigma_{\mathrm{out}}^2, \sum \sigma_{\mathrm{in},i}^2, \sum \sigma_{\mathrm{out}} \sigma_{\mathrm{in},i}, \sum \sigma_{\mathrm{in},i} \sigma_{\mathrm{in},j}\right] \tag{9}$$

$$\mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}. \tag{10}$$

To obtain the coefficient vectors $\mathbf{A}$ and $\mathbf{B}$, the ordinary least-squares criterion is followed to minimize the sum of squared residuals with the Euclidean L2-norm $||\cdot||$, written as follows [26]:

$$\begin{cases} \hat{A} = \mathrm{argmin}\left\{\|\mu_{\mathrm{L}} - \mathbf{X}\mathbf{A}\|_2^2\right\} \\ \hat{B} = \mathrm{argmin}\left\{\|\sigma_{\mathrm{L}}^2 - \mathbf{Y}\mathbf{B}\|_2^2\right\}. \end{cases} \tag{11}$$

Eventually, the estimated coefficients $\hat{A}$ and $\hat{B}$ are obtained by solving the normal equation, as shown in (12). The polynomial results can be calculated exclusively by the coefficients above

$$\begin{cases} \hat{A} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mu_{\mathrm{L}} \\ \hat{B} = \left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\sigma_{\mathrm{L}}^2. \end{cases} \tag{12}$$

For the worst case dynamic power estimation, the 99th percentile of the load capacitance distribution is taken as a probabilistic estimation of maximum power [10]. Despite the polynomial model being used in the end, the distribution maintains a Gaussian-like shape. The three-sigma rule of thumb expresses that nearly all values are taken to lie within three standard deviations of the mean, and thus it is empirically useful to treat 99.86% probability as the worst case result [32]. Hence, the worse case load capacitance of each gate can be calculated through $\mu_{\mathrm{L}} + 3\sigma_{\mathrm{L}}$.

B. Cycle-Accurate Dynamic Power Modeling

In some works [25], [34], the dynamic power consists of leakage power and switching power. While in other works [6], [37], [38], the power model is separated into three main components: dynamic power, short-circuit power, and

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

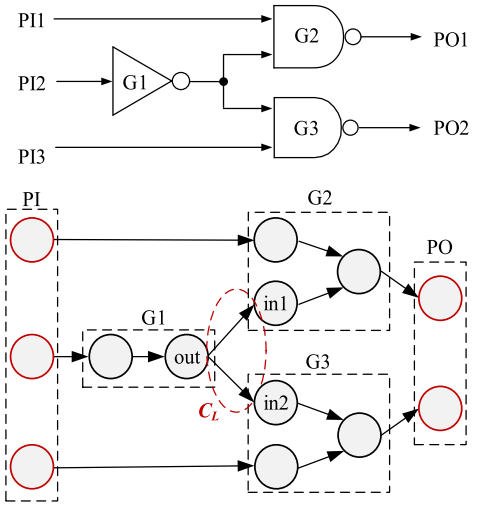


Fig. 5. Circuit graph consists of PI, combinational gates (G_i), and PO.

leakage power. This work mainly focuses on switching power estimation and its key element: activity factor, which can also be used to evaluate the short-circuit power [39].

In each cycle, a number of gates are switched ($0 \rightarrow 1$ or $1 \rightarrow 0$) because of the transformation of input signals. The dynamic power P_{dyn} of cycle k can be calculated by summarizing the power of the switched gates. The ratio of the capacitance of the switched gates to the total capacitance is represented as activity factor α. So that the dynamic power should be the ratio times the peak dynamic power. Equation (13) shows the dynamic power consumption of a target circuit in cycle k, where f is the clock frequency, V_{dd} is the supply voltage, C_{total} is the total load capacitance of the whole circuit, and $\alpha[k]$ is the activity factor of cycle k [27]

$$P_{\text{dyn}}[k] = \alpha[k] \cdot C_{\text{total}} \cdot V_{\text{dd}}^2 \cdot f. \tag{13}$$

Usually, the multiplied result $\alpha[k] \cdot C_{\text{total}}$ is equivalent to the effective capacitance $C_{\text{eff}}[k]$, which can be calculated by summarizing the load capacitance C_{L} of the toggled gates whose input signals are switched ($0 \rightarrow 1$ or $1 \rightarrow 0$) in cycle k. Finally, the cycle-accurate dynamic power can be calculated through the following equation:

$$P_{\text{dyn}}[k] = C_{\text{eff}}[k] \cdot V_{\text{dd}}^2 \cdot f = \sum C_{\text{L}}[k] \cdot V_{\text{dd}}^2 \cdot f. \tag{14}$$

It is important to identify the toggled gates at each cycle for estimating the dynamic power. As shown in Fig. 5, the circuit is first transformed to a directed acyclic graph (DAG) which consists of primary inputs (PI), combinational gates (G_i), and primary outputs (PO). Each gate is converted into a group of input nodes and an output node, with node capacitances $C_{\text{in},i}$ and C_{out}, respectively. Then, the load capacitance of each cell can be predicted using (5), by means of the capacitance propagation of the whole circuit.

Before a detailed description of the cycle-accurate estimation, we first give the following definitions.

1) *Toggled Pin:* A pin is toggled in cycle $[k]$ when its value is changed compared with the value in cycle $[k-1]$.

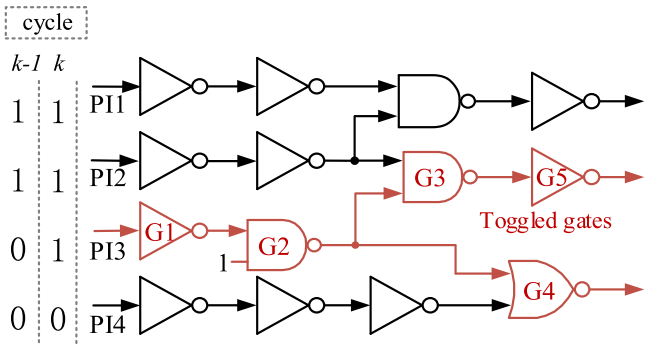


Fig. 6. Toggled gates during the circuit activity analysis.

Algorithm 1 Calculating the Dynamic Power of Each Cycle

Input: input vector of cycle k, DAG
Output: dynamic power $P_{dyn}[k]$ in each cycle

```
1.  reset_to_untoggled( );
2.  foreach pin p in DAG do
3.      analyze_rise_fall_state( );
4.      if state not equal to previous state do
5.          p->toggled ← true
6.      end if
7.      if p->toggled do
8.          C_eff [k] ← C_eff [k] + C_L
9.      end if
10. end for
11. P_dyn [k] ← C_eff [k]
12. return P_dyn[k]
```

2) *Toggled Gate:* A gate is toggled in cycle $[k]$ when its output pin is a toggled-pin.
3) *Toggled Port:* The ports are the input pins of the whole circuit. A port is toggled in cycle $[k]$ when the input vector of the port is changed compared with that in cycle $[k-1]$.

Dynamic power analysis of a target design can be performed by identifying the gates that are toggled at a particular cycle, ignoring all untoggled gates. As shown in Fig. 6, the toggled port of the input vector in the cycle $[k]$ is PI3 and gates G_1 to G_5 are toggled gates. The effective capacitance C_{eff} is the summation of the load capacitance of gates G_1 to G_5. The pseudocode for calculating the cycle-accurate dynamic power is given in Algorithm 1. First, all gates, pins, and ports are unmarked as untoggled. The state of each pin is analyzed and compared with the state in the previous cycle. If the state is changed, the pin is set to be toggled. The load capacitances of toggled gates are summarized as the effective capacitance in each cycle. After that, the dynamic power in each cycle will be calculated through (14).

In this detailed power analysis, signal activities are propagated into gate-level behaviors, and the cycle-by-cycle power of the target design is computed. Since the full cycle-by-cycle power traces are generated instead of the constant average power in the original McPAT, our process tends to be with high precision. Meanwhile, our DAG-based dynamic power

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

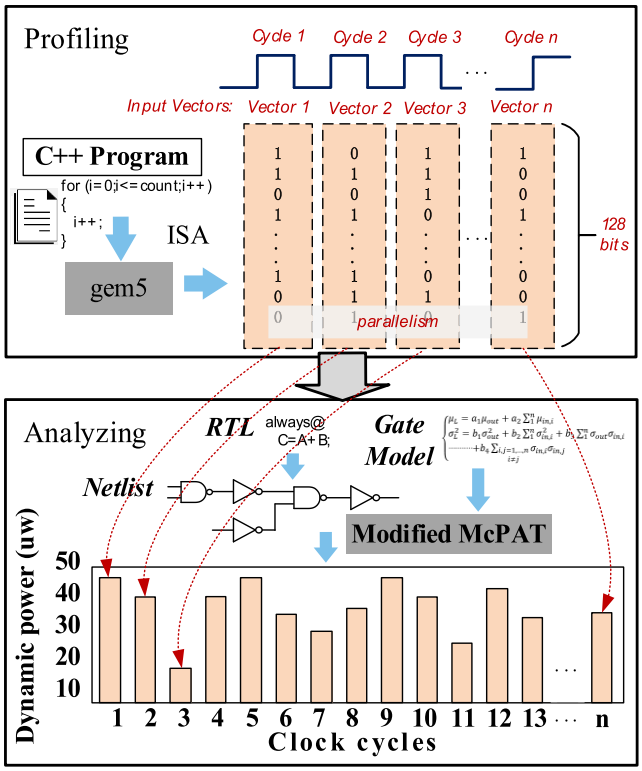


Fig. 7. Overview of the cross-layer cycle-accurate dynamic power analysis flow.

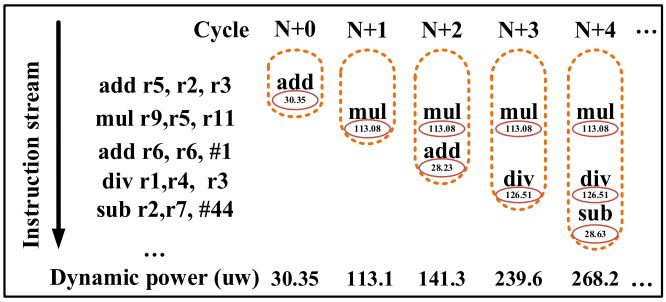


Fig. 8. Cycle-accurate dynamic power estimation across instruction streams.

Algorithm 2 Parallel Analysis Flow in Spark

Input: Benchmarks, netlists and period Δn
Output: cycle-accurate dynamic power

1. **for** i **from** 1 **to** N **step by** Δn
2. **for** cycle k **from** i **to** $i+\Delta n$ **do**
3. $\#RDD_1$: input vector, instruction type T, stats.txt ← benchmark.$pipe$(gem5);
4. $\#RDD_2$: $p[k] \leftarrow RDD_1.map$(Algorithm 1);
5. **end for**
6. RDD_3: config ←$RDD_2 + RDD_1.map$(stats.txt, p);
7. RDD_4: $P_{dyn}[\frac{i-1}{\Delta n}+1] \leftarrow RDD_3.pipe$(modified McPAT);
8. **end for**
9. $P_{dyn} \leftarrow RDD_4.collect()$
10. **return** P_{dyn}

* The character # means the parallel operation

analysis achieves significant performance benefits compared with the commercial power analysis tool PTPX, which needs additional simulations for getting the circuit activity file (.vcd).

IV. Architectural Parallel Analysis

The cross-layer power evaluation flow from the gate-level to the architecture-level is discussed in Section IV-A. The data streaming method is established using Spark to realize the high-speed evaluation demand, which is introduced in Section IV-B.

A. Cross-Layer Analysis Flow

Our proposed dynamic power analysis method adopts combined toolchains to accomplish cross-layer estimation. Fig. 7 depicts the overall flow, which consists of two stages: 1) profiling and 2) analyzing. In the profiling stage, it takes the benchmark as input and exports the instruction sequences obtained by gem5 simulations. At the architecture level, an extraction strategy for getting the instruction sequence is designed as an extension to the RISC-V ISA decoder in gem5. In each cycle, the type of instruction is dumped and the values of operand registers are converted into a binary format (represented by the orange dash boxes). The binary data are considered to be input vectors of the circuit in the corresponding cycle.

In the analyzing stage, the RTL of ALU, the gate-level load capacitance model, and input vectors are taken. The RTL is first synthesized into the gate-level netlist and then converted into DAG. The load capacitance model is integrated into the modified McPAT for power measurements. The coefficients of the model are stored in an external file, which can be indexed by the gate type and the input pin. Based on the DAG, the cycle-accurate dynamic power is calculated in the modified McPAT through Algorithm 1. As can be seen in the orange bars in Fig. 7, a real-time dynamic power is observed within every clock cycle. The profiling and analyzing processes can be performed in parallel (the red dash-dotted line) and will be introduced in detail in Section IV-B.

As illustrated in Fig. 8, the dynamic power for every cycle is extracted considering instruction streams, such as addition (add) and multiply (mul). For example, the first add instruction contains two input operands. The values in the corresponding operand registers (r2 and r3) are converted into binary streams as the input vectors of ALU and then do the power analysis. For multi-cycle instructions (mul and div), their overall powers are divided into several values for each cycle.

B. Streaming-Based Parallel Acceleration

The high-speed power evaluation is realized using data streaming with a computing engine, Spark [24]. The engine uses the master–slave architecture that consists of a master node and several worker nodes. The master node manages the resources of the node and assigns tasks to worker nodes. Spark Streaming is one of the key components in Spark specially designed for real-time streaming problems.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

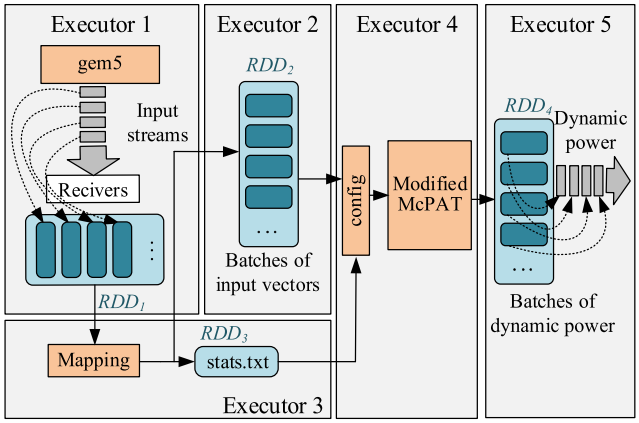


Fig. 9. Streaming-based power evaluation workflow.

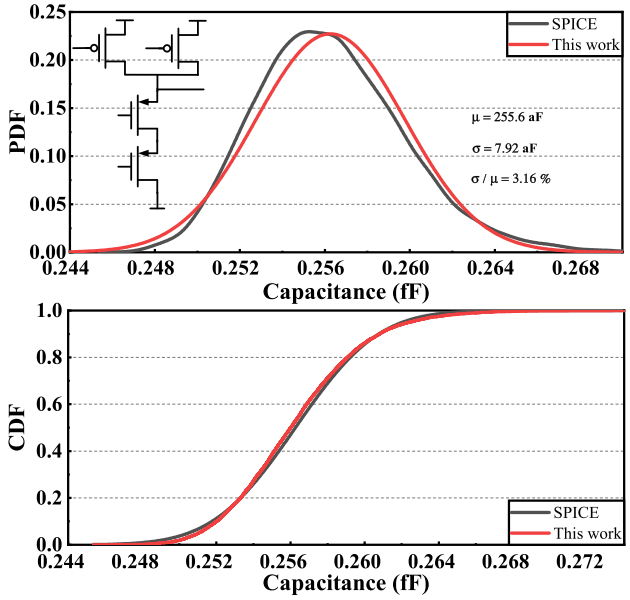


Fig. 10. Comparison of the output node capacitance modeling.

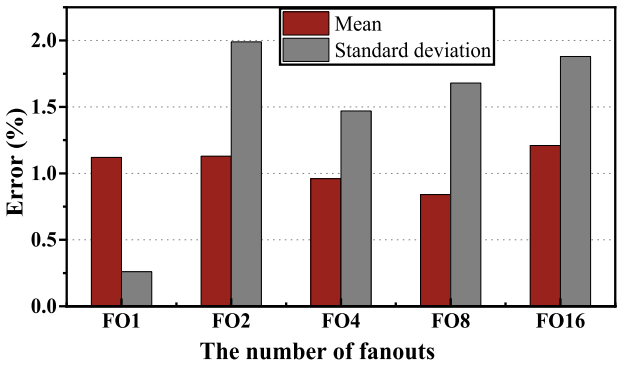


Fig. 11. Average relative errors of the mean and standard deviation compared with SPICE simulations.

The dynamic power analysis process is given in Algorithm 2. In the data processing stage, we create a receiver by Spark Receiver class on one of the worker nodes. The receiver collects input vectors generated from gem5 and converts them into Spark's resilient distributed datasets, represented by RDD_1. It also contains the instruction type to identify which functional unit to analyze. RDD_1 is transformed into another one by the map method of Spark. The map transformation maps the input vector onto the circuit netlist of the instruction type T. The cycle-by-cycle effective capacitance C_{eff} of the corresponding input vector is calculated using Algorithm 1. The output stream of RDD_2 is fed into the modified McPAT to estimate the dynamic power $P_{\text{dyn}}[(i-1/\Delta n)+1]$. Finally, the dynamic power of the whole benchmark can be analyzed by collecting the output stream of each RDD_3 using the collect method. To process RDD data using external toolchains (McPAT and gem5), we use the pipe method in Spark. The input vector and the instruction type T are piped from gem5 and taken as the input for Algorithm 1. Similarly, our modified McPAT is used by the pipe method.

Specifically, the multithreaded implementation in Spark can take advantage of machines with a large number of cores to decrease the computation time. Parallelized collections can be created by calling the parallelize method. The number of partitions as an important parameter is set to cut the stream into partitions. Spark will take each partition as a thread for parallel processing. Once the parallelized collection is created, multiple groups of input vectors can be operated simultaneously. The overall parallel workflow is shown in Fig. 9. Batches of input vectors are collected from gem5 and put into the RDD at the same time. The dynamic power of each RDD is also calculated in parallel.

V. Evaluation

A. Experimental Setup

At the gate level, each gate is modeled and verified using SMIC 28-nm PDK. The netlists of functional units of PULPino are generated through Design Compiler. In the architecture-level analysis, SPEC CPU2006 benchmarks are used to collect the instruction sequences [31]. The instruction sequence profiling, the DAG establishment, and the effective capacitance calculation are realized using C++ language. The supply voltage is set to 0.6 V and the operating frequency is set to 500 MHz. Spark 3.0.0. is installed in the Linux system (CentOS release 7.9.2009). All experiments are run on a 3.60-GHz Intel i7-7700 Processor with eight cores and 16-GB memory.

B. Gate Model Accuracy

The gate-level capacitance modeling is built using the following standard cells for simplification, which are INVX1/2/4/8, NAND2X1/2/4/8, and NOR2X1/2/4/8. These are the most basic gates and can be used for the logic synthesis of any circuit. Each gate is simulated through SPICE with 10k Monte Carlo samples. Fig. 10 shows the pdf and the cumulative distribution function (cdf) of the output node capacitance of NAND2X1. The pdf from the proposed model matches closely with the Monte Carlo simulations and the match for the cdf is even better.

To quantize the influence of output load, every gate with a different number of fan-outs is modeled by regression through

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

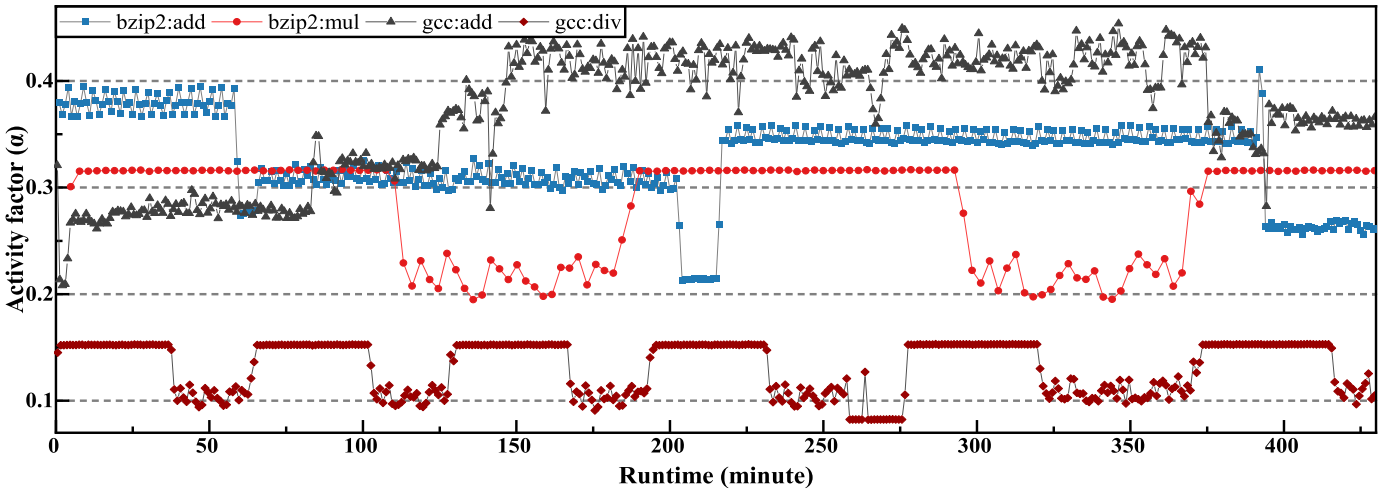


Fig. 12. Activity factors of the instruction add/mul/div.

Monte Carlo simulations. Five kinds of fan-out numbers are considered, which are FO1, FO2, FO4, FO8, and FO16. The load capacitance distribution of each gate is represented by the mean μ_{L} and the standard deviation σ_{L}. The relative errors of predicted μ_{L} and σ_{L} are calculated with the following equation:

$$\text{error} = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{abs}(\text{Sim}_i - \text{Pred}_i)}{\text{Sim}_i} \times 100\%. \tag{15}$$

The average accuracies of estimating load capacitances of all gates are shown in Fig. 11. The X-axis represents the number of fan-outs and the Y-axis represents the average errors of all gates with the corresponding fan-out. The gates are listed in the beginning of Section V-B. The average relative errors of the mean and standard deviation of the load capacitance model are 1.1% and 1.4%, respectively. With the increasing number of fan-out gates, the mean errors are relatively stable (around 1%) compared with standard deviation errors (less than 2%). This indicates that the load capacitance can be well-approximated by the proposed method. For a given technology node, the time cost of Monte Carlo simulations for getting the training set is 9.55 h on average. The proposed gate model is uniquely defined by the fitting coefficients and can be applied to different designs in the same technology node. Also, in all cases, the coefficients are estimated using MATLAB and merely take minutes, which is considered to be negligible overhead.

C. Dynamic Power Analysis

The analysis is based on PULPino [12], an open-source RISC-V microprocessor. At the architecture level, SPEC 2006 benchmarks are simulated using gem5 simulator [31]. Taking bzip2 and gcc for demonstration, Fig. 12 shows the activity factors of different instruction types calculated using our method. The X-axis shows the runtime of benchmarks in minutes. For the same instruction type, i.e., add, the activity factors behave differently with benchmarks and time periods. For different instruction types, i.e., mul and div, they also show very diverse activity factors. These activity factors are estimated by tracking real-time instructions from gem5. In contrast, the power from PTPX is calculated with VCD files that store circuit switching activities generated from extra RTL simulations. PTPX reads a VCD file, sets gate-level states of toggled pins, and logically propagates the toggled gates to find out the corresponding activity factor. The interaction between PTPX and RTL simulator is slow and these VCD files require a huge space for storage. In our work, without the need of saving files, the switching activities are obtained by analyzing the instruction sequences from gem5 and automatically fed into the modified McPAT using data streaming. Therefore, the calculation process is faster and more flexible than activity tracking by PTPX. Moreover, the process is more accurate than McPAT, which uses an empirical constant factor regardless of the detailed circuit information.

Fig. 13(a) shows the activity factor (α) of h264ref (right Y-axis) and the remaining part of (13) before multiplying α (left Y-axis), i.e., $C_{\text{total}} \cdot V_{\text{dd}}^2 \cdot f$. The impact of process variations is exhibited by the variation in $C_{\text{total}} \cdot V_{\text{dd}}^2 \cdot f$ between the best case ($\mu - 3\sigma$) and the worst case ($\mu + 3\sigma$). The worst case shows 1.3× higher than the mean case on average and 1.8× higher than the best case. Under the circumstances, the power of a chip with best case process variations can be similar to another chip with the worst, taking the 2.5 billion cycles as an example. In addition, α and $C_{\text{total}} \cdot V_{\text{dd}}^2 \cdot f$ show very different phase behavior during the simulation. α is mainly determined by the toggled gates of functional units, while $C_{\text{total}} \cdot V_{\text{dd}}^2 \cdot f$ is mainly determined by the types of functional units. In Fig. 13(a), $C_{\text{total}} \cdot V_{\text{dd}}^2 \cdot f$ is entirely different between 2 and 3 billion cycles, demonstrating that the main type of executed functional unit is changed. Despite that, α can still maintain the same between different units.

Based on the activity factors, the dynamic power of each functional unit is calculated in the modified McPAT. Based on the BSIM4 device model, the SPICE simulations are regarded as golden results. In addition, Fig. 13(b) shows the worst case power estimations ($\mu + 3\sigma$) compared with PTPX, PRIMAL [18], and McPAT for 24 billion cycles. The

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

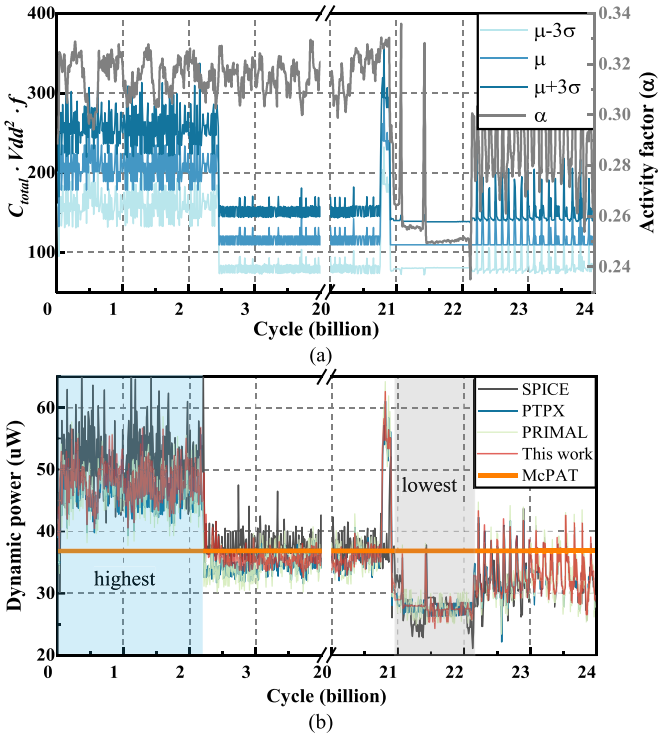


Fig. 13. (a) Result of $C_{\text{total}} \cdot V_{\text{dd}}^2 \cdot f$ and α of h264ref. (b) Cycle-by-cycle dynamic power analysis results of h264ref compared with SPICE, PTPX, and McPAT.

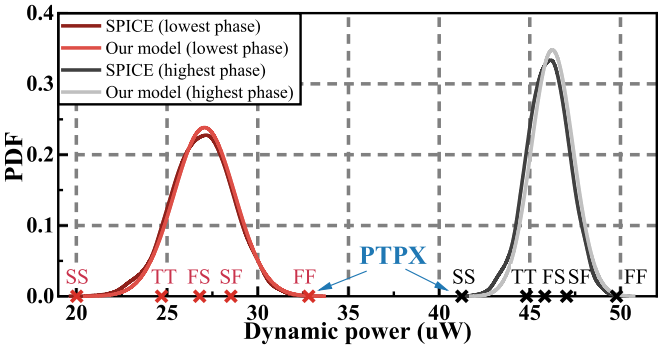


Fig. 14. Power distribution comparison between our model, SPICE simulations, and PTPX simulations on five process corners (h264ref).

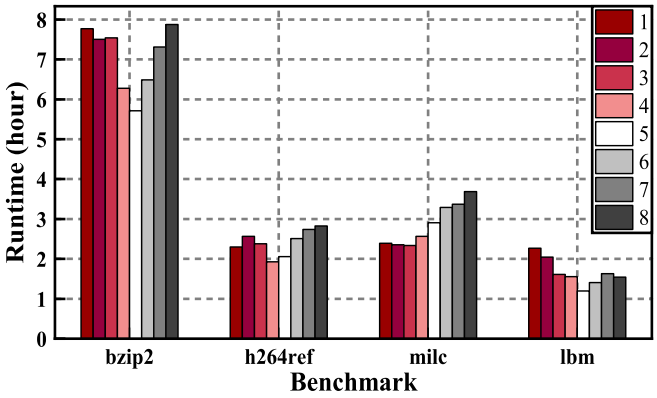


Fig. 15. Runtime of different benchmarks with different number of executors (1 billion instructions).

phase behavior of h264ref is determined by both the toggled gates and the type of functional units. The highest power reaches 60 μW, while the lowest power drops to nearly 25 μW. The highest power phase maintains 2.1 billion cycles at the beginning and the lowest power phase maintains almost 1 billion cycles. Compared with the dynamic power from SPICE, the power based on our model with a 5.4% error is sufficiently precise, while PTPX shows an average error of 7.3% compared with SPICE. The PRIMAL-based method losses accuracy without considering the impact of process variations. The error of PRIMAL is about 10.8% compared with SPICE on average. The dynamic power using McPAT is pessimistic (the error is larger than 30%) because it ignores the changing phases of the activity factors during execution. Since the simulation time to gather dynamic power is prohibitive, it is clear that cycle-accurate dynamic power measurements are crucial to future power-aware researches.

The variability of dynamic power per cycle due to process variations is also collected during the evaluation process. Given an example of h264ref, two groups of power distributions are illustrated in Fig. 14, separated from the lowest power phase and the highest power phase in Fig. 13. The mean of the dynamic power of the left group is 27.1 μW and the standard deviation is 1.7 μW, whereas the mean of the right group is 46.2 μW and the standard deviation is 1.1 μW. The distribution can also be roughly represented by five corners (FF, SS, TT, FS, and SF) from PTPX, as shown at the bottom of Fig. 14. For worse case power estimation, the result from PTPX is close to SPICE only when the corner is FF but still loses accuracy compared with our model. In addition, the ratio ((standard deviation)/mean) from the lowest power phase is 1.5× times larger than that from the highest power phase, which further emphasizes the importance of accounting for the process variation effect in the low-power scenario.

D. Streaming-Based Analysis Method

The evaluation can speed up with the implementation of Spark, benefitting from the multi-core and data streaming mechanisms. Specifically, the worker node of Spark contains several executors and each executor is assigned to each action (each RDD in Algorithm 1). The number of executors can be specified with the flag –executor-cores when invoking spark shell from the command line. For example, –executor-cores *n* means that *n* executors are allocated to *n* cores so that they can run at the same time. Fig. 15 shows the Spark execution time of each benchmark with the different number of executors (from 1 to 8) when running 1 billion instructions. When the number of executors is set to 4, the execution time reaches the lowest, because the evaluation framework contains mainly four actions. Each action can be allocated onto a core at the same time when the resources are sufficient.

By dividing each executor into several partitions, the data stream from each executor is then divided, so that the action for the whole stream can be distributed to different threads. That is, input vectors from gem5 are divided into several

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

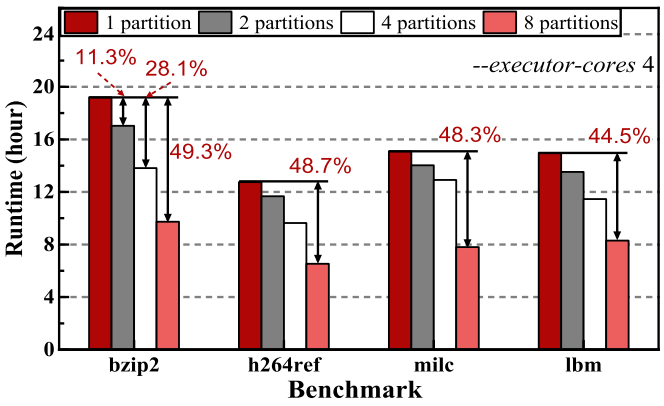


Fig. 16. Runtime of the proposed streaming-based framework with different partitions (1 billion instructions).

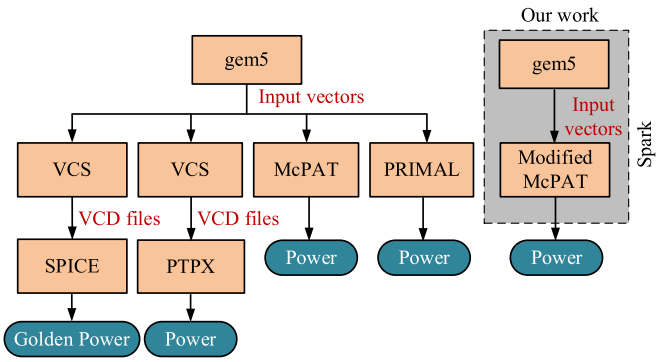


Fig. 17. Dynamic power analysis flows of SPICE, PTPX, McPAT, PRIMAL, and our work.

streams and each stream is mapped to calculate the effective capacitance in parallel. Fig. 16 shows the runtime comparison for 1 billion instructions of different benchmarks with the number of partitions 1, 2, 4, and 8. The total runtime decreases when the amount of partitions is increased. The results show that the streaming-based framework has a 46.1% speedup in eight partitions compared with the single partition on average.

To compare the time overhead, Fig. 17 depicts the analysis flows of the traditional PTPX-based method, the McPAT-based method, the PRIMAL-based method, and our method. For commercial tool PTPX, the dynamic power analysis is realized using VCD files from the RTL simulator VCS [30]. The inputs of VCS are input vectors of each cycle and are obtained from gem5 simulation. The second comparison is using input vectors from gem5 and then transferring them to the modified McPAT for cycle-accurate power analysis. The input vectors need to be saved into a file as McPAT is another external simulator. This is quite time-consuming and requires high memory storage. PRIMAL trains the convolutional neural network (CNN) models with verification testbenches. The trained models can be used to generate detailed power profiles under different workloads, simulated by gem5.

Fig. 18 shows the comparison of the time consumption of those methods. The remaining three comparisons are our streaming-based method with different numbers of executors and partitions. Without the need for storing and loading,

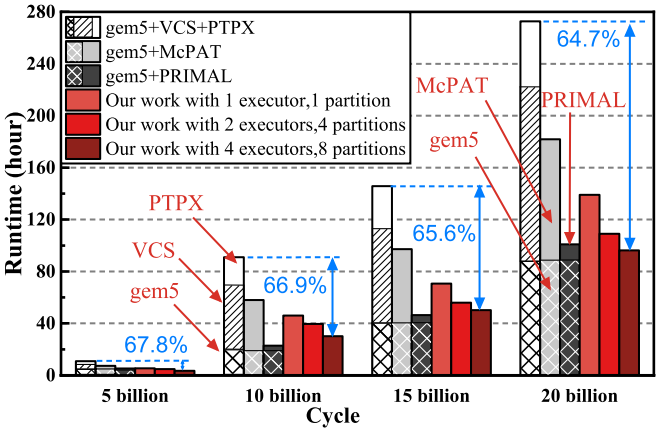


Fig. 18. Time consumption of dynamic power analysis between PTPX, McPAT, and our work (gcc).

the input vectors are converted into streams and piped from gem5 to McPAT automatically and simultaneously. Experiments show that the runtime with 20 billion cycles takes 272, 181, 99, and 86 h for gem5+VCS+PTPX, gem5+McPAT, gem5+PRIMAL, and our framework with four executors and eight partitions, respectively. Through allocating resources in the data streaming engine, the dynamic power can be almost immediately calculated with the input vector that is just generated by gem5. The performance improvements come from avoiding disk I/O reading and writing overhead. These results indicate that our method is 48.8% faster on average than gem5+McPAT simulations without Spark because of the huge file transmission overhead. Besides, the streaming-based power analysis is 65.2% faster than PTPX. Groups of input vectors from gem5 can be parallelly fed into McPAT in our work, while the process in PTPX only can be done sequentially. Our method shows a similar speed to gem5+PRIMAL. The speed of our model depends on the circuit complexity of functional units. Conversely, the speed of the CNN model in PRIMAL depends on the complexity of neural networks. It proves that our methodology is more accurate and faster compared with others.

VI. Conclusion

In this article, we build a bottom-up streaming-based method that calculates the dynamic power consumption of a processor. It first uses gem5 to collect input vectors of functional units and dumps them in the form of data streams. The streams are caught using Spark Streaming to calculate the activity factor of each vector. The activity factor is calculated by estimating the effective capacitance, which is estimated by gate-level capacitance models and the input vector of the target circuit. The cycle-accurate dynamic power is estimated through modified McPAT. The overall power evaluation framework is realized through multithreaded implementation of Spark. Compared with using dynamic power results from SPICE simulations, our proposed method achieves about 95.6% accuracy, while the accuracy of the commercial tool PTPX is less than 93.7%. Furthermore, in the four-executor

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.

eight-partition mode, our approach is 65.2% faster than PTPX and is 48.8% faster than the traditional architecture-level simulators.

In future work, the overall bottom-up evaluation flow can be improved from the architecture level to the circuit level. At the architecture level, more macroblocks can be expanded, such as caches and buffers. At the circuit level, the model for the leakage power and the short-circuit power can be improved and added, considering the low-voltage effects and process variation effects.

REFERENCES

[1] S. P. Mohanty, N. Ranganathan, E. Kougianos, and P. Patra, *Low-Power High-Level Synthesis for Nanoscale CMOS Circuits*. New York, NY, USA: Springer, 2008.

[2] D. Stamoulis and D. Marculescu, "Can we guarantee performance requirements under workload and process variations?" in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2016, pp. 308–313.

[3] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," *J. Big Data*, vol. 2, no. 1, pp. 1–20, Dec. 2015.

[4] A. Lukefahr *et al.*, "Composite cores: Pushing heterogeneity into a core," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp. 317–328.

[5] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies," *ACM SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 51–62, Jun. 2008.

[6] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture (Micro)*, Dec. 2009, pp. 469–480.

[7] S. K. Rethinagiri, O. Palomar, R. B. Atitallah, S. Niar, O. Unsal, and A. C. Kestelman, "System-level power estimation tool for embedded processor based platforms," in *Proc. 6th Workshop Rapid Simulation Perform. Eval. Methods Tools (RAPIDO)*, 2014, pp. 1–8.

[8] S. L. Xi, H. Jacobson, P. Bose, G.-Y. Wei, and D. Brooks, "Quantifying sources of error in McPAT and potential impacts on architectural studies," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2015, pp. 577–589.

[9] D. Trilla, C. Hernandez, J. Abella, and F. J. Cazorla, "Modeling the impact of process variations in worst-case energy consumption estimation," in *Proc. 22nd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2019, pp. 601–605.

[10] J. Pallister, S. Kerrison, J. Morse, and K. Eder, "Data dependent energy modeling for worst case energy consumption analysis," in *Proc. 20th Int. Workshop Softw. Compil. Embedded Syst.*, Jun. 2017, pp. 51–59.

[11] *PrimeTime-PX User Guide, Version 2008.12*, Synopsys, Mountain View, CA, USA, 2008.

[12] A. Traber *et al.*, "PULPino: A small single-core RISC-V SoC," in *Proc. 3rd RISCV Workshop*, 2016.

[13] A. Tang, Y. Yang, C.-Y. Lee, and N. K. Jha, "McPAT-PVT: Delay and power modeling framework for FinFET processor architectures under PVT variations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1616–1627, Sep. 2015.

[14] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proc. 22nd Digit. Avionics Syst. Conf.*, 2003, pp. 93–104.

[15] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," in *Proc. 24th ACM Int. Conf. Supercomput. (ICS)*, 2010, pp. 147–158.

[16] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, May 2011.

[17] W. Fu, M. Ling, W. Wang, and L. Shi, "AMPS: Accelerating McPAT power evaluation without cycle-accurate simulations," *IEEE Embedded Syst. Lett.*, vol. 12, no. 1, pp. 13–16, Mar. 2020.

[18] Y. Zhou, H. Ren, Y. Zhang, B. Keller, B. Khailany, and Z. Zhang, "PRIMAL: Power inference using machine learning," in *Proc. 56th Annu. Design Automat. Conf.*, Jun. 2019, pp. 1–6.

[19] Y. Zhang, H. Ren, and B. Khailany, "GRANNITE: Graph neural network inference for transferable power estimation," in *Proc. 57th ACM/IEEE Design Automat. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[20] (2016). *Apache Hadoop: Welcome to Apache Hadoop!* Accessed: Nov. 1, 2016. [Online]. Available: http://hadoop.apache.org/

[21] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online," in *Proc. NSDI*, vol. 10, no. 4, 2010, p. 20.

[22] T.-W. Huang and M. D. F. Wong, "On fast timing closure: Speeding up incremental path-based timing analysis with MapReduce," in *Proc. ACM/IEEE Int. Workshop Syst. Level Interconnect Prediction (SLIP)*, Jun. 2015, pp. 1–6.

[23] T.-W. Huang and M. D. F. Wong, "Accelerated path-based timing analysis with MapReduce," in *Proc. Symp. Int. Symp. Phys. Design*, Mar. 2015, pp. 103–110.

[24] (2016). *Apache SPARK: Apache Spark?* Accessed: Nov. 1, 2016. [Online]. Available: http://spark.apache.org/

[25] I. H. Abbassi, F. Khalid, O. Hasan, A. M. Kamboh, and M. Shafique, "McSeVIC: A model checking based framework for security vulnerability analysis of integrated circuits," *IEEE Access*, vol. 6, pp. 32240–32257, Jun. 2018.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer-Verlag, 2009.

[27] N. H. E. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. London, U.K.: Pearson, 2015.

[28] S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, "Variation-tolerant dynamic power management at the system-level," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 9, pp. 1220–1232, Sep. 2009.

[29] S. Chandra *et al.*, "Variation-aware system-level power analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1173–1184, Aug. 2010.

[30] (2004). *Synopsys VCS Verilog Simulator*. [Online]. Available: http://www.synopsys.com/products/simulation/simulation.html

[31] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.

[32] V. Czitrom and P. D. Spagon, *Statistical Case Studies for Industrial Process Improvement*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997.

[33] I. H. Abbasi *et al.*, "Formal verification of gate-level multiple side channel parameters to detect hardware Trojans," in *Proc. Int. Workshop Formal Techn. Saf.-Crit. Syst.* Cham, Switzerland: Springer, 2016, pp. 75–92.

[34] S. Schuermans and R. Leupers, *Power Estimation on Electronic System Level Using Linear Power Models*. Cham, Switzerland: Springer, 2019.

[35] W. Fu, L. Jin, M. Ling, Y. Zheng, and L. Shi, "A cross-layer power and timing evaluation method for wide voltage scaling," in *Proc. 57th ACM/IEEE Design Automat. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[36] *HSPICE MOSFET Models Manual, Version X-2005.09*, Synopsys, Mountain View, CA, USA, Sep. 2005.

[37] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEICE Trans. Electron.*, vol. 75, no. 4, pp. 371–382, 1992.

[38] M. M. Maryan, M. Amini-Valashani, and S. J. Azhari, "A new circuit-level technique for leakage and short-circuit power reduction of static logic gates in 22-nm CMOS technology," *Circuits, Syst., Signal Process.*, vol. 40, no. 7, pp. 3536–3560, Jul. 2021.

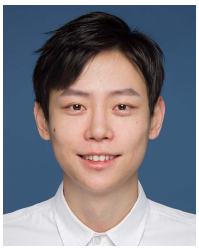
[39] K. Nose and T. Sakurai, "Analysis and future trend of short-circuit power," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 9, pp. 1023–1030, Sep. 2000.



Leilei Jin received the B.S. degree in electronic information engineering from China University of Petroleum, Qingdao, China, in 2018. She is currently working toward the Ph.D. degree in electrical engineering with Southeast University, Nanjing, China.

Her main research interests include multilevel partitioning method and statistical timing analysis in low-voltage circuit designs.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.



Wenjie Fu received the B.S. degree in microelectronics from Wenzheng College, Soochow University, Suzhou, China, in 2016. He is currently working toward the Ph.D. degree in electrical engineering with Southeast University, Nanjing, China.

His main research interests include performance evaluation in computer architecture and statistical static timing analysis in low-voltage circuit designs.



Ming Ling (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Southeast University, Nanjing, China, in 1994, 2001, and 2011, respectively.

He is currently working as an Associate Professor with the National ASIC System Engineering Technology Research Center, Southeast University. His main research interests include memory subsystem of system-on-chip (SoC), embedded software, and SoC architecture.



Longxing Shi (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Southeast University, Nanjing, China, in 1984, 1987, and 1992, respectively.

He is currently a Professor and the Dean of the National ASIC System Engineering Research Center, Southeast University. His research interests include system-on-a-chip design, VLSI design, and power IC design.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on November 22,2026 at 13:01:22 UTC from IEEE Xplore. Restrictions apply.