Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Algorithm and evaluation of generating pseudo-datasets for integrated circuit power analysis

Zejia LYU[1], Jizhong SHEN[†‡1], Xi CHEN[2]

[1]*College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China*

[2]*Shanghai Hexin Industrial Software Co., Ltd., Shanghai 201210, China*

[†]E-mail: jzshen@zju.edu.cn

Received Aug. 2, 2024; Revision accepted Mar. 7, 2025; Crosschecked July 21, 2025; Published online Sept. 12, 2025

Abstract: Average power analysis plays a crucial role in the design of large-scale digital integrated circuits (ICs). The integration of data-driven machine learning (ML) methods into the electronic design automation (EDA) fields has increased the demand for extensive datasets. To address this need, we propose a novel pseudo-circuit generation algorithm rooted in graph topology. This algorithm efficiently produces a multitude of power analysis examples by converting randomly generated directed acyclic graphs (DAGs) into gate-level Verilog pseudo-combinational circuit netlists. The subsequent introduction of register units transforms pseudo-combinational netlists into pseudo-sequential circuit netlists. Hyperparameters facilitate the control of circuit topology, while appropriate sequential constraints are applied during synthesis to yield a pseudo-circuit dataset. We evaluate our approach using the mainstream power analysis software, conducting pre-layout average power tests on the generated circuits, comparing their performance against benchmark datasets, and verifying the results through circuit topology complexity analysis and static timing analysis (STA). The results confirm the effectiveness of the dataset, and demonstrate the operational efficiency and robustness of the algorithm, underscoring its research value.

Key words: Graph computation; Electronic design automation (EDA); Pseudo-dataset; Average power analysis
https://doi.org/10.1631/FITEE.2400677 **CLC number:** TP391.7

1 Introduction

Power analysis plays a pivotal role in the design of large-scale digital integrated circuits (ICs), where accurate and efficient power evaluation during the design phase can substantially improve chip performance. In the industry, power analysis typically refers to average power estimation facilitated by the electronic design automation (EDA) software. Notable examples of commercial software for this purpose include Synopsys PrimeTime PX (PTPX) and Cadence Voltus. Typically, average power analysis is performed at the gate level, where power is calculated based on circuit activity.

Circuit activity can be categorized into annotated and vectorless types. The annotated method uses simulators like Synopsys Verilog compiled simulator (VCS) with predefined test vectors to monitor activity at each circuit node, typically recorded in switching activity interchange format (SAIF) files. Although the simulation-based annotated method offers high accuracy, it is time-consuming and demands significant storage space. Consequently, the industry often employs a vectorless method in the pre-simulation phase, which uses a mathematical model to estimate the activity probability at each node and then computes power consumption.

The concept of probability-based power estimation was first introduced by Najm (1993), and has since spurred extensive research within the academic community on such theoretical models

‡ Corresponding author
ORCID: Zejia LYU, https://orcid.org/0009-0009-1019-5124; Jizhong SHEN, https://orcid.org/0000-0002-9031-2379

(Burch et al., 1992; Xie, 2023). These models are used solely for algorithm validation, and do not require large case sizes. However, as circuit complexity escalates, traditional algorithms struggle with efficiency and accuracy, particularly in large-scale combinational and sequential circuits. This challenge has led to a growing interest in data-driven machine learning (ML) approaches. Several researchers have explored innovative methods; for instance, Zhou Y et al. (2019) introduced power inference using machine learning (PRIMAL), using convolutional networks for power estimation, while Kumar and Gerstlauer (2019) adopted a decision tree approach. Others like Zhang et al. (2020), Rakesh et al. (2023), and Khan et al. (2024) employed graph neural networks to represent circuit topology for power prediction, while Li et al. (2022), Fang et al. (2023), and Kumar et al. (2023) used traditional neural networks. In addition to the mentioned power prediction, there are circuit power optimization methods, such as the ML-based circuit power optimization proposed by Zou et al. (2024).

Despite the advancements in power analysis techniques, the requirement for extensive circuit data and diverse topological scenarios remains a significant challenge for model training (Nasser et al., 2021). Current standard datasets in the IC industry are often inadequate for ML applications, lacking both quantity and specificity (Brglez et al., 1989; Hansen et al., 1999). Moreover, accessing large-scale industrial or competition data frequently entails navigating copyright and other legal constraints (Xie et al., 2023a, 2023b), posing substantial barriers to data acquisition.

In response to the limitations of existing datasets, to support the needs of ML-driven average power analysis methods, we propose a novel pseudo-circuit dataset generation algorithm specifically tailored for power analysis, as depicted in Fig. 1. Our methodology leverages graph topology calculation to generate pseudo-circuit datasets. Initially, a random directed acyclic graph (DAG) is created, followed by mapping DAG nodes to gate-level nodes to form a combinational circuit. By integrating register units, these combinational circuits are converted into sequential circuits, which are subsequently optimized using the Synopsys design compiler (DC) under defined constraints. The accuracy of circuit activity is verified through golden power analysis from Synopsys VCS simulation with random vectors. This approach allows for the generation of an unlimited number of dataset samples without relying on open-source register–transfer level (RTL) code. In random vector tests on sequential circuits, we observe a signal non-flipping phenomenon due to loop impacts, which results in low average power consumption. The signal non-flipping phenomenon, termed the zero power issue, is further explored to understand the influence of sequential circuit topology on power consumption.

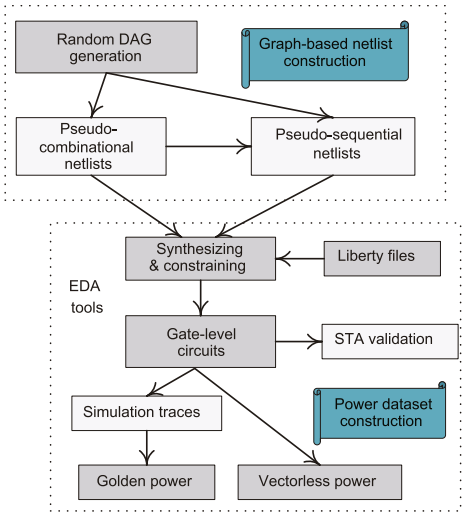

Fig. 1 Overview of the pseudo-circuit dataset generation process for pre-simulation power analysis

2 Combinational circuit netlists and DAG topology

Combinational circuits are characterized by steady-state outputs that depend solely on the current input values and not on any previous states (Kaeslin, 2008). The output nodes are set as F_i for $i = 1, 2, \ldots, n$, and the input nodes as x_j for $j = 1, 2, \ldots, m$, with the relationship $F_i = f(x_1, x_2, \ldots, x_m)$. Here, $f(\cdot)$ symbolizes the function encapsulated by the combinational logic circuit, and n and m are the numbers of output and input nodes, respectively.

The architectural elements of the circuit—AND, OR, NOT gates, and other combinational logic units are abstracted as nodes $V = \{v_1, v_2, \ldots, v_M\}$. The connections between inputs, outputs, and these nodes are represented as directed edges $E = \{e_1, e_2, \ldots, e_N\}$. The circuit topology is typically a DAG with node set V and edge set E, i.e.,

$G=(V,E)$. Given the inherent DAG-like properties of combinational circuits, it is feasible to initially construct a DAG and subsequently adapt it into a combinational circuit configuration.

2.1 Random DAG generation algorithm

In scenarios requiring specific numbers of inputs and outputs, traditional brute force methods for generating random DAGs are often suboptimal due to inefficiencies and performance limitations stemming from the random structures that fail to meet circuit specifications. We propose an efficient and rapid algorithm for generating random DAG topologies that adhere to the following constraints:

1. m inputs, n outputs;

2. A topology consistent with a DAG and comprising a single-connected component.

We define the sets of primary input nodes as $\mathcal{X}$, internal interconnection nodes as $\mathcal{W}$, and primary output nodes as $\mathcal{O}$. Here, $\mathcal{W}$ and $\mathcal{O}$ are collectively referred to as output node set $\mathcal{O}'$. The initial state of the graph $\mathcal{G}$ includes only nodes without any connecting edges. We introduce the random multi-connected component DAG generation method in Algorithm 1, which efficiently establishes initial connections within $\mathcal{G}$. $\langle p,o\rangle$ denotes the graph edge, referring to $p\rightarrow o$. This algorithm ensures that each component of $\mathcal{G}$ is a subgraph conforming to the structure of a DAG. γ represents the expectation of the number of nodes, and is used to modulate the complexity of the resultant topology, with higher values indicating the increased complexity.

During the random selection of input nodes, the size of set $\mathcal{X}$ is approximately m. However, the size of set $\mathcal{O}$ can substantially deviate from n. This discrepancy arises as edges that do not adhere to the DAG structure are eliminated. This process may also lead to the formation of multi-connected components within the computational graph $\mathcal{G}_{\mathrm{M}}$. To resolve these potential problems, we introduce Algorithm 2.

Algorithm 2 enhances the graph's structure by examining and connecting separate components, ensuring that the graph maintains weak connectivity and forms a single-connected component. Post-processing with Algorithm 2 results in a single-connected computational graph $\mathcal{G}_{\mathrm{S}}$. Together, Algorithms 1 and 2 guarantee that $\mathcal{G}_{\mathrm{S}}$ adheres to the necessary topological specifications. Nonetheless, the primary output node set ($\mathcal{O}_{\mathrm{S}}$) in $\mathcal{G}_{\mathrm{S}}$ may not align with the expectation. To mitigate this issue, we introduce Algorithm 3 that augments the topology on the existing $\mathcal{O}_{\mathrm{S}}$, thus minimizing the difference in the

Algorithm 1 Random multi-connected component DAG generation method

Require: Graph $\mathcal{G}$ defined only by nodes without edges
Ensure: Multi-connected DAG graph $\mathcal{G}_{\mathrm{M}}$

$\mathcal{S}\leftarrow\mathcal{W}+\mathcal{X}$
for all $o\in\mathcal{O}'$ **do**
  Randomly select γ nodes from $\mathcal{S}$ as parent nodes for o, forming the set $\mathcal{P}$
  for all $p\in\mathcal{P}$ **do**
    Add edge $\langle p,o\rangle$ to $\mathcal{G}$
    if $\mathcal{G}$ does not conform to DAG structure **then**
      Remove edge $\langle p,o\rangle$
    end if
    Add p to $\mathcal{S}$
  end for
end for

Algorithm 2 Single-connected component assurance

Require: $\mathcal{G}_{\mathrm{M}}$
Ensure: $\mathcal{G}_{\mathrm{S}}$

$\mathcal{G}_{\mathrm{S}}=\mathcal{G}_{\mathrm{M}}$
while $\mathcal{G}_{\mathrm{S}}$ is not weakly connected **do**
  List all connected components of $\mathcal{G}_{\mathrm{S}}$
  Select different nodes from two different components
  Connect these two nodes
  Update $\mathcal{G}_{\mathrm{S}}$
end while

Algorithm 3 Adjustment of the primary output nodes

Require: $\mathcal{G}_{\mathrm{S}}$ and the expected output node number n
Ensure: Computational graph $\mathcal{G}$ with adjusted output

Let $\mathcal{G}=\mathcal{G}_{\mathrm{S}}$
while true **do**
  $\mathcal{O}_{\mathrm{S}}\leftarrow\mathcal{G}$
  $\mathcal{O}_{\mathrm{S}}^{+},\mathcal{O}_{\mathrm{S}}^{-}\leftarrow\mathcal{O}_{\mathrm{S}}$
  for all $\sigma\in\mathcal{O}_{\mathrm{S}}^{+}$ **do**
    Select nodes from $\mathcal{O}_{\mathrm{S}}^{-}$ as parent $\mathcal{P}_{\mathrm{S}}$ for σ
    for $p\in\mathcal{P}_{\mathrm{S}}$ **do**
      Add edge $\langle p,\sigma\rangle$ to $\mathcal{G}$
    end for
  end for
  if the number of output nodes in $\mathcal{G}$ approximates n **then**
    Break loop
  end if
end while

number of primary input nodes.

Algorithm 3 segregates the output nodes into two categories: retained node set ($\mathcal{O}_{\mathrm{S}}^{+}$) and discarded node set ($\mathcal{O}_{\mathrm{S}}^{-}$). The algorithm subsequently appoints nodes from $\mathcal{O}_{\mathrm{S}}^{-}$ as progenitors for the nodes in $\mathcal{O}_{\mathrm{S}}^{+}$. This involves a stochastic selection of parent nodes from $\mathcal{O}_{\mathrm{S}}^{-}$ for $\mathcal{O}_{\mathrm{S}}^{+}$, which may not ensure a precise match to the desired number of output nodes, n. Consequently, multiple iterations may be required to mitigate the error within an acceptable threshold. Post-processing with Algorithm 3, the resultant computational graph $\mathcal{G}$ forms a single-connected DAG, which is suitable for implementation in combinational circuits.

2.2 From DAG to combinational circuit netlist

Upon deriving the abstract topology represented by $\mathcal{G}$, we convert this computational graph into a pseudo-combinational circuit. The conversion follows these rules:

1. Nodes are initially categorized based on their indegree and outdegree into three distinct types, primary inputs, wires, and primary outputs. Nodes with an indegree of zero are primary inputs, those with an outdegree of zero are primary outputs, and all remaining nodes are wires.

2. For nodes categorized as wires, gate units are assigned variably based on the indegree size. Specifically, if the indegree is one, a gate unit is randomly assigned either as NOT or BUF. If the indegree is one, the gate unit type is randomly selected from AND, NAND, OR, or NOR.

As shown in Fig. 2, the methodology for generating a pseudo-random combinational circuit netlist from a random DAG is delineated. The process begins with the generation of nodes based on the predefined number of primary input and output nodes. Using Algorithm 1, these nodes are interconnected to form a multi-connected DAG structure, denoted as $\mathcal{G}_{\mathrm{M}}$. To integrate multi-connected components into a coherent structure, Algorithm 1 is again utilized, resulting in a unified DAG, $\mathcal{G}_{\mathrm{S}}$. To avoid $\mathcal{G}_{\mathrm{S}}$ failing to align with the primary output node count n, adjustments are made using Algorithm 3, thereby achieving a final computational graph $\mathcal{G}$ that satisfies the specified requirements. The graph node is then converted into gate units at random to convert graph into the desired random combinational circuit netlist.

3 Sequential circuits and DCG topology

Sequential circuits are defined as circuits whose steady-state output at any given time depends not only on the current input but also on the input from previous states (Kaeslin, 2008). Let the primary

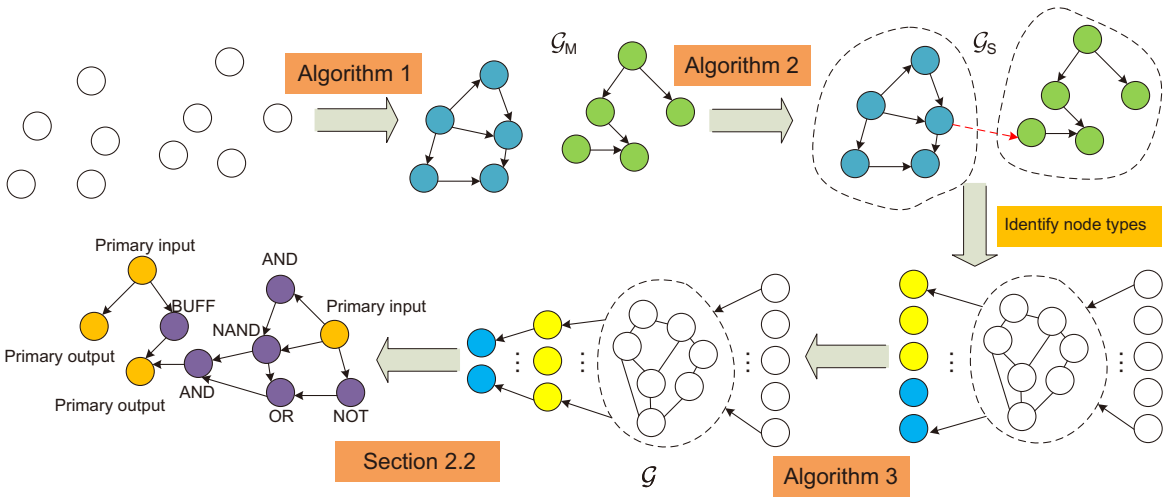

Fig. 2 Random combinational circuit netlist generation. Algorithm 1 processes the nodes into different connected components, Algorithm 2 transforms multi-connected components into a single-connected component, and Algorithm 3 adjusts the number of identified output pins. Finally, the generated DAG topology is converted into the combinational circuit netlist in Section 2.2

input nodes be denoted by x_i, $i = 1, 2, \ldots, m$, the output state equation by $f_j(\cdot)$, $j = 1, 2, \ldots, n$, and primary output nodes by y_j. Additionally, define the drive excitation function as $g_r(\cdot)$, $r = 1, 2, \ldots, k$, the drive excitation as z_r, the signal state function as $h_o(\cdot)$, $o = 1, 2, \ldots, l$, and the signal state as q_o. k and l are the numbers of drive excitation functions and signal state functions, respectively. The output logic for sequential circuits can thus be expressed through the following equations:

$$y_j = f_j\left(x_1, x_2, \ldots, x_m, q_1, q_2, \ldots, q_l\right), \quad (1)$$

$$z_r = g_r\left(x_1, x_2, \ldots, x_m, q_1, q_2, \ldots, q_l\right), \quad (2)$$

$$q_o^{t+1} = h_o\left(z_1, z_2, \ldots, z_k, q_1^t, q_2^t, \ldots, q_l^t\right), \quad (3)$$

where t denotes discrete time step indexed by clock periods.

Sequential circuits, unlike combinational circuits, incorporate a state storage structure consisting of registers and combinational logic arranged in a loop. Our research focuses on sequential circuits that adhere to a directed cycle graph (DCG) structure. Directly generating a DCG structure into a practical circuit netlist can be challenging. We propose an efficient and swift method that leverages gate-level combinational circuits to generate a pseudo-sequential circuit netlist in compliance with a DCG structure.

The transformation from a combinational circuit netlist to a sequential circuit netlist can be facilitated by introducing registers. This method involves selecting a specific number of primary inputs and outputs from the existing gate-level combinational circuit and connecting them using predefined gate-level registers. This process effectively transforms the DAG combinational circuit into a DCG-structure pseudo-sequential circuit netlist. It is crucial that the descriptions of these gate-level registers align with the logic descriptions used in the DC synthesis phase.

The ratio of the number of the selected input/output (IO) ports to the number of DAG IO ports is defined as α, $\alpha = l/m$, which is called the sequential circuit generation factor, and the functional trend of the generated circuits can be changed by varying this factor. For example, a larger α implies that there are more registers in the circuit, which may correspond to the memory function chip, while a smaller α corresponds to the computation function chip.

4 Simulation results

4.1 Dataset generation flow and parameter setting

The proposed algorithm can generate pseudo-circuit Verilog netlists that contain gate-level unit descriptions. These netlists are random in function and do not have strict specific practical application scenarios or functional meanings. Pseudo-netlists are mainly used in the field of power analysis, and this subsection will start with the synthesis and power consumption testing of the generated pseudo-netlists.

For synthesis, we use the generated pseudo-gate-level netlist, applying it to the Semiconductor Manufacturing International Corporation (SMIC) 40-nm digital circuit process library. We employ various components, such as AND, NAND, OR, NOR, BUF, INV, and DFF. Appropriate timing constraints are set to facilitate the synthesis of the gate-level Verilog descriptions into a gate-level circuit netlist based on the industrial process library.

The randomly-generated pseudo-gate-level netlist may exhibit redundancies in logic functions or issues like logic constants stemming from nested circuit designs. It is essential to meticulously analyze the internal logic of circuits to eliminate any redundant functionality. The DC offers methods for checking and correcting the logic functions and timing of circuits. The synthesis process with DC aims not only to align the circuits with process libraries, but also to optimize the circuit design and ensure the integrity of the generated pseudo-circuits.

The number of IO nodes in combinational circuits is randomly selected between 50 and 400, with the number of wires varying from 500 to 5000. For sequential circuits, the initial number of IO nodes ranges from 100 to 600, with the number of wires also between 500 and 5000. A parameter setting of $\gamma = 2$ is applied. The pseudo-combinational circuits' netlist for generating the DAG structure uses a conversion factor of $\alpha = 0.3$ to transition to a pseudo-sequential circuit. During the DC synthesis, the software would adjust the IO ports and wires based on timing constraints.

4.2 Pseudo-power datasets

As per the algorithm discussed earlier, 1000 sets of combinational and 1000 sets of sequential circuit netlists are generated. Then DC is used to generate the pseudo-dataset, as depicted in Table 1.

The dataset comprises samples where the number of logic gates in the combinational circuits is within a range from 900 to 5700 with the average of approximately 3300. The number of primary IO nodes in these circuits varies from 20 to 80. In sequential circuits, the gate count ranges from 1100 to 7400, averaging about 4900 gates. The number of primary IO nodes ranges from 90 to 430. Notably, the generated netlists maintain consistency in the numbers of primary inputs and outputs, exhibiting a broad diversity in the numbers of output ports and internal gate-level units. Variations between the pseudo-circuit configurations and pseudo-topology parameters arise due to corrections and simplifications made during the synthesis, specifically in timing, power consumption, and functional adjustments.

4.3 Comparison with state-of-the-art dataset methods

ML methods rely on a large amount of sample data, but there is still a gap in public datasets dedicated to power consumption analysis for ML methods. Established datasets, such as ISCAS-85 (Hansen et al., 1999), MCNC (Brglez et al., 1989), and EPFL (Amarú et al., 2015) offer limited samples and scenarios, which do not fully meet the demands of ML-based methods. Alternative datasets for circuit tasks often derive from specific open-source RTL codes, such as RISC-V designs (OnchipUIS, 2016, 2021; YosysHQ, 2021), CPU-Core designs (OpenCores Team, 2024), or other circuit codes (Corno et al., 2000; Zhou GF et al., 2018; Amid et al., 2020).

Table 2 presents a comparative overview of representative digital IC datasets for ML alongside the proposed dataset. Current dataset methodologies (Gautier et al., 2016; Chowdhury et al., 2021; Chai et al., 2022; Wei et al., 2023) rely on collating specific RTL codes, thereby expanding case sizes compared with traditional datasets. However, these datasets remain constrained in the sample volume. By contrast, the proposed algorithm employs a graph-based circuit generation approach for gate-level circuits focused on power consumption analysis. This method liberates itself from dependence on open-source RTL codes, enabling the generation of an unlimited array of samples by configuring computational graph parameters. Moreover, the resulting circuits exhibit diverse and intricate topologies, facilitating ML methods in effectively learning the internal circuit topologies.

Table 1 Overview of pseudo-power datasets

Type	Metric	Number of input nodes	Number of output nodes	Number of wires	Number of logical gates	Number of registers
Combinational circuit	Mean	49	58	3311	3370	
	Min	22	35	887	939	
	Max	79	83	5638	5715	
Sequential circuit	Mean	254	256	4769	4918	107
	Min	90	98	686	1104	37
	Max	436	440	4950	7429	185

Table 2 Comparison between the proposed power consumption dataset and the ML EDA dataset of application-specific integrated circuit (ASIC) and field programmable gate array (FPGA) in existing studies

Work	Number of cases	Number of sources	Platform & level	Usage
CircuitNet (Chai et al., 2022)	20 000	6	ASIC PR design	DRC; IR drop
OpenABC-D (Chowdhury et al., 2021)	870 000	29	ASIC RTL	Synthesis
HLSDataset (Wei et al., 2023)	19 000	34	FPGA HLS	Power; timing
Spector (Gautier et al., 2016)	8000	9	FPGA HLS	NA
Proposed	Unlimited	NA	ASIC gate-level Verilog	Power

NA: not available; PR: placement & routing; HLS: high-level synthesis; DRC: design rule checking

4.4 Static timing analysis

In digital circuit systems, maintaining the correct timing relationship is crucial, as improper timing can lead to erroneous circuit behaviors. This employs static timing analysis (STA) to verify the signal integrity of the circuit and ascertain if the design constraints are met. We use Synopsys PrimeTime (PT) for this analysis, and set the clock period of the sequential circuit as 10 ns.

As shown in Fig. 3, both the maximum and minimum time slacks of the sequential circuits exceed 0. As the case size increases, the minimum margins remain stable, while the maximum margins decrease to the level of 10^{-2} ns. Since combinational circuits lack clock constraints, our research also examines the maximum arrival time from the primary input to the primary output in these circuits. We observe that the maximum signal arrival time increases with the increase of the circuit size, reaching up to 16 ns.

In conclusion, the timing slacks identified are sufficient, validating the reliability of our proposed pseudo-circuit dataset.

4.5 Simulations of different power analysis methods

This subsection evaluates the performance of prevalent average power analysis methods applied to digital circuits at the pre-layout routing stage, using our dataset. We examine the capabilities of mainstream software algorithms for vectorless power analysis as comparative methods. Power focuses on the average power types prevalent in mainstream digital integrated circuits: leakage power, internal power, and switching power. The industry software tools analyzed include PTPX and Voltus, while OpenROAD's OpenSTA (Ajayi and Blaauw, 2019) is used solely for power analysis in sequential circuits, given its inability to perform such analysis in combinational circuits.

The "Golden" result is established through a random vector test. Initially, the highly accurate activity traces are evaluated using the Monte Carlo method, applying random vectors to assess circuit activity across generated samples. The test uses Synopsys VCS software to simulate and generate circuit activity files. These files, along with the netlists, constraints, process libraries, and other relevant data, are input into PTPX for comprehensive power consumption analysis.

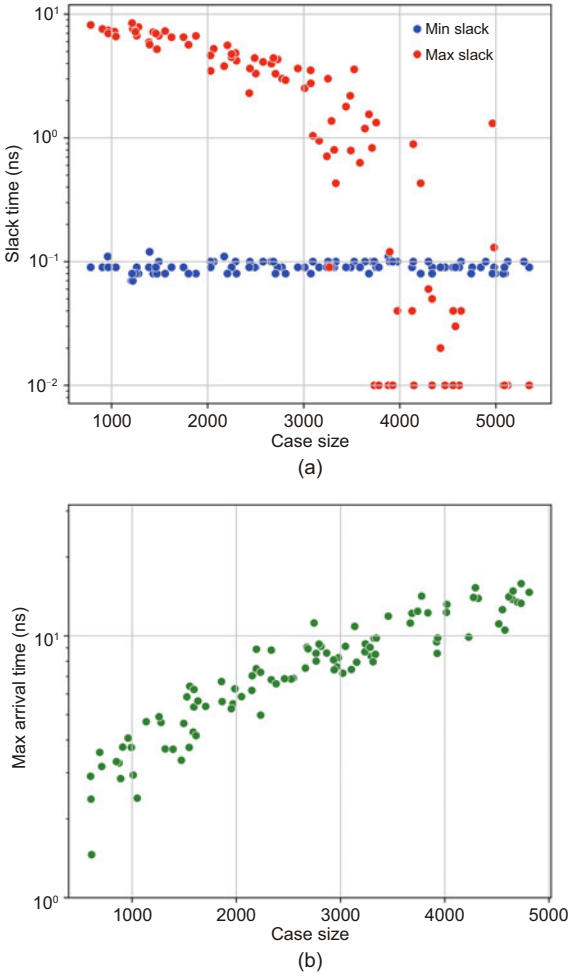

Fig. 3 Relationship between STA and case size: (a) slacks of the sequential circuits; (b) the maximum arrival time of the combinational circuit signal. References to color refer to the online version of this figure

Subsequent to these simulations, the circuit activity file, along with other netlists, constraints, and process library files, is imported into PTPX to determine the final power analysis results. A vectorless power analysis method from each software tool is used as the baseline for comparison. Here, vectorless power analysis means that no switching activity file is required. Instead, it directly performs power analysis based on the primary input signal probability (SP) and toggle rate (TR). This vectorless power analysis method is then compared to the Golden to quantify discrepancies.

In this context, SP is defined as the ratio of the duration at a high signal level to the total test period, while TR refers to the ratio of signal transition

proportion to the same test duration.

Specific parameters are set as follows: SP at the primary input port is fixed at 0.5, and TR varies randomly between 0.01 and 0.4. The signal transfer time is designated as 0.05 ns, with a clock period of 10 ns for sequential circuits.

To demonstrate the validity of the proposed power dataset, a benchmark comprising a blend of wildly acknowledged academic and industrial datasets is created. The combinational circuit benchmark includes samples from ISCAS-85 (Hansen et al., 1999), EPFL (Amarú et al., 2015), and Oracle's OpenSPARC (Parulkar et al., 2008) open-source project. The sequential circuit benchmark is derived from ISCAS-89 (Brglez et al., 1989).

To elucidate the outcomes of various average power analysis algorithms, we analyze the internal, switching, and total power consumption across all samples, and derive the power corresponding to probability density distributions. However, leakage power is excluded due to its negligible error margin. Fig. 4 displays the power consumption test box plots for both the benchmark and the generated datasets, highlighting the mean power consumption values and the upper and lower distribution limits.

In combinational circuits, the values of internal, switching, and total power indicate that, compared to Golden's results, PTPX's results are higher, whereas Voltus's results are lower. The power trend in the proposed dataset aligns with that of the benchmark dataset. In sequential circuits, PTPX aligns most closely with Golden's average values in the benchmark dataset, while Voltus and OpenSTA exhibit substantial discrepancies. Although the generated dataset trends align with PTPX and Voltus from the benchmark dataset, they diverge

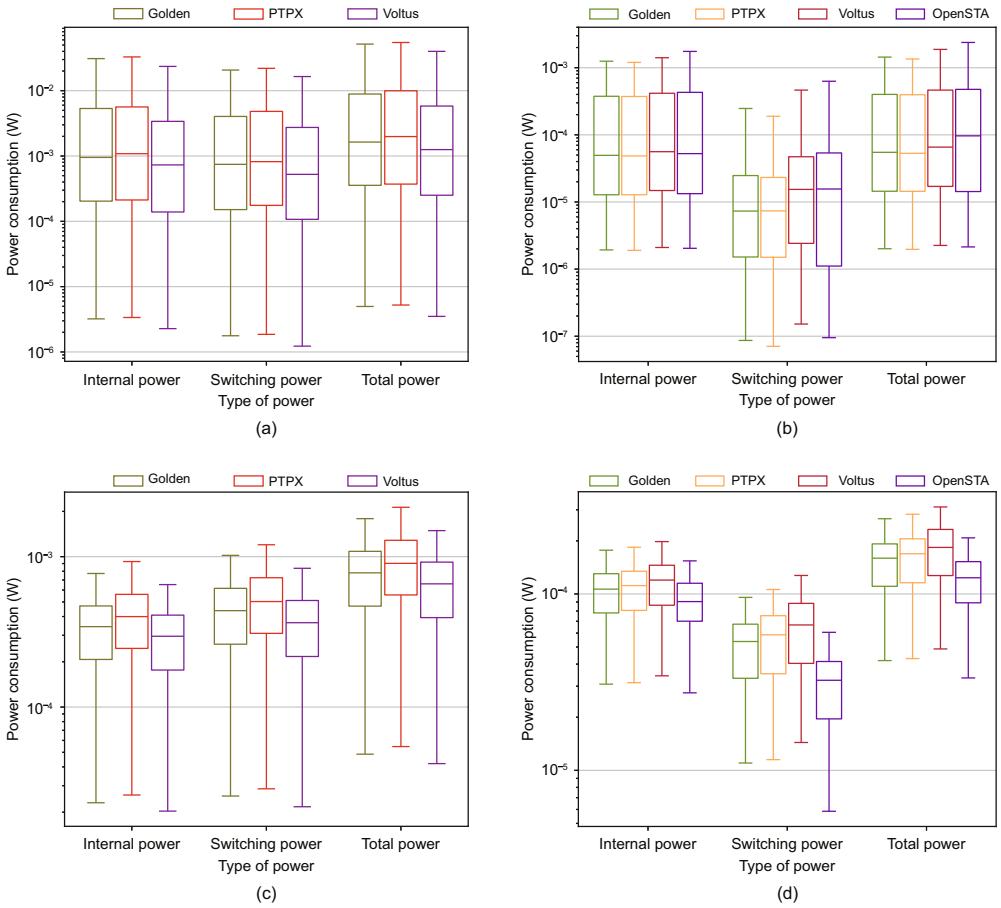

Fig. 4 Results of the three power consumptions for all samples shown on the left for combinational circuits and on the right for sequential circuits: (a) benchmark combinational circuit power consumption; (b) benchmark sequential circuit power consumption; (c) proposed combinational circuit power consumption; (d) proposed sequential circuit power consumption. References to color refer to the online version of this figure

from OpenSTA. Considering being an open-source tool, OpenSTA's calculations lack precision, whereas PTPX and Voltus, as industry standards, provide more reliable results.

The data distribution range for the benchmark dataset spans from 10 gates to 400 000 gates, while the generated dataset predominantly encompasses around 10 000 gates. This discrepancy arises due to the prevalent use of circuits sized around 5000 gates in the field of deep learning EDA power estimation and probabilistic prediction (Shi et al., 2023), which influences the bounds of case size distributions across different datasets.

The consistency in power consumption distribution trends between PTPX and Voltus corroborates the practical utility of the proposed algorithm.

4.6 Circuit topology complexity

We further illustrate the effectiveness of the proposed algorithm in terms of circuit topology complexity. Sequential circuits are derived from combinational circuits, so this subsection focuses on topological complexity analysis of combinational circuits. Assuming that the number of IO ports and the number of internal interconnects are determined, the complexity of the combinational circuits can be adjusted by the expectation γ of the number of nodes mentioned in Algorithm 1.

We define the error ERR between different methods as follows:

$$\mathrm{ERR}=\frac{|P_{\mathrm{vectorless}}-P_{\mathrm{golden}}|}{P_{\mathrm{golden}}}\times 100\%, \qquad (4)$$

where $P_{\mathrm{vectorless}}$ is the power of vectorless power analysis method, such as PTPX, Voltus, and OpenSTA, and P_{golden} is the Golden power.

Under different circuit complexities, the errors of PTPX and Voltus behave differently when the same input stimulus is guaranteed. The higher the circuit topology complexity, the higher the spatial correlation of the signals inside the circuit, and the higher the power errors of PTPX and Voltus. Using this point, we obtain Fig. 5a, where γ takes the values of 1.2, 1.4, 1.6, 1.8, and 2.0. For example, $\gamma = 1.2$ means that the probability of a node having two fan-in nodes is 0.2, the probability of one fan-in node is 0.8, and the expected number of fan-in nodes is 1.2. According to Fig. 5a, as γ increases, the circuit complexity rises, and the power consumption

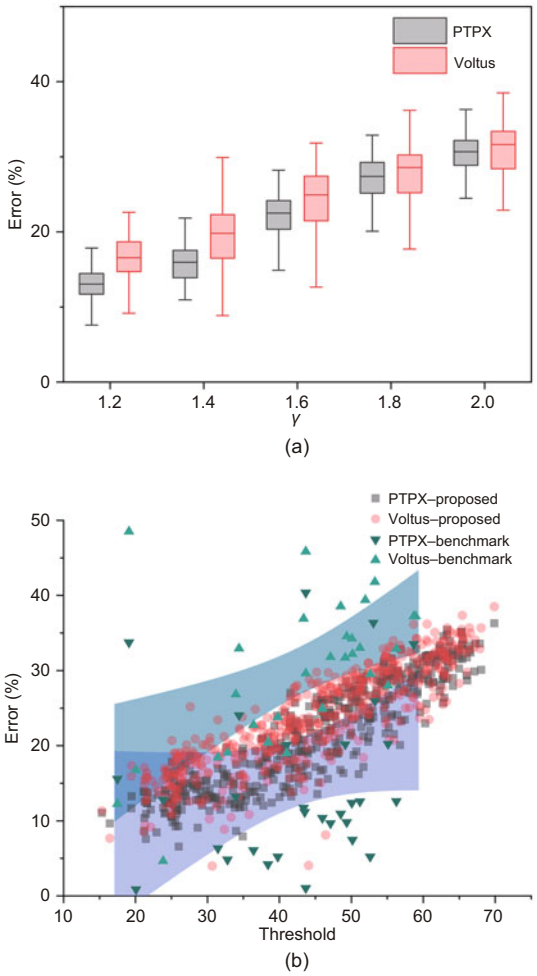

Fig. 5 Simulations on the topological complexities of the generated combinational circuits, where the vertical coordinates are all the total power consumption errors: (a) effect of γ value on the power consumption error; (b) threshold measures, where the upper blue region is the 95% confidence interval of the PTPX error for benchmark and the lower purple region is the 95% confidence interval of the Voltus error. References to color refer to the online version of this figure

errors of both PTPX and Voltus rise, proving the effectiveness of γ on the circuit complexity.

To further illustrate the effectiveness of the proposed algorithm, we validate it from the perspective of circuit complexity in comparison with a standard dataset. We first propose a measure to describe the circuit complexity: the threshold measure. The threshold measure is described in Algorithm 4; i.e., SP and TR of the circuit are analyzed by inference. Due to the influence of circuit spatial correlation, TR tends to increase with the deepening of the circuit hierarchy, threshold measure mainly plays the roles of

resetting the TR when the set threshold is reached and adding up the number of times that the threshold is triggered, and the probability of triggering the threshold is taken as a threshold measure after the completion of the inference at all the nodes.

Algorithm 4 Threshold measurement

Require: Circuit topology G^{t} and input pin SP
Ensure: Measurement T

```
Topo ← Topology ordering of G^t
N ← Number of nodes of G^t
T ← 0
for n ∈ Topo do
    Calculate the SP and TR according to the circuit logic
    if TR > 0.2 then
        TR ← 0.2
        T ← T + 1
    end if
end for
T ← T/N × 100
```

SP is the probability of the signal being high, and TR is the frequency of flipping during the test cycle. The calculation of SP and TR does not consider the spatial correlation of the signals; e.g., for a NAND logic gate with two inputs A and B, SP of its output pin Z is $\mathrm{SP}(Z) = 1 - \mathrm{SP}(A) \times \mathrm{SP}(B)$, and TR is $\mathrm{TR}(Z) = \mathrm{SP}(A) \times \mathrm{TR}(B) + \mathrm{SP}(B) \times \mathrm{TR}(A)$. As the circuit hierarchy deepens, signal spatiality accumulation results in a large TR, and using this feature, we approximate the complexity of the circuit by setting a threshold and resetting TR to characterize the circuit. The circuits generated under different γ values are compared with the benchmark dataset to compute the threshold measure, and the power consumption error distributions of PTPX and Voltus are statistically obtained in Fig. 5b. The error distributions of the proposed algorithm are in the 95% confidence interval of the benchmark algorithm, and the trend is basically the same, which validates the effectiveness of the proposed algorithm for the generation of the dataset from the viewpoint of circuit topology.

4.7 Sample generation time overhead

The time to generate a dataset with a large number of samples requires high generation efficiency for each sample. The time used for generating a single sample includes the time of pseudo-netlist generation, DC synthesis processing, and power analysis. While synthesis and power tests are standard chip processes, we discuss the efficiency of pseudo-netlist generation. Considering that the sequential circuits are based on combinational circuits with gate-level register components, the key factor of generation efficiency is the generation time overhead of the DAG topology, i.e., the algorithmic efficiency discussed in Section 2. To test the efficiency of the algorithm, we generate 1000 pseudo-netlists of different combinational circuits ranging in size from a few gates to 20 000 gates.

The test hardware platform is an Intel Xeon CPU E5-2695 v3 @ 2.30 GHz Dual-CPU server, which has a maximum of 56 threads, and a single sample is run on a single thread. As shown in Fig. 6, there is an approximately linear relationship between the case size and the time overhead of generating DAGs. The test here generates a maximum circuit of 24 000 gates, which takes nearly 40 min. However, the samples can be generated in parallel, and for multiple sample datasets, parallel computation can significantly reduce the runtime.

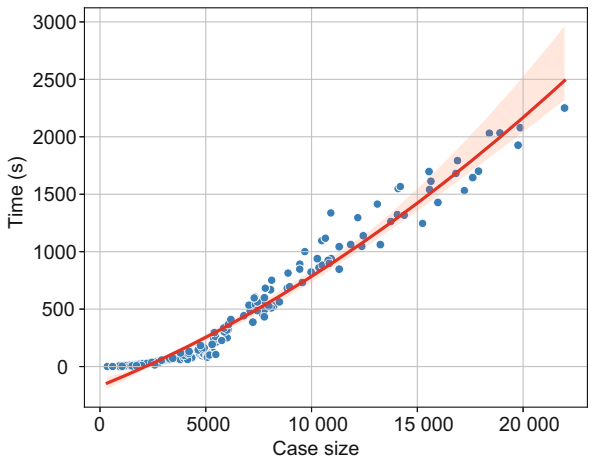

Fig. 6 Time used for generating DAGs of different case sizes

4.8 Sequential circuit zero-value phenomenon

Due to the effect of loops in the circuit, the sequential circuit may have no signal switching registers in some wires throughout the test duration. For sequential circuits with a large number of loops, a large number of nodes without signal switching implies that the switching and short-circuit power

consumption in the circuit tend to be zero-valued, and the circuit's vectorless average power consumption tends to be the same as that of the leakage power consumption. The proposed algorithm can control the percentage of zero values of switching activity in the sequential circuits by adjusting the sequential circuit generation factor α defined in Section 3. We first set up 100 circuits with α ranging from 0.01 to 1.0, with all other parameters being the same, to test their circuit activation performance. According to Fig. 7a, as the sequential circuit generation factor α increases, there are more feedback loops in the circuits. Monte Carlo random vectors are used to test the internal switching activity of the circuits. We find that the proportion of non-zero values is decreasing, which indicates that the increase of feedback loops in the circuits will decrease the circuit switching activity. The high α value circuits correspond to the storage circuit model, while the low α value circuits correspond to the computational task circuit model. α has an approximately linear correlation with the percentage of zeros in Monte Carlo tested circuits, whose results show that this linear relationship is independent of the circuit size.

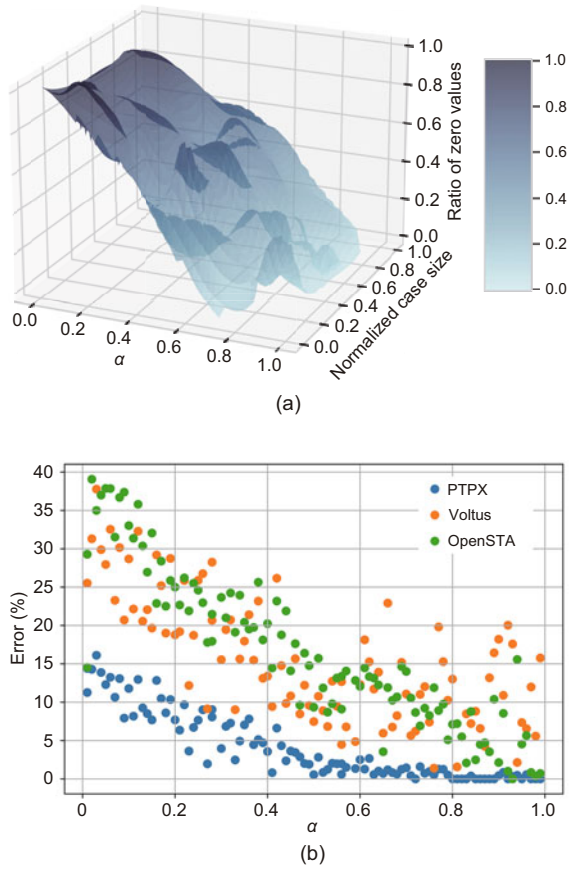

Fig. 7 Sequential circuit Monte Carlo method for flipping active non-zero value duty cycle tests: (a) effect of α on circuit flip-flop activity; (b) performance of different power test methods for different α values of power analysis results. References to color refer to the online version of this figure

We also test the vectorless power analysis algorithms with different generation factors α, as shown in Fig. 7b, using the same sample circuits as in Fig. 7a to test the error performance of different algorithms in terms of the average power dissipation on these samples. To normalize the results, the error computation method is different from that of the previous section. The error calculation differs from the previously mentioned one as follows:

$$\mathrm{ERR} = \frac{|P_{\mathrm{vectorless}} - P_{\mathrm{golden}}|}{\max\left(P_{\mathrm{golden}}, P_{\mathrm{vectorless}}\right)} \times 100\%. \quad (5)$$

According to Fig. 7b, as the circuit generation factor increases from 0 to 1.0, PTPX has the smallest error value for the average power consumption, which is at the bottom of the list. When α tends to be 0, the circuits' topology tends to have fewer register loops, which means the combinational circuits. When α tends to 1.0, according to Fig. 7a, the percentage of zeros in the circuit rises, the error of PTPX is close to 0%, and the errors of Voltus and OpenSTA also decrease.

In summary, based on the results in Fig. 7, we can control the topology of the circuit by adjusting the sequential circuit generation factor α to generate richer sequential circuit samples.

5 Conclusions

In response to the current increase in demand for power consumption analysis circuit amounts, this study proposes a novel method for generating abundant pseudo-datasets for power consumption analysis. Unlike traditional approaches, our method uses graph topology computation to efficiently generate random DAG structures. These DAGs are then transformed into pseudo-combinatorial circuit netlists, with registers added to convert them into pseudo-sequential circuit netlists. We manage the complexity of these circuits by setting an expected node count γ, and address the zero-value problem in power analysis of sequential circuits by introducing a sequential circuit generation factor α that

influences their topology. The produced gate-level description netlists are subject to DC synthesis constraints to create the pseudo-dataset. We validate this dataset using mainstream power analysis tools such as PTPX, Voltus, and OpenSTA, and compare the power consumption results with a standard dataset. Additional evaluations include STA, DAG generation time, and an investigation into the zero-value phenomenon of sequential circuits. Our findings demonstrate that the proposed algorithm can rapidly and efficiently generate a large and diverse set of pseudo-combinatorial and sequential circuit samples. By examining complexity and comparing it with that of benchmark datasets, the results demonstrate and validate the effectiveness of the proposed dataset.

Future research will focus on generating larger-scale pseudo-circuits and expanding the methodology to other areas of IC design, such as IR drop, to enhance the application of learning-based methods within the EDA field.

Contributors

Zejia LYU conducted the investigation, simulation validation, and data organization, and drafted the paper. Xi CHEN assisted in the process of investigation and validation. Jizhong SHEN provided simulation guidance and supervision, and revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Ajayi T, Blaauw D, 2019. OpenROAD: toward a self-driving, open-source digital layout implementation tool chain. Proc Government Microcircuit Applications and Critical Technology Conf, p.1105-1110.

Amarú L, Gaillardon PE, De Micheli G, 2015. The EPFL combinational benchmark suite. Proc 24th Int Workshop on Logic & Synthesis.

Amid A, Biancolin D, Gonzalez A, et al., 2020. Chipyard: integrated design, simulation, and implementation framework for custom SoCs. *IEEE Micro*, 40(4):10-21. https://doi.org/10.1109/MM.2020.2996616

Brglez F, Bryan D, Kozminski K, 1989. Combinational profiles of sequential benchmark circuits. IEEE Int Symp on Circuits and Systems, p.1929-1934. https://doi.org/10.1109/ISCAS.1989.100747

Burch R, Najm F, Yang P, et al., 1992. McPOWER: a Monte Carlo approach to power estimation. IEEE/ACM Int Conf on Computer-Aided Design, p.90-97. https://doi.org/10.1109/ICCAD.1992.279392

Chai ZM, Zhao YX, Lin YB, et al., 2022. CircuitNet: an open-source dataset for machine learning applications in electronic design automation (EDA). *Sci China Inform Sci*, 65(12):227401. https://doi.org/10.1007/s11432-022-3571-8

Chowdhury AB, Tan B, Karri R, et al., 2021. OpenABC-D: a large-scale dataset for machine learning guided integrated circuit synthesis. https://doi.org/10.48550/arXiv.2110.11292

Corno F, Reorda M, Squillero G, 2000. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Des Test Comput*, 17(3):44-53. https://doi.org/10.1109/54.867894

Fang WJ, Lu Y, Liu S, et al., 2023. MasterRTL: a pre-synthesis PPA estimation framework for any RTL design. IEEE/ACM Int Conf on Computer-Aided Design, p.1-9. https://doi.org/10.1109/ICCAD57390.2023.10323951

Gautier Q, Althoff A, Meng PF, et al., 2016. Spector: an OpenCL FPGA benchmark suite. Int Conf on Field-Programmable Technology, p.141-148. https://doi.org/10.1109/FPT.2016.7929519

Hansen MC, Yalcin H, Hayes JP, 1999. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. *IEEE Des Test Comput*, 16(3):72-80. https://doi.org/10.1109/54.785838

Kaeslin H, 2008. Digital Integrated Circuit Design: from VLSI Architectures to CMOS Fabrication. Cambridge University Press, New York, USA, p.386-458. https://doi.org/doi.org/10.1017/CBO9780511805172

Khan S, Shi ZY, Li M, et al., 2024. DeepSeq: deep sequential circuit learning. Design, Automation & Test in Europe Conf & Exhibition, p.1-2. https://doi.org/10.23919/DATE58400.2024.10546639

Kumar AKA, Gerstlauer A, 2019. Learning-based CPU power modeling. ACM/IEEE 1st Workshop on Machine Learning for CAD, p.1-6. https://doi.org/10.1109/MLCAD48534.2019.9142100

Kumar AKA, Al-Salamin S, Amrouch H, et al., 2023. Machine learning-based microarchitecture-level power modeling of CPUs. *IEEE Trans Comput*, 72(4):941-956. https://doi.org/10.1109/TC.2022.3185572

Li M, Khan S, Shi Z, et al., 2022. DeepGate: learning neural representations of logic gates. Proc 59th ACM/IEEE Design Automation Conf, p.667-672. https://doi.org/10.1145/3489517.3530497

Najm FN, 1993. Transition density: a new measure of activity in digital circuits. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 12(2):310-323. https://doi.org/10.1109/43.205010

Nasser Y, Lorandel J, Prévotet JC, et al., 2021. RTL to transistor level power modeling and estimation techniques for FPGA and ASIC: a survey. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 40(3):479-493. https://doi.org/10.1109/TCAD.2020.3003276

OnchipUIS, 2016. MRISCV. https://github.com/onchipuis/mriscv [Accessed on July 26, 2024].

OnchipUIS, 2021. VexRiscv. https://github.com/SpinalHDL/VexRiscv [Accessed on July 26, 2024].

OpenCores Team, 2024. OpenCores. https://opencores.org [Accessed on July 26, 2024].

Parulkar I, Wood A, Hoe JC, et al., 2008. OpenSPARC: an open platform for hardware reliability experimentation. 4th Workshop on Silicon Errors in Logic-System Effects, p.1-6.

Rakesh MB, Das P, Sai Pranav KR, et al., 2023. GRILAPE: graph representation inductive learning-based average power estimation for frontend ASIC RTL designs. 36th Int Conf on VLSI Design and 22nd Int Conf on Embedded Systems, p.1-6. https://doi.org/10.1109/VLSID57277.2023.00053

Shi ZY, Pan HY, Khan S, et al., 2023. DeepGate2: functionality-aware circuit representation learning. IEEE/ACM Int Conf on Computer-Aided Design, p.1-6. https://doi.org/10.1109/ICCAD57390.2023.10323798

Wei ZG, Arora A, Li RH, et al., 2023. HLSDataset: open-source dataset for ML-assisted FPGA design using high level synthesis. IEEE 34th Int Conf on Application-Specific Systems, Architectures and Processors, p.197-204. https://doi.org/10.1109/ASAP57973.2023.00040

Xie ZY, 2023. Efficient runtime power modeling with on-chip power meters. Proc Int Symp on Physical Design, p.168-174. https://doi.org/10.1145/3569052.3578927

Xie ZY, Pan JY, Chang CC, et al., 2023a. The dark side: security and reliability concerns in machine learning for EDA. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 42(4):1171-1184. https://doi.org/10.1109/TCAD.2022.3199172

Xie ZY, Zhang T, Peng YF, 2023b. Security and reliability challenges in machine learning for EDA: latest advances. 24th Int Symp on Quality Electronic Design, p.1-6. https://doi.org/10.1109/ISQED57927.2023.10129359

YosysHQ, 2021. PicoRV32—a Size-Optimized RISC-V CPU. https://github.com/YosysHQ/picorv32 [Accessed on July 26, 2024].

Zhang YQ, Ren HX, Khailany B, 2020. GRANNITE: graph neural network inference for transferable power estimation. 57th ACM/IEEE Design Automation Conf, p.1-6. https://doi.org/10.1109/DAC18072.2020.9218643

Zhou GF, Zhou JY, Lin HJ, 2018. Research on NVIDIA deep learning accelerator. 12th IEEE Int Conf on Anti-counterfeiting, Security, and Identification, p.192-195. https://doi.org/10.1109/ICASID.2018.8693202

Zhou Y, Ren HX, Zhang YQ, et al., 2019. PRIMAL: power inference using machine learning. Proc 56th Annual Design Automation Conf, Article 39. https://doi.org/10.1145/3316781.3317884

Zou SN, Zhang JX, Shi BZ, et al., 2024. PowerSyn: a logic synthesis framework with early power optimization. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 43(1):203-216. https://doi.org/10.1109/TCAD.2023.3297069