2025 Design, Automation & Test in Europe Conference (DATE 2025)

Efficient Hold Buffer Optimization by Supply Noise-Aware Dynamic Timing Analysis

Lishuo Deng, Changwei Yan, Cai Li, Zhuo Chen and Weiwei Shan*
National ASIC System Engineering Research Center, Southeast University, Nanjing, China
emails: {dengls, yancw, licai, chenzhuo_asic, wwshan}@seu.edu.cn; *Corresponding author

***Abstract*—As the CMOS process scales down, digital circuits become more susceptible to hold time violations due to increased sensitivity to supply voltage fluctuations. Since hold time violation is fatal, sufficient hold fixing buffers need to be inserted into the short paths to prevent it. However, by assuming a constant power supply level, traditional hold fixing causes imprecise and overly conservative timing analysis and hence leads to circuit overhead and degraded performance. To address this, we propose a power supply noise (PSN)-aware dynamic timing analysis for realistic hold time analysis and efficient hold buffer optimization, which integrates a machine learning-based timing model into the conventional design flow. Building on the highly effective application of the Weibull cumulative distribution function and machine learning for dynamic PSN-aware timing analysis, we propose introducing an additional parameter for PSN amplitude, which has a significant impact on delay, and narrowing the overall parameter range using real PSN waveforms extracted from the *RedHawk*. This approach achieves a prediction error of only 3.45% for cell delay and 5.1% for path delay, while also reducing dataset acquisition costs. To the best of our knowledge, this work is the first to apply PSN-aware dynamic timing analysis specifically for hold optimization, mitigating the pessimism of traditional static timing analysis (STA) and effectively minimizing redundant hold fixing buffers while remaining compatible with existing design workflows. Since short paths often overlap with critical paths, reducing redundant hold buffers not only decreases area overhead but also enhances performance. Applied to a 22 nm, 64-point Fast Fourier Transform (FFT) circuit, our EDA compatible method combined with a greedy algorithm reduces hold buffers by 55%, achieving not only 6.79% circuit area reduction but also 8.1% performance improvement due to the elimination of redundant buffers in short and critical paths.**

Keywords—Power Supply Noise, Timing Analysis, Hold Buffer Optimization, Greedy Algorithm

I. INTRODUCTION

As integrated circuit technology advances, the complexity of digital circuits continues to increase, particularly with the scaling down of power voltages and the increase in circuit density. This scaling enhances the probability of hold time violations, which occur when the data signal changes too soon after the clock edge, causing the receiving register to capture incorrect data. Hold timing violations are critical for circuits and cannot be remedied by changing the operating frequency of the chip, as can be done with setup violations. Moreover, the considerable number of short paths causing hold violations are more sensitive to variations. Therefore, it is essential to ensure the absolute correctness of hold timing during the timing sign-off process.

In current digital chip design processes, to prevent on-chip variations from causing hold time violations, a series of buffers or delay elements are added as hold fixing buffers to lengthen the short paths at the fast corner like FF process corner, 1.1X supply voltage and best temperature (Fig. 1 top). However, pessimistic estimations of on-chip variations especially voltage variations often lead to the incorporation of

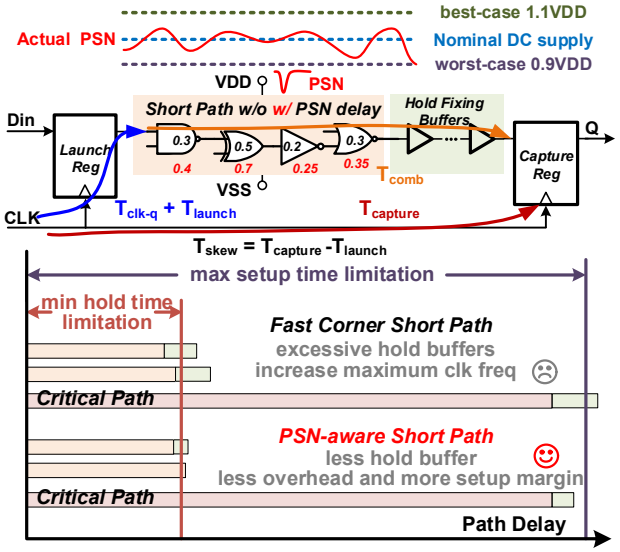

Fig. 1. Hold time fixing, power supply noise and our motivation.

an excessive number of delay elements [1]. Since short paths often overlap with long critical paths, redundant buffers introduce extra signal paths and delay, reducing operating frequency and degrading performance (Fig. 1). Thus, minimizing the design margins reserved for hold time by more efficient and accurate timing analysis such as statistical static timing analysis (SSTA) and most possible point (MPP) theory are crucial for decreasing circuit area and optimizing performance [1], [2].

Traditional static timing analysis (STA) typically involves using the lowest and highest voltage for setup and hold timing sign-off respectively, disregarding the potential impacts of power supply noise (PSN). This approach assigns the worst-case or best-case voltage to each circuit element, resulting in imprecise and overly conservative but PSN causes varying increases in device delay along the paths (Fig. 1). Thus, some studies have focused on addressing PSN issues to reduce the setup timing margins reserved for worst-case [3], [4], [5], [6], [7], [8] but no attempts have been made for hold time optimization.

Generally, lowering the voltage helps alleviate hold time violations before setup violations occur due to the reduced voltage. This circuit principle can be explained by the hold time constraint (1) [9]. As the voltage decreases, clock skew (T_{skew}) increases, but the delays of both the combinational logic and registers increase even more. Thus, considering the PSN effectively avoids pessimistic estimates for hold time (Fig. 1 bottom). Based on this, we naturally employ PSN-aware dynamic timing analysis to eliminate redundantly inserted hold fixing buffers, while also reducing setup margins.

$$T_{hold} + T_{skew} < T_{comb} + T_{clk\text{-}q} \quad (1)$$

In this paper, we propose a PSN-aware dynamic timing analysis for commercial EDA compatible hold time analysis and efficient greedy algorithm based hold buffer optimization,

978-3-9826741-0-0/DATE25/© 2025 EDAA

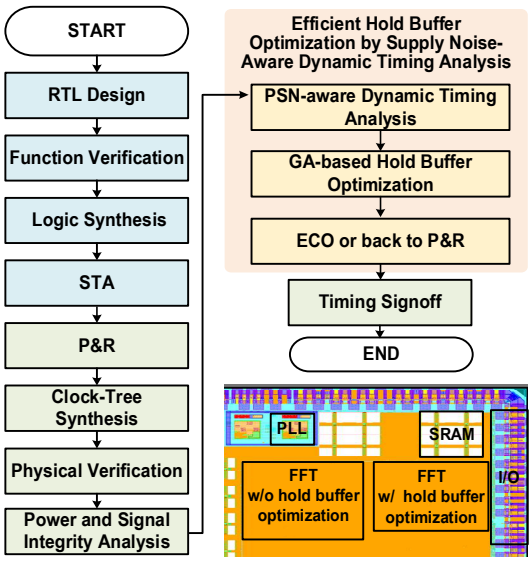

Fig. 2. Our proposed flow for hold buffer optimazation by supply noise-aware dynamic timing analysis and the FFT circuit layout.

which is integrated into the digital IC design flow (Fig. 2). Our main contributions within this work are as follows:

- Building upon the state-of-the-art (SoTA) work [3], we propose introducing an additional peak parameter for varying PSN amplitude caused by different input vector, and obtaining the overall parameter range using real PSN waveforms extracted from the *RedHawk*. This approach achieves a prediction error of only 3.45% for cell delay and 5.1% for path delay, while also reducing dataset acquisition costs by 79.66% compared with SoTA.
- We integrate the multilayer perceptron (MLP)-based delay model into traditional timing analysis workflow, incorporating commercial EDA tool *PrimeTime* to achieve PSN-aware dynamic timing analysis.
- We achieve hold buffer optimization by employing PSN-aware dynamic timing analysis results in conjunction with the greedy algorithm. This effective method has also been integrated into the existing design flow from RTL design to tape-out and applied to a 22 nm, 64-point Fast Fourier Transform (FFT) circuit, resulting in a 55% reduction in redundant delay elements, a 6.79% optimization in circuit area, and an 8.1% improvement in performance.

II. Background and Related Work

A. Hold Fixing Methodology

Hold time refers to the period during which data must remain stable following the arrival of the clock edge at the capture register. If data transitions occur within this hold time, the register may capture unstable data instead of the intended data from the next cycle, potentially resulting in data errors. In current digital chip design processes, to prevent on-chip variations from causing hold time violations, delay elements are added as hold fixing buffers during the design phase to lengthen the short paths [10]. The optimization of hold buffer insertion aims to minimize the number of hold buffers integrated, under the condition that no hold violations occur on short paths. Golanbari M S et al. [1] developed a standard cell library based on the characteristics of cells in the near-threshold regime and introduced an iterative repair process for hold time violations using statistical static timing analysis (SSTA). By accurately extracting timing violation data under the consideration of statistical distributions of near-threshold delays, they successfully reduced the energy cost by 43.3%. Furthermore, to address the high computational cost associated with SSTA, Chen Z in 2018 [2] proposed a modeling approach based on the Most Probable Point (MPP) theory. This method incorporates the nonlinear behavior of circuits and correlation coefficients, significantly enhancing the precision of the designs.

B. Power Supply Noise

Undesirable fluctuations or changes in supply voltage, known as Power Supply Noise (PSN), are introduced by the Power Distribution Network (PDN), power ports, and the inductive or resistive parameters of chip packaging [11] (Fig. 3 left). The noise introduced by inductance is commonly referred to as *Ldi/dt* effect, which depends on the rate of change of instantaneous current in the PDN, as well as the parasitic inductance of the packaging and on-chip components. The noise introduced by resistance is typically known as IR Drop, resulting from the current and resistance within the PDN and represents the low-frequency component of PSN. As transistor geometries and supply voltages shrink, cell delay becomes increasingly sensitive to voltage fluctuations [4]. Assessing PSN-induced additional delays using static IR Drop [5], [6] or average dynamic noise [7], [8] often lacks accuracy due to insufficient waveform information.

Recently, the adoption of machine learning has provided promising advancements [3], [4], [12]. To capture the impact of PSN on device delay, Chen Y et al. utilized the Weibull cumulative distribution function to fit dynamic power noise [3], introducing three power noise parameters that affect the delay of logic gates (Fig. 3 right). The shape parameters α and scale parameter β which are transformed to transition time τ, adjust the curvature and stretching of the distribution function along the x-axis, allowing precise characterization of PSN variations. The device delay is also affected by the relative timing between the input signal transition and the occurrence of PSN. If the input signal transitions occur long after the noise, it no longer influences the device delay. Therefore, the range for the input transition start time ω is determined. They then assessed the impact of these three power noise parameters, input transition times, and output capacitance on the delay and transition times of each timing arc. They constructed a cell-level timing model based on MLP, integrating it into an open-source STA engine to achieve runtime computational efficiency and flexibility.

However, this PSN-aware delay model developed using machine learning poses compatibility challenges with traditional timing analysis workflows and often requires validation of the delay model's accuracy using an open-source

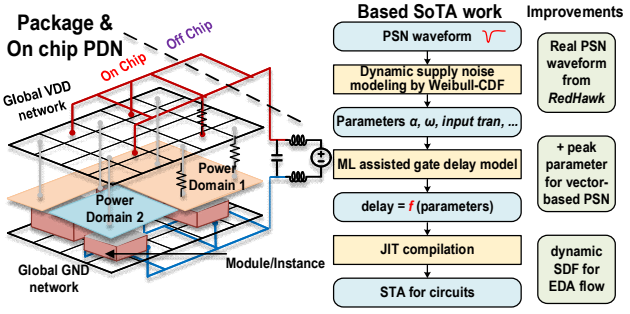

Fig. 3. PDN, SoTA PSN-aware timing analysis [3] and our improvements.

STA engine. Additionally, it only considers vector-less PSN, where the normalized amplitude of all PSN is the same, lacking consideration of vector information that reflects real conditions. Moreover, the overly broad range of MLP input parameters does not reflect the relatively concentrated range of power supply parameters in the same circuit, limiting the model's effectiveness and accuracy.

III. PSN-AWARE DYNAMIC TIMING ANALYSIS

To achieve precise and reliable fitting of PSN waveforms and thereby establish a circuit delay model for timing analysis and hold buffer optimization, we have expanded upon the work [3] incorporating peak parameters to fit waveforms with varying magnitudes of supply noise and narrowing the overall parameter range using real PSN waveforms extracted from the *RedHawk* (Fig. 3 right). This methodology reduces the data acquisition costs for the delay model dataset based on actual power noise conditions and integrates the MLP-based delay model into the conventional timing analysis process.

A. Dynamic Noise Modeling and Dataset Generation

Fig. 4 (left) illustrates the various PSN parameters and signal parameters affecting the delay of an inverter. PSN occurs at time T0 and reaches its peak at time T1. The input signal *I* switches at time ω after T0, with the time required for the voltage to change from 10% to 90% being referred to as the input transition time (input tran). The inverter delay to be solved is defined as the time taken for the output signal voltage to reach 50% after the input signal voltage transitions from 50%. By combining six parameters (PSN parameters α, τ, ω, *peak*, and signal parameters *input tran*, C_{load}) that impact device delay and referencing the analysis results of real PSN, we determine the parameter ranges and sampling intervals. This enables the generation of a PSN-aware device delay dataset.

The parameters input tran and C_{load} can be obtained from the standard cell library. The range of values for the PSN parameters is determined next. Previous SoTA work [3] selected a broad range for other input parameters, though the PSN parameters within the same circuit tend to be more concentrated. In particular, for the shape parameter α, the fitting results from real PSN data indicate that the noise waveform exhibits a steep shape, requiring a parameter range that better reflects actual circuit conditions.

Unlike existing methods, we begin by extracting the names of all instances along a specified path from the circuit's timing report after layout placement. These names are then imported into the power noise sign-off tool, *RedHawk*, to retrieve the PSN waveform for each instance on the path. *RedHawk* is considered the golden standard for PSN analysis. By using real PSN waveforms to determine the parameter ranges for instances along the circuit paths, the parameter space is effectively narrowed, significantly improving the accuracy of the subsequent machine learning-based model that establishes the relationship between noise and delay. Finally, by invoking the noise fitting procedure based on the method of Weibull-CDF, we obtain the PSN parameters α and τ for each instance. To cover various PSN scenarios, we extracted the PSN parameters for 1,000 paths from our FFT circuit. In this work, the range for the shape parameter α is set to [17.5, 21.0] with a sampling interval of 0.5, while the range for the transition time τ is set to [0.085 ns, 0.105 ns] with a sampling interval of 0.005 ns.

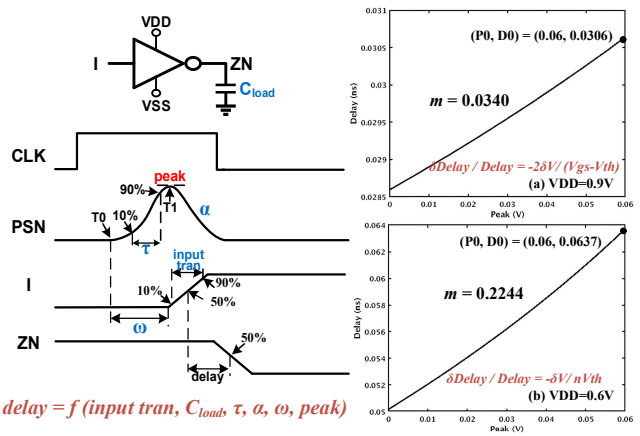

Fig. 4. The definition of PSN parameters and signal parameters (left) and impact of the Peak Parameter on delay (right).

Fig. 4 (right) shows the impact of the newly added peak parameter on device delay, with results presented for operating voltages of 0.9V and 0.6V, respectively. The relationship between peak voltage and device delay is nearly linear because of the relatively small voltage variations. The effect of different peak parameters on delay is expressed using a linear coefficient. At normal voltage (0.9V), the slope of the line is m=0.034, with the device delay at P0=0.06V being D0=0.0306ns. The device delay corresponding to a different peak voltage V_{peak} is calculated using (2). The same applies to an operating voltage of near-threshold voltage (0.6V) or any other necessary voltage for sign-off. By incorporating the peak parameter, we capture the fine-grained variations in voltage peaks caused by vector-base PSN, which improves the accuracy of delay predictions.

$$Delay = m(V_{peak} - P0) + D0 \tag{2}$$

Based on the analysis of the simulation data with our improvements to the SoTA work, the ranges for the input and output parameters are summarized in Table I. A nested loop gate-level SPICE simulation is employed to generate the delay model dataset. First, a MATLAB script generates 8 sets of shape parameters α and 5 sets of transition times τ using the Weibull-CDF method. These 40 different PSN waveforms are stored in piecewise linear (PWL) file format. The peak parameter is set to the maximum PSN amplitude, which is 0.06V based on the *RedHawk* simulation results of the FFT application platform. For each iteration, the SPICE simulation software reads in one set of PSN waveforms and performs a nested loop simulation over 9 sets of input transition start times ω, 8 sets of input transition times, and 8 sets of output loads. This process ultimately generates 20340 data points for each cell.

B. Multi-Layer Perceptron (MLP) Model

We use MLP to train the input parameters, capturing the complex relationship between six input parameters and

TABLE I. THE PARAMETER SETTINGS OF TRAINING DATA GENERATION

	Parameter	Size	Range
Input	α	8	[17.5, 21.0]
	τ	5	[0.085ns, 0.105ns]
	ω	9	[0.1ns, 1.7ns]
	peak	1	[0, 0.06V]
	input tran	8	Foundry cell lib file
	C_{load}	8	Determined
Output	delay	1	-

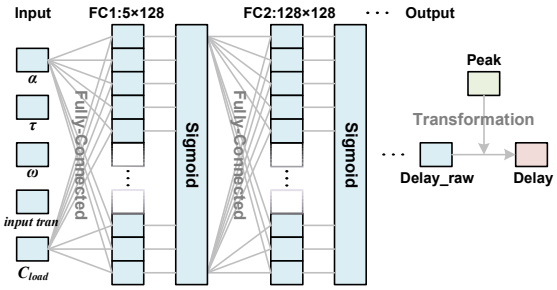

Fig. 5. The architecture of the proposed MLP model.

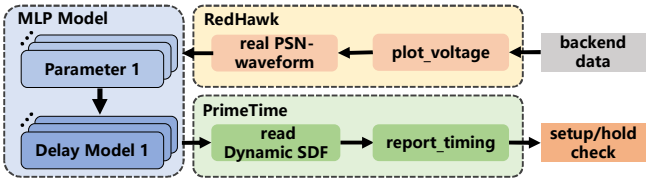

Fig. 6. PSN-aware dynamic timing analysis compatible with traditional EDA processes.

device delay. We explore various network configurations, and the final model consists of three hidden layers with 128 neurons each considering the accuracy and training time (Fig. 5). The MLP model consists of an input layer, three hidden layers with 128 neurons each, and an output layer. The input feature size is 5, representing five parameters, while the output size is 1. The impact of the peak parameter on delay is characterized linearly using (2), which avoids the need to increase the input parameters of the MLP. Instead, an additional linear transformation is applied to the output based on the value of the peak parameter. The activation functions, as well as the loss function used during training, are consistent with work [3]. Detailed results can be found in Section V.

C. Integration with Traditional EDA Flow

After the delay model established, we need to integrate it into the STA flow to perform PSN-aware dynamic timing analysis on various paths in the circuit. By writing the varying delay parameters into the dynamic Standard Delay Format (SDF), this approach bridges the gap between timing sign-off tools and PSN sign-off tools.

In Fig. 6, the PSN-aware delay model is constructed using the previously described method, where the range of noise parameters is determined by the actual noise waveforms of circuit instances. During this period, pre-built delay models for different standard cells are used. By specifying different input parameters, the delay information for various circuit instances can be quickly generated and we obtain a dynamic SDF that accounts for PSN. The dynamic SDF is then used to back-annotate the impact of different PSN on timing. Then, using *PrimeTime* to read the dynamic SDF, a timing report is generated, enabling faster PSN-aware dynamic timing analysis compared with generating modified SDF from *RedHawk*.

IV. Redundancy Hold Fixing Buffer Removal

The improved PSN-aware dynamic timing analysis method described above is applicable to both setup time and hold time analysis. This work is the first to apply this kind of timing analysis method to optimize hold buffers, revealing that under the influence of PSN, the hold time becomes more relaxed, leading to redundancy in the delay elements used for hold time fixing. A greedy algorithm is proposed to eliminate

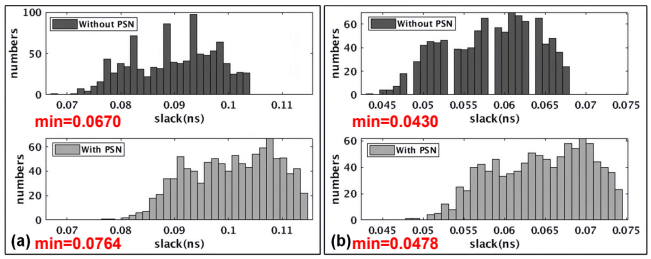

Fig. 7. Hold time slack distribution: (a) normal VDD; (b) 1.1× VDD.

these redundant delay elements and PSN-aware timing analysis results serve as constraint conditions.

A. Hold-time Dynamic Timing Analysis

Conducting our improved PSN-aware dynamic timing analysis across the entire FFT circuit allows the extraction of the 1,000 shortest paths with the tightest hold timing margins. Fig. 7 shows an overall histogram of timing margin data distribution, where the y-axis represents the number of paths in each interval. The upper and lower parts respectively show the timing distribution without and with the consideration of PSN. When using PSN-aware dynamic timing analysis, the hold time margin shows an overall increasing trend, indicating that lower voltage tends to relax hold time constraints. Additionally, Fig. 7 (a) and (b) show the results for operating voltages of supply voltage and 1.1X supply voltage, respectively. It is evident that adding a 10% voltage guard band for hold fixing buffer insertion results in much more conservative outcomes compared to those considering PSN. Thus, accounting for the impact of PSN leads to a more optimistic perspective on hold time, highlighting the redundancy in the delay elements used for hold time fixing. Removing these redundant delay elements can improve both the area and timing performance of the system.

B. Greedy Algorithm for hold buffer removal

To further optimize the removal of redundant hold buffers in the circuit based on PSN-aware dynamic timing analysis, we utilize greedy algorithm. The greedy algorithm does not seek a globally optimal solution but instead breaks the problem into several subproblems, ensuring that each is solved with the locally optimal decision at the time, without revisiting prior decisions. This approach is particularly well-suited for fixing hold time violations in integrated circuits, where the complexity of the design necessitates efficient optimization strategies. Building on the traditional greedy algorithm, this work categorizes redundant delay elements for targeted optimization. This algorithm can also be seamlessly integrated into existing EDA processes. The implementation steps are as follows:

1) Generate the dynamic SDF: Perform PSN-aware dynamic timing analysis on the post-layout circuit to generate the dynamic SDF.

2) Generate the timing report: Match different delay elements from the dynamic SDF and generate a timing report for all hold time-related paths that pass through the delay elements. Additionally, count the types and quantities of the delay elements. Specifically, when a match is found for the HOLD_BUF field in the SDF, a TCL script is generated. This script produces a timing report for all hold time-related paths that pass through the corresponding HOLD_BUF.

3) Set criteria for classification and evaluation: As shown in algorithm 1, if the timing path contains only one type of delay element, the delay of that element is directly compared

to the path's timing margin. If the delay is smaller than the timing margin, the delay element is removed. If the timing path contains two or more types of delay elements, it is first determined whether there is any overlap between the types of delay elements. If there is no overlap, the total delay of the delay elements on the path is compared to the timing margin. If it is smaller than the timing margin, the corresponding delay elements are removed. If there is an overlap between delay elements on different paths, path cross-analysis is required. Since most paths are handled in the first two filtering rounds, the number of overlapping paths is limited. For these paths with overlapping delay elements, timing analysis is performed on each path, and redundant delay elements are removed one by one until a timing violation occurs.

4) Output all redundant delay elements: Combine the results from each step to generate the final file containing all redundant delay elements.

Algorithm 1: Greedy algorithm

Input: All paths P passing through delay units and the timing reports with timing margin information.
Output: The set of redundant delay buffer instances.

/* **Step 1:** Process the path set P_1 containing only a single delay unit *BUF*. */
for each path $p \in P_1$**:**
if delay(*BUF*) ≤ timing_margin(p) **then**
  Remove *BUF* from path p
else
  Retain *BUF* in path p
end
/* **Step 2:** Process the path set P_2 containing N delay units BUF_i where BUF_i ($i \in [1,N]$) do not overlap. */
for each path $p \in P_2$**:**
if ∑delay(BUF_i) ≤ timing_margin(p) then
  Remove BUF_i from path p
else
  Retain BUF_i in path p
/* **Step 3:** Process the path set P_3 containing N delay units BUF_i where BUF_i ($i \in [1,N]$) overlap. */
for each path $p \in P_3$**:**
  Repeat until no further delay buffer can be removed:
  for each i in [1, N]:
    if ∑delay(BUF_i) ≤ timing_margin(p) **then**
      if ∑ delay(BUF_i) ≤ timing_margin(other paths containing BUF_i) **then**
        Remove BUF_i from path p and other paths containing BUF_i
      else
        Retain (BUF_i) in path p
    end
end

V. Experimental Results

A. PSN Fitness and Delay Model Accuracy

1) PSN Fitness Accuracy: We evaluated the effectiveness of the Weibull fitting method in modeling our *RedHawk* generated real PSN waveforms. By aligning the horizontal axis to the data point index, we simultaneously displayed the fitted cumulative distribution function and the original normalized noise waveform of two consecutive cycles, as shown in Fig. 8. Although the rising edge contains very few data points, the method accurately captures the rising behavior of the noise waveform. We used the mean squared error (MSE) to quantify the difference between the fitted function P_i and the original norminal PSN function G_i, as shown in (3), where n is the number of data points. All MSE are below 5×10^{-4}, with an average of 3.217×10^{-4}, indicating excellent fitting performance.

$$\text{MSE of fitted function} = \frac{1}{n}\sum_{i=1}^{n}(P_i - G_i)^2 \tag{3}$$

To further validate the effectiveness of the fitting method, we conducted SPICE simulations (at TT, 25°C) using standard cells to measure the delay error caused by waveform fitting. Both the actual noise waveform and the fitted noise waveform were represented in piece-wise linear (PWL) format. Moreover, we evaluated the delay error caused by function fitting by calculating the Average Absolute Relative Deviation (AARD), as shown in (4). Here, Pi is the inverter delay obtained from SPICE simulations using the fitted noise waveform, and Gi is the delay using the actual sampled noise waveform. The average AARD of the delay error is only 2.66%, confirming the high accuracy of the fitting method.

$$\text{AARD} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{P_i - G_i}{G_i}\right| \tag{4}$$

2) MLP Architecture: To evaluate the model's generalization capability on unsampled data, we randomly divide the dataset into a training set (90%) and a testing set (10%). We used the Adam optimizer, a first-order gradient-based optimization algorithm that can automatically adjust the learning rate to accelerate the convergence of the loss function. The convergence curve of the inverter delay model's loss function is shown in Fig. 9. The loss decreases rapidly and eventually converges to a low value (<0.04), indicating that the model effectively learns the relationships in the data, thereby reducing prediction errors. The inverter delay model achieved an AARD of 3.0% on the test set, demonstrating high prediction accuracy on unsampled data.

3) Delay Model Accuracy: We developed PSN-aware delay models for multiple standard cells commonly used in

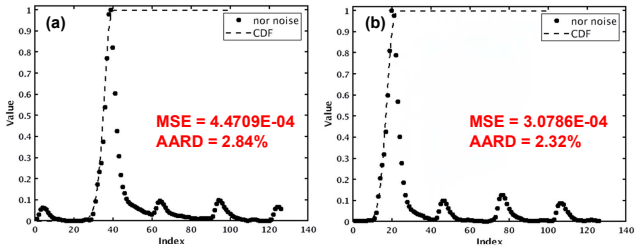

Fig. 8. PSN fittness by Weibull-CDF method (two consecutive cycles).

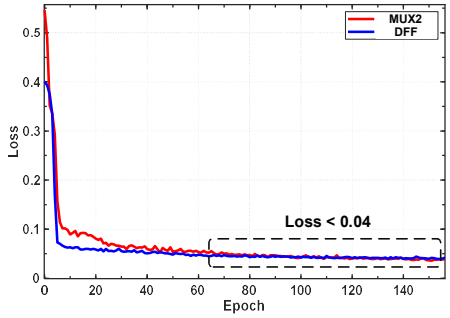

Fig. 9. Convergence curve of the delay model's loss function.

TABLE II. COMPARISON OF PSN-AWARE DEVICE DELAY MODELS

	TCAD 2024 [3]	This Work
Technology	40nm	22nm
Input Parameter	5	6
PSN Type	Vector-less	Vector-based
MSE of Fitted function	1.8095E-02	3.217E-04
AARD of Weibull	3.11%	2.66%
Dataset	100000	20340
MLP Depth	4	5
AARD of Cell Delay	4.89%	3.45%
AARD of Path Delay	6.27%	5.1%

short paths, including inverters (INVD1, INVD4), logic gates (NR2D2, AN3D1, OA211D1, AO211D4), multiplexers (MUX2D1), flip-flops (DFQD1), and delay elements (BUFFD1, DEL075). For logic gates with multiple inputs, we used timing arcs from timing reports to simplify the modeling work. Our models achieved an average AARD of 3.5% compared to SPICE simulation results for these cells. By incorporating specified PSN parameters and signal parameters as inputs, and updating the delay information in the SDF file, we obtained a dynamic SDF that is aware of PSN. Using this dynamic SDF, the path delay predictions for the six monitored short paths showed an average AARD of 5.1%, with a maximum AARD of 5.3%. Considering that traditional lookup table delay models using linear interpolation have errors up to 9.25% [13] compared to SPICE results, this is acceptable.

Table II compared our improvement with the SoTA work [3], and found that our method shows better accuracy with fewer data. Specifically, we achieved an AARD of 3.45% for cell delay and 5.1% for path delay using 20340 samples.

B. Circuit Implementation of Hold Buffer Optimization

We implemented an FFT circuit with our hold buffer optimization method and the original FFT circuit for comparison using a 22 nm CMOS process. This allowed us to verify the improvements in area efficiency and system performance offered by our method. The FFT circuit is based on the R2^{2}SDF algorithm architecture, consisting of three single-path delay feedback units, each containing butterfly computation units, twiddle factors, and complex multipliers. Both designs share the same RTL, with the optimized FFT circuit reducing only the hold buffers while maintaining identical functionality and correctness.

Using traditional hold fixing flow by an experienced backend engineer, our design originally used eight types of delay elements, including BUFF and DEL in the standard cell library. We applied the proposed method to optimize the hold time paths by finding and removing redundant hold buffers. Consistent with our analysis in Section IV, most hold time critical paths contained only a single delay unit, with only 181 paths having overlapping buffers. The optimization results are shown in Fig. 10, where the quantities of all types of hold fixing buffers have decreased. We reduced the total number of delay elements by 55%, and the area occupied by delay units decreased by 56%. This optimization reduced the overall area of the FFT circuit by 6.79%, from 16,030 μm² to 14,942 μm². To assess the impact on performance, we conducted a critical path analysis. The longest path delay was reduced from

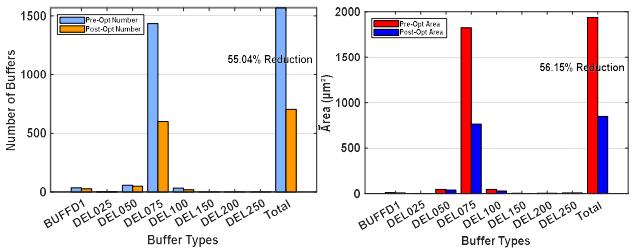

Fig. 10. Hold buffer optimization results of the FFT circuit.

TABLE III. COMPARISON OF HOLD BUFFER OPTIMIZATION

	[1]	[2]	This Work
Process	45nm	45nm	22nm
Application Platform	DMA	DSP	FFT
Method	SSTA	MPP	PSN-aware DTA + Greedy algorithm
Area	↓35.4% buffer	↓7% total	**↓56% buffer ↓6.79% total**
Delay	-	↓23%	↓8.1%

1.48 ns to 1.36 ns, an improvement of 8.1%, increasing the system's maximum operating frequency from 676 MHz to 735 MHz at 0.9V. This approach has also been applied to a RISC-V processor [14], resulting in a 14.8% performance improvement.

Table III compares our hold buffer optimization method by PSN-aware dynamic timing analysis and greedy algorithm with other studies. Our method reduced the area of hold buffers by 56% and the total area by 6.79%, outperforming other studies.

VI. CONCLUSION

In this work, we presented an optimized PSN-aware dynamic timing analysis method for hold buffer optimization. Our approach is fully compatible with existing EDA tools and seamlessly integrated into traditional RTL-to-GDSII design flows. By introducing the peak parameter into the PSN modeling process and leveraging real PSN waveform, we captured finer variations in power noise, improving the accuracy of timing predictions while reducing the data acquisition cost. The proposed method was applied to a 22 nm, 64-point FFT circuit, where a greedy algorithm successfully eliminated 55% of redundant delay elements, resulting in a 6.79% reduction in chip area and an 8.1% performance improvement. This work marks the first application of PSN-aware dynamic timing analysis in hold optimization, highlighting the potential of noise-aware methodologies for future digital circuit designs.

ACKNOWLEDGMENT

The authors gratefully acknowledge Yufei Chen and Lizheng Ren for their valuable technical suggestions. Funded by the National Natural Science Foundation of China (92464204) and the Fundamental Research Funds for the Central Universities. The corresponding author is Weiwei Shan (wwshan@seu.edu.cn).

REFERENCES

[1] M. S. Golanbari, S. Kiamehr, and M. B. Tahoori, "Hold-time violation analysis and fixing in near-threshold region," in *2016 26th International Workshop on Power and Timing Modeling, Optimization*

and Simulation (PATMOS), Bremen, Germany: IEEE, Sep. 2016, pp. 50–55.

[2] Z. Chen, H. Wang, G. Xie, and J. Gu, "A Comprehensive Stochastic Design Methodology for Hold-Timing Resiliency in Voltage-Scalable Design," *IEEE Trans. VLSI Syst.*, vol. 26, no. 10, pp. 2118–2131, Oct. 2018.

[3] Y. Chen, Z. Guo, R. Wang, R. Huang, Y. Lin, and C. Zhuo, "Dynamic Supply Noise Aware Timing Analysis With JIT Machine Learning Integration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 5, pp. 1511–1524, May 2024.

[4] Y. Chen, X. Dong, W.-K. Shih, and C. Zhuo, "Invited Paper: Unleashing the Potential of Machine Learning: Harnessing the Dynamics of Supply Noise for Timing Sign-Off," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, San Francisco, CA, USA: IEEE, Oct. 2023, pp. 1–6.

[5] K. Peng, Y. Huang, R. Guo, W.-T. Cheng, and M. Tehranipoor, "Emulating and diagnosing IR-drop by using dynamic SDF," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2010, pp. 511–516.

[6] K. Peng, Y. Huang, P. Mallick, W.-T. Cheng, and M. Tehranipoor, "Full-circuit SPICE simulation based validation of dynamic delay estimation," in *2010 15th IEEE European Test Symposium*, May 2010, pp. 101–106.

[7] M. Hashimoto, J. Yamaguchi, T. Sato, and H. Onodera, "Timing analysis considering temporal supply voltage fluctuation," in *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference*, Jan. 2005, pp. 1098-1101 Vol. 2.

[8] T. Okumura, F. Minami, K. Shimazaki, K. Kuwada, and M. Hashimoto, "Gate delay estimation in STA under dynamic power supply noise," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2010, pp. 775–780.

[9] G. Neuberger, G. Wirth, and R. Reis, *Protecting Chips Against Hold Time Violations Due to Variability*. Dordrecht: Springer Netherlands, 2014.

[10] M. Chentouf, F. Stevmelin, and Z. E. A. Alaoui Ismaili, "Power-aware hold optimization for ASIC physical synthesis," *Integration*, vol. 76, pp. 13–24, Jan. 2021.

[11] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *Proceedings of International Conference on Computer Aided Design*, Nov. 1996, pp. 524–531.

[12] Y.-C. Liu, C.-Y. Han, S.-Y. Lin, and J. C.-M. Li, "PSN-aware circuit test timing prediction using machine learning," *IET Computers & Digital Techniques*, vol. 11, no. 2, pp. 60–67, 2017.

[13] W. Raslan and Y. Ismail, "Deep-learning cell-delay modeling for static timing analysis," *Ain Shams Engineering Journal*, vol. 14, no. 1, p. 101828, Feb. 2023.

[14] W. Shan, K. Zhou, K. Li, Y. Du, Z. Chen, J. Qian, H. Ge, J. Yang, and X. Si, "14.2 Proactive Voltage Droop Mitigation Using Dual-Proportional-Derivative Control Based on Current and Voltage Prediction Applied to a Multicore Processor in 28nm CMOS," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2024, pp. 256–258.