SMART-GPO: Gate-Level Sensitivity Measurement with Accurate Estimation for Glitch Power Optimization

Yikang Ouyang
HKUST(GZ)

Yuchao Wu
HKUST(GZ)

Dongsheng Zuo
HKUST(GZ)

Subhendu Roy
Cadence Design Systems

Tinghuan Chen
CUHK-Shenzhen

Zhiyao Xie
HKUST

Yuzhe Ma
HKUST(GZ)

Abstract

Dynamic power consumption is a significant concern in modern integrated circuits. This issue is primarily caused by signal toggling, including unwanted toggles known as glitches. With the number of operations increasing in circuits, glitches can lead to significant additional dynamic power. This paper presents SMART-GPO, a novel framework that efficiently and accurately estimates and reduces glitch power. Our approach samples cycles for accurate glitch estimation, followed by gate-sizing and V_{th} assignment to optimize glitch power based on sensitivity measurements. We validated SMART-GPO on the Berkeley Out-of-Order Machine (BOOM) and Rocket SoCs with TSMC N28 technology. It achieves a mean absolute percentage error (MAPE) of 2% on glitch power estimation when running power analysis on only 1% simulation cycles. The optimization results demonstrate that our framework reduces glitch power by more than 9%, which outperforms previous approaches substantially.

ACM Reference Format:
Yikang Ouyang, Yuchao Wu, Dongsheng Zuo, Subhendu Roy, Tinghuan Chen, Zhiyao Xie, and Yuzhe Ma. 2025. SMART-GPO: Gate-Level Sensitivity Measurement with Accurate Estimation for Glitch Power Optimization . In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3697746

1 Introduction

The growing diversification of integrated circuit (IC) applications, spanning from low-power mobile devices to high-end data centers, has elevated power consumption as a key metric in evaluating electronic devices. In CMOS circuits, power consumption can be divided into two categories: dynamic power and static power. Dynamic power is related to signal toggles in circuits, which causes the capacitors in circuits to be charged and discharged [1]. Glitch is a hazardous behavior that incurs additional toggles and consumes power. Distinct from functional toggles, glitch activity is collectively determined by switching activity, gate delay, and the imbalance of arrival times at gate input pins. It is reported that glitch power can represent up to 40% of the total power [2]. Therefore, it is essential to eliminate the glitches such that the total power consumption is reduced.

ASPDAC '25, January 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01…$15.00
https://doi.org/10.1145/3658617.3697746

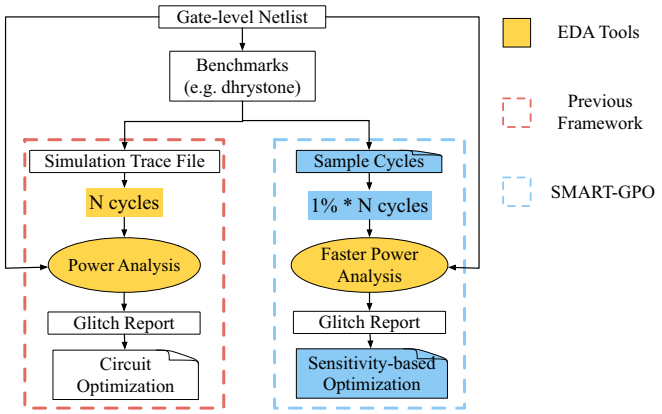

Figure 1: The overview of our framework compared with traditional glitch power analysis and optimization frameworks.

Reducing glitch power in modern designs involves two major challenges, including accurate glitch analysis and effective low-power optimization.

Glitch power analysis is commonly conducted using two categories of methods. The first category is probability-based, proposed in the 1990s [3, 4, 5]. These methods typically propagate switching activities from inputs to outputs in the netlist while incorporating simple timing models to estimate the arrival time differences. Although fast, these methods are less accurate due to neglecting signal correlation and the over-simplified timing model. With the scaling of circuit sizes, it is realized that glitches are difficult to model due to the complex interactions between switching activities and arrival times. Consequently, simulation-based methods that incorporate timing information are favored. They run gate-level simulations and cycle-accurate power analysis with detailed timing on simulation traces by commercial tools [6] or customized glitch analysis algorithms [7, 8, 9]. Although these methods are accurate as they capture glitch activities in detail, they are time-consuming. For instance, conducting a cycle-accurate glitch power analysis with dhrystone benchmark on BOOM SoC [10] requires more than 40 hours. Similar runtime issues have also been reported in [9]. Hence, these analysis methods may not be feasible when a design is optimized in an iterative manner.

In addition to glitch power estimation, optimizing a design to reduce glitch power is another critical task. Glitch power optimization is usually performed through Engineering Change Orders (ECOs), which typically involve gate sizing and V_{th} (threshold voltage) assignment on the logic gates. Previous studies on glitch power optimization can be categorized into numerical optimization and metric-based

methods. In [11], glitches are reduced by gate sizing, which is formulated as a geometric programming (GP) problem [12]. However, solving the GP problem becomes computationally prohibitive for netlists containing millions of gates in a modern design. Besides, in advanced technology nodes, gate delay may not be accurately described by a linear model required by GP [13]. For metric-based methods, they devise intricate metrics to indicate the amount of glitch power a gate consumes due to glitch generation and propagation [9, 14, 15]. Then, optimization techniques such as gate down-sizing and increasing V_{th} are performed on the gates based on the ranking of metrics. Unfortunately, no matter how intricate the metrics are, they can only guide glitch optimization locally as more timing imbalances and glitches may be potentially incurred elsewhere in the design after ECO. What's worse, these two categories both suffer from the escalating scale and complexity of netlists and hence, are only tested on small combinational circuits [9, 14, 15].

In this paper, we introduce sensitivity measurement for glitch power optimization equipped with fast yet accurate glitch power estimation. As shown in Figure 1, for an efficient estimation, our method samples a small number of cycles from the benchmark and runs cycle-accurate power analysis to estimate glitch power at each gate. This approach reduces the time required for full power analysis and preserves detailed timing based on simulation traces to reflect actual glitch activities, which can be both fast and accurate. Based on the accurate estimation, we further propose a gate-level sensitivity measurement framework for glitch power reduction. Different from the previous glitch power optimization approaches that are purely based on intricate metrics for gate selection in ECO and lack a guarantee on glitch power reduction [9, 14, 15], we are able to correlate the action of optimizing a gate directly with the total glitch power reduction by measuring the sensitivity of each gate, which thus guarantees glitch power reduction after ECO. Our contributions are summarized as follows:

- We propose a synergistic framework for accurate glitch power estimation and effective optimization.
- A random sampling approach is proposed to estimate glitch power accurately and efficiently, which facilitates further glitch power optimization.
- Furthermore, we propose a gate-level sensitivity measurement framework to analyze the sensitivity of each gate for glitch power reduction, in which a Monte-Carlo sampling approach is applied. The gates with high sensitivity are selected during the ECO step to ensure total glitch power reduction.
- Experimental results on two large RISC-V designs show that the proposed method can achieve high estimation accuracy for glitch power, with the mean absolute percentage error of 2%. Moreover, the glitch power on two designs can be reduced by 9% and 17%, which outperforms baseline methods substantially.

2 Preliminary and Problem Formulation

2.1 Glitch Power

The dynamic power consumption of circuits is brought about by signal toggles. In addition to the toggles that fulfill circuit functionality, there are extra toggles. These extra toggles are called glitches, and they add to dynamic power without affecting the circuit functionality.

Glitches are typically analyzed using the inertial delay model [16]. A gate will ignore an incoming signal if the following signals (arriving at the same or a different input) arrive within the gate's inertial delay. According to the inertial delay model, there are two types of glitches that consume power:

(1) Generated glitches (Figure 2a). A glitch can be generated when the difference in arrival times of any pair of input pins is larger than the gate's inertial delay, and the signal patterns on pins satisfy the condition for a glitch.
(2) Propagated glitches (Figure 2b). A glitch may propagate to transitive fan-out gates if the width of the glitch pulse exceeds their inertial delay and the conditions allow for propagation, further consuming power.

They can contribute to more than 35% dynamic power of combinational gates on BOOM, as depicted in Figure 2c.

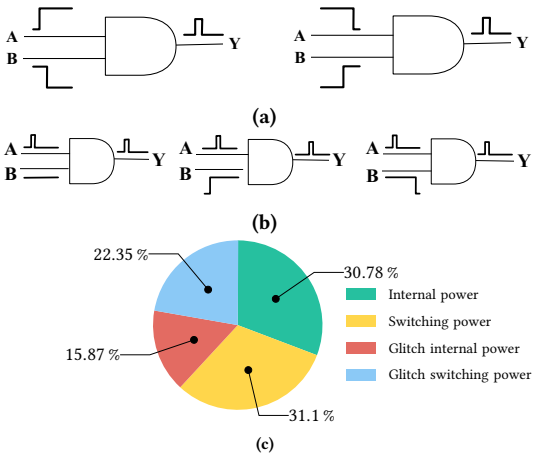

Figure 2: (a) glitch generation. (b) glitch propagation. (c) components of dynamic power.

Table 1: Notations in this work

Notations	Explanations
N	The number of cycles in the benchmark.
$c_j, j \in [1, N]$	The jth cycle out of a total of N cycles in the benchmark.
π_j	The variable that defines if c_j will be sampled, $\pi_j \in \{0, 1\}$.
$g_k, k \in [1, K]$	The kth gate out of a total of K gates in the design.
$p_{k,j}$	The glitch power of the kth gate at the jth cycle.
p_k	The average glitch power of gate k on all cycles.
$\tilde{p}_k$	The average glitch power of gate k on sampled cycles.

2.2 Problem Formulation

(1) Given a netlist with K gates and a benchmark with N cycles, the first task is to estimate glitch power at the gate level accurately and efficiently.
(2) With the glitch power estimation approach, the next task is to reduce the total glitch power by optimizing the design.

3 Glitch Power Estimation

In this section, we will elaborate on our methodology for fast and accurate glitch power estimation. Some notations are shown in Table 1 for a clearer illustration. The overview of our estimation workflow is shown in Figure 3, which is essentially based on a random sampling approach.

Typically, the power consumption for circuits is evaluated by simulating long-running benchmarks and running power analysis

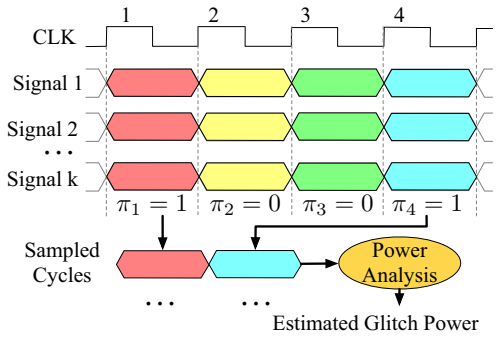

Figure 3: An illustration of sampling cycles for power analysis.

on the simulation traces. However, it is impractical to run power analysis on the entire simulation trace due to the unacceptable runtime, especially when cycle-accurate power with detailed timing is needed for a glitch power analysis. Although some ML-based methods have been proposed for a fast power estimation at both design level [17, 18, 19, 20, 21] and gate level [22, 23], they do not apply to glitch power estimation as they ignore detailed timing at gate level. Compared with them, sampling-based methods propose to estimate power consumption by sampling a small subset of simulation traces and running power analysis on them with gate-level timing information [24, 25, 26], thus can take glitch activities into account.

Our fast glitch power estimation also adopts this sampling idea. It aims to estimate glitch power at the gate level when simulated by a benchmark, as shown in Equation (1).

$$p_k = \frac{1}{N}\sum_{j=1}^{N} p_{k,j}. \tag{1}$$

Suppose we sample each cycle j with a sampling variable π_j, the estimated average glitch power of gate k is as follows:

$$\tilde{p}_k = \frac{1}{\sum_{j=1}^{N}\pi_j}\sum_{j=1}^{N}\pi_j p_{k,j}. \tag{2}$$

Here, each term $p_{k,j}$ is masked by multiplying a binary sampling variable π_j, which reflects our sampling strategy. This sampling process can also be illustrated in Figure 3, where sampled cycles are used for power analysis in EDA flow.

While there are many sampling strategies, the most statistically robust strategy is random sampling without replacement [24]. Thus π_j should adhere to a binomial distribution, as shown in Equation (3).

$$\pi_j \sim \mathcal{B}(1, r). \tag{3}$$

The sampling ratio r decides how many cycles to be sampled w.r.t the size of the benchmark, which trades off between the speed and accuracy of the estimation. Given the distribution of the sampling variable as in Equation (3), we can write the expectation of estimated glitch power as:

$$\mathbb{E}[\tilde{p}_k] = \frac{1}{Nr}\sum_{j=1}^{N}\mathbb{E}(\pi_j p_{k,j}) = \frac{1}{Nr}\sum_{j=1}^{N}(r p_{k,j}) = \frac{1}{N}\sum_{j=1}^{N} p_{k,j} \tag{4}$$

The expectation of estimated glitch power, as shown in Equation (4), is equivalent to Equation (1), thus is unbiased. According to the observation from [24, 25], with enough samples, the estimated glitch power should be accurate.

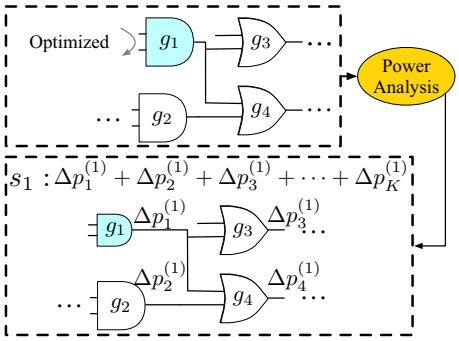

Figure 4: The definition of sensitivity of g_1.

4 Glitch Power Optimization

On FPGA, glitches are reduced by inserting flip-flops [27] or balancing arrival times at the routing stage [28]. On ASICs, glitch power can be optimized by ECO, which involves down-sizing gates (also increasing V_{th}) to increase gate delay for filtering glitches. Previous glitch power optimization approaches [9, 15] are based on an iterative procedure. Firstly, a metric is defined to rank the significance of the gates for glitch power consumption. In each iteration, one gate is selected based on the ranking, which is then downsized to the smallest size and assigned high V_{th} as long as there is no timing violation. However, these metrics only reflect glitch activities and guide the optimization locally, which fails to consider that new glitches can be generated elsewhere in the design. Such complex glitch behaviors make them uncertain of total glitch power reduction and thus may not guarantee a positive result.

Motivated by this, we aim to correlate the gate optimization directly with the total glitch power reduction in the netlist, which provides a more global view of design optimization and glitch power reduction. Thus, we propose a gate-level sensitivity measurement method to evaluate the potential glitch power reduction through gate sizing. For brevity, we refer to *optimizing a gate* as sizing a gate with the smallest size and assigning high V_{th} in the remaining context. *It should be noted that we only use the smallest size and high V_{th} as* [9, 15] *because calling incremental timing updates outside the commercial EDA tool is not often efficient. However, it would be easier for the tools to explore different sizes (or even buffering) for each gate and incorporate ones without timing violations. This setting does not affect the feasibility of the proposed framework.*

4.1 Sensitivity Measurement

Given the netlist and the technology information, the target is to optimize the netlist such that the glitch power $P = \sum_{i=1}^{K} p_i$ is minimized, in which the timing constraints should be satisfied.

Optimizing a gate may result in glitch power changes on various other gates in the design, where the power change is possibly positive or negative. We can represent the glitch power change on g_i as Δp_i and denote the portion of this change attributable to optimizing g_k as $\Delta p_i^{(k)}$. The relationship between glitch power changes and gate optimizations can be represented as a matrix S:

$$\mathbf{S} = \begin{bmatrix} \Delta p_1^{(1)} & \Delta p_1^{(2)} & \cdots & \Delta p_1^{(K)} \\ \Delta p_2^{(1)} & \Delta p_2^{(2)} & \cdots & \Delta p_2^{(K)} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta p_K^{(1)} & \Delta p_K^{(2)} & \cdots & \Delta p_K^{(K)} \end{bmatrix}. \tag{5}$$

Here, we assume individual gate optimizations are independent w.r.t the objective, which is referred to as a *first-order greedy search* for an NP-hard problem [29].

To characterize the potential impact of optimizing a gate to the glitch power change, we define the sensitivity of g_k as the total glitch power change after g_k is optimized, i.e.,

$$s_k = P^{(k)} - P = \sum_{i=1}^{K} p_i^{(k)} - \sum_{i=1}^{K} p_i = \sum_{i=1}^{K} \Delta p_i^{(k)}. \tag{6}$$

Thus, the per-gate sensitivity defined in Equation (6) is essentially the sum of column k in S. This concept is exemplified in Figure 4. The glitch power changes on gates can be obtained from our fast estimation to calculate the sensitivity of g_1.

In the context of glitch power optimization, the total glitch power should be minimized. Gates with the most negative sensitivities are referred to as the most *sensitive* and are prioritized as the final optimization candidates. However, evaluating sensitivities requires running glitch power analysis on K versions of the optimized netlist, which is prohibitive for large designs. In the next section, we propose an efficient computation for sensitivity.

4.2 Efficient Sensitivity Calculation

4.2.1 Sensitivity Estimation via Monte-Carlo. To find the most sensitive gates efficiently, we estimate their sensitivity without exhaustively evaluating all possible netlists. Fortunately, Monte-Carlo method [30] provides us an estimator to approximate the sensitivity s_k by drawing independent batches of gates $B_1, B_2, \ldots, B_M$ following a distribution $\mathcal{P}(g_k)$, and then compute the average of the sensitivity s_k evaluated at these batches [31]:

$$s_k = \frac{1}{M'} \sum_{j=1}^{M} f_j(g_k), \forall g_k \in B_j, B_j \sim \mathcal{P}(g_k), \tag{7}$$

where $f_j(g_k)$ means the computed sensitivity of batch j based on Equation (6). M' is the number of times g_k is sampled among M batches. Therefore, to estimate $s_k, \forall k$, we can sample batches of gates in the netlist for optimization and compute their sensitivity, which is facilitated by our fast glitch power estimation and can be launched in parallel as well.

4.2.2 Prior Distribution for Batch Sampling. Generally speaking, when more batches are sampled, the sensitivity computation will be more accurate [32], which incurs a trade-off between runtime and accuracy. To achieve a higher efficiency of batch sampling-based computation, we can leverage certain prior distributions for sampling instead of just sampling uniformly. A straightforward way is that the sampling probability can be correlated with the per-gate glitch power. Apart from this per-gate glitch power, other metrics also signify the glitch power a gate may cause, like glitch criticality metric [9] or power metric [15]. Nonetheless, we can guide our sampling process with these metrics. Suppose each gate g_k is associated with a metric m_k. The probability of sampling a gate equals the normalized metric:

$$\mathcal{P}(g_k) = \frac{m_k}{\sum_{i=1}^{K} m_i} \tag{8}$$

4.2.3 Fine-grained Sensitivity Computation. After the ECO steps are performed on the sampled gates, the proposed fast glitch power estimation is launched to obtain the new glitch power. All the following equations refer to the sensitivity computation in one batch B_j, so we ignore the subscript j for brevity. For a batch of sampled gates, the sensitivity defined in Equation (6) can be written as the total glitch power change averaged on the optimized gates:

$$s_k = \frac{1}{|B|} \sum_{i=1}^{K} \Delta p_i b_k, \tag{9}$$

where b_k is a binary variable that is 1 if $g_k \in B$ and 0 otherwise.

By sizing gates and calculating sensitivity in batches, the requirement for computation resources is eased. However, in Equation (9), the sensitivities are the same for gates in a batch, which implies that the optimized gates contribute equally to the total glitch power reduction, which may not be true. In our approach, we will use a more fine-grained computation for sensitivity to reflect the actual glitch activities by modifying Equation (9) with a weight $w_{k,i}$:

$$s_k = \sum_{i=1}^{K} w_{k,i} \Delta p_i b_k. \tag{10}$$

This calibration can be illustrated in Figure 5, where optimizing g_4 does not contribute to the glitch power reduction of g_2. Accordingly, when g_i is not in the fanout of the concerned gate g_k, $w_{k,i}$ should be set to 0. Further, the $w_{k,i}$ should be reduced as g_i becomes more distant from g_k as the glitch power reduction on g_i may be attributed to other optimized gates. For example, Δp_3 should be attributed to both the optimization of g_1 and g_3. This behavior can be represented by an factor γ that attenuates $w_{k,i}$ as g_i becomes more distant from g_k, i.e.,

$$w_{k,i} = \gamma^l, \tag{11}$$

where l is the difference of logic level between g_k and g_i.

Combining the above Monte-Carlo estimation with prior distribution on glitch power and calibration for computation, our optimization framework is depicted by Algorithm 1. At first, we independently

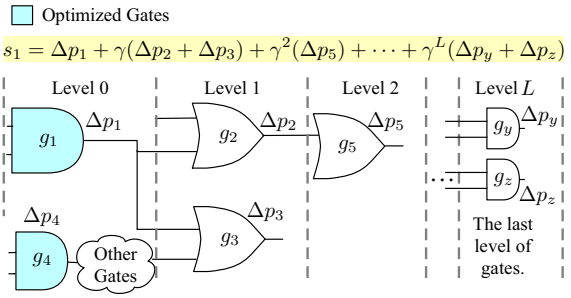

Figure 5: Example of sensitivity computation for g_1. The levels are counted starting from g_1.

Algorithm 1 Gate Sizing for Glitch Power Reduction

Require: Original netlist, where each gate g_k is with metric m_k.
1: Initialize M batches independently and sample gate g_k with probability $\frac{m_k}{\sum_{i=1}^{K} m_i}$.
2: **for** each batch B **in parallel do** ▷ Parallel processing of batches
3: Optimize gates in B to minimum size, assign high V_{th}.
4: Run fast glitch power estimation for optimized netlist for B.
5: Compute sensitivity by Equation (10) for all sampled gates.
6: **end for**
7: Average sensitivity by Equation (7) across batches.
8: Select the most sensitive gates for final sizing.

Table 2: Profiles of benchmarks and estimation results.

Design	Benchmark	T_{GS} [a] (s)	# of cycles	T_A [b] (h)	$\mathbf{T_S}$ [c] **(h)**	True Glitch Power (mW)	Estimated Glitch Power (mW)	**MAPE**	**K-τ**
BOOM	dhrystone	2916	502947	47	**2.8**	2.823	2.821	**0.602%**	**0.99**
	median	832	167573	12	**0.9**	1.864	1.830	**0.397%**	**0.96**
	mt-matmul	948	194153	16	**1.1**	1.915	1.910	**0.627%**	**0.97**
	mt-vvadd	2708	579325	50	**3.2**	1.716	1.718	**0.233%**	**0.98**
	multiply	1046	184757	17.5	**1.0**	2.636	2.640	**0.645%**	**0.97**
	qsort	4290	687071	70	**3.8**	3.528	3.537	**0.062%**	**0.99**
Rocket	dhrystone	602	529661	12	**0.7**	0.693	0.697	**0.559%**	**0.99**
	median	143	165077	3.5	**0.2**	0.240	0.238	**0.835%**	**0.99**
	mt-matmul	194	206643	4.5	**0.3**	0.317	0.319	**0.576%**	**0.98**
	mt-vvadd	518	610296	11	**0.8**	0.188	0.190	**1.291%**	**0.98**
	multiply	1046	184757	17.5	**0.3**	0.658	0.650	**1.471%**	**0.99**
	qsort	782	628464	37	**0.9**	1.010	1.012	**0.347%**	**0.99**

[a] The time for running gate-level simulation. [b] The time for running power analysis on all cycles in the benchmark.
[c] The time for running power analysis on sampled cycles.

sample batches of gates. Then, we optimize gates for each batch and run fast glitch power estimation on optimized netlists in parallel. Finally, we compute sensitivities for sampled gates and average them across batches to select the most sensitive ones as the final optimization candidates. We also run a gate-by-gate ECO as in [9] to avoid timing violations during the gates optimization.

5 Experimental Results

In this section, we first demonstrate the accuracy and efficiency of our glitch estimation framework. We then provide glitch optimization results with our sensitivity measurement based on these estimations.

5.1 Experimental Setup

To validate our framework in real-world scenarios, we apply it to two RISC-V SoCs, BOOM and Rocket [10]. The designs are synthesized by Synopsys Design Compiler with the TSMC N28 process. Target frequencies are set to 500 MHz for both designs. There are in total 288789 gates in BOOM and 114750 gates in Rocket. Simulations are run by Synopsys VCS using six commonly used benchmarks, as demonstrated in Table 2. The (sampled) simulation traces are sent to PrimeTime PX for power analysis. PrimeTime PX also handles the V_{th} assignment and gate sizing.

5.2 Glitch Power Estimation Results

Our framework efficiently estimates glitch power by analyzing sampled cycles in benchmarks. Empirically, we sample 1% cycles for power analyses, i.e., $r = 0.01$ in Equation (3). We will illustrate our framework's efficiency in glitch power estimation and its contribution to optimization.

In terms of glitch power estimation, we evaluate the error against the ground truth using the Mean Absolute Percentage Error (MAPE), as defined in Equation (12).

$$\text{MAPE} = \sum_{k=1}^{K} \frac{|p_k - \tilde{p}_k|}{p_k}. \tag{12}$$

Our method runs more than 10X faster than a complete power analysis and achieves an MAPE of less than 2%, as shown in the sixth and second last columns ($\mathbf{T_S}$ and **MAPE**) in Table 2, respectively. Ideally, the speed-up of power analysis should be linearly proportional to the number of cycles being analyzed, i.e., 100X. However, we observe that PrimeTime runs slower when it processes non-continuous time windows. We attribute the non-ideal speed-up to the internal overhead of the analysis tools.

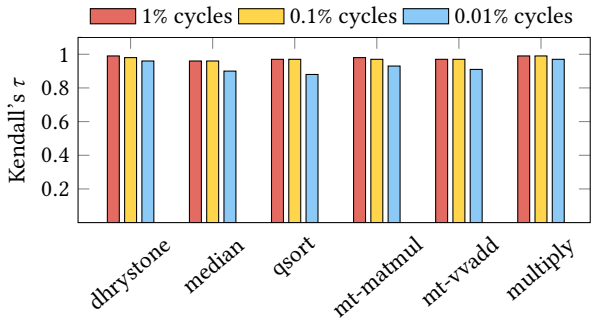

Figure 6: Glitch power estimation Kendall's τ on BOOM.

In addition to accurately estimating glitch power on gates, our goal is to facilitate further optimizations. Since glitch power is often optimized in a gate-by-gate fashion based on metrics related to per-gate glitch power, the correlation between the estimated per-gate glitch power ranking and the true per-gate glitch power ranking is critical for effective netlist optimization. We use Kendall's tau [33] to quantify this correlation, with values close to one indicating high agreement between the rankings. As shown in the last column (**K-τ**) of Table 2, all Kendall's tau values exceed 0.96, demonstrating that our estimated gate-level glitch power rankings closely match the true rankings. To understand the impact of sample sizes and sampling strategies on the estimation correlation, we further sample 0.1% and 0.01% of cycles and show the result on BOOM in Figure 6. Even random sampling 0.1% can reach a Kendall's τ larger than 0.9. This high correlation validates our method's ability to guide the gate-by-gate optimization process effectively. We attribute our framework's high accuracy and efficiency to the sampling strategy employed. It aligns with the observation in previous works [24, 25] that run power analysis on randomly sampled cycles is able to estimate power accurately, with glitch power included.

Table 3: Glitch power optimization results (Unit: mW).

Design	Benchmark	Original	PGM	PM [15]	GCM [9]	**Ours**
BOOM	dhrystone	2.823	2.832 (↑0.32%)	2.794 (↓1.03%)	3.156 (↑11.79%)	**2.674 (↓5.28%)**
	median	1.864	1.755 (↓5.85%)	1.647 (↓11.64%)	2.025 (↑8.64%)	**1.548 (↓16.95%)**
	mt-matmul	1.915	1.986 (↑3.71%)	1.877 (↓1.98%)	2.294 (↑19.79%)	**1.790 (↓6.53%)**
	mt-vvadd	1.716	1.835 (↓6.93%)	1.713 (↓0.17%)	2.116 (↑ 23.31%)	**1.611 (↓6.12%)**
	multiply	2.636	2.625 (↓0.42%)	2.535 (↓3.83%)	2.928 (↑11.08%)	**2.426 (↓ 8.0%)**
	qsort	3.528	3.271 (↓7.28%)	3.368 (↓4.54%)	3.618 (↑2.55%)	**3.130 (↓11.28%)**
	Average	2.414	2.384 (↓1.24%)	2.322 (↓3.81%)	2.690 (↑11.43%)	**2.196 (↓ 9.01%)**
Rocket	dhrystone	0.693	0.618 (↓10.88%)	0.601 (↓13.37%)	0.630 (↓9.19%)	**0.585 (↓15.58%)**
	median	0.240	0.212 (↓11.6%)	0.206 (↓13.82%)	0.207 (↓13.65%)	**0.172 (↓28.31%)**
	mt-matmul	0.317	**0.291 (↓8.23%)**	0.309 (↓2.65%)	0.315 (↓0.76%)	0.294 (↓7.13%)
	mt-vvadd	0.188	0.174 (↓7.35%)	0.181 (↓3.72%)	0.189 (↑0.80%)	**0.174 (↓7.45%)**
	multiply	0.658	0.493 (↓25.06%)	0.479 (↓27.20%)	0.505 (↓23.30%)	**0.467 (↓29.01%)**
	qsort	1.010	0.929 (↓7.89%)	0.897 (↓11.02%)	0.974 (↓3.40%)	**0.862 (↓14.51%)**
	Average	0.517	0.453 (↓12.47%)	0.445 (↓13.90%)	0.470 (↓9.16%)	**0.426 (↓17.70%)**

Table 4: Total power results. (Unit: mW)

Design	Original	PGM	PM[15]	GCM[9]	**Ours**
BOOM	11.23	10.91 (↓2.9%)	10.92 (↓2.8%)	11.16 (↓0.6%)	**10.82 (↓3.7%)**
Rocket	3.48	3.37 (↓3.3%)	3.40 (↓2.2%)	3.41 (↓2.1%)	**3.35 (↓3.8%)**

5.3 Sensitivity-based Optimization Results

In this section, we present the optimization results obtained from the sensitivity measurement method. For baseline metric-based methods that optimize glitch power purely based on metric rankings, we choose per-gate glitch power (denoted as PGM), power metric [15] (denoted as PM), and glitch criticality metric [9] (denoted as GCM). For sensitivity measurement, we draw 15 independent batches, each with 1000 gates. On BOOM, we sample gates based on PGM; on Rocket, we sample with PM as they give the best optimization results. We empirically set the attenuation factor γ in Equation (11) to 0.8.

Additionally, regarding the number of levels a glitch will propagate, we found that typically a glitch can propagate no more than five levels. The sensitivity computation yields 3590 gates and 1206 gates on BOOM and Rocket, respectively, with negative sensitivity (that should result in net glitch power reduction), which are optimized. For baseline methods, we optimize the top 4000 gates for BOOM and 2000 for Rocket ranked by the corresponding metric as these gates contribute to more than 80% of the sum of metrics in the netlist. Optimization results are shown in Table 3, where PGM, PM [15], and GCM [9] are baseline methods. For PGM and GCM baselines on BOOM, we can see that the glitch power increases for some benchmarks due to new glitches generated elsewhere in the design, which validates that solely relying on local metrics does not guarantee glitch power reduction. Our method can reduce glitch power by more than 10% on *qsort* and *median* benchmarks on BOOM and nearly 30% on *median* and *multiply* on Rocket. On average, our method reaches a 9.01% and 17.70% glitch power reduction on BOOM and Rocket, which are 5.2% and 3.8% higher than baseline methods. It

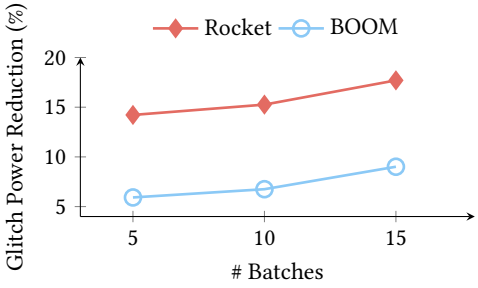

Figure 7: Number of batches for sensitivity measurement vs. Glitch power reduction.

can be seen that the sensitivity-based method provides a global view to correlate optimization candidates to total glitch power reduction.

In addition to glitch power, we compare the total power reduction for combinational gates in Table 4, as down-sizing and assigning high V_{th} also reduce dynamic and leakage power. The results are averaged across six benchmarks. Our method achieves the highest total power reduction compared with baseline methods (PGM, PM, GCM). Since gates that cause high glitch power usually have high switching activities, optimizing them also reduces total power. It is worth noting that this reduction seems minimal since the experiments are in TSMC N28, where leakage power dominates. However, in the latest technology node like FinFET, the relative contribution of dynamic power components has increased significantly due to better leakage control. So the improvement in total power is expected to be more in the latest technology nodes for this work.

The sensitivity we rely on for optimization is approximated by Monte-Carlo estimation. To understand the impact of the number of batches on sensitivity analysis and optimization results, we further sample different numbers of batches, including 5, 10, and 15 batches. The glitch power reductions are shown in Figure 7. Even only sample 5 batches, our method can reach larger glitch power reduction than baseline methods. As the number of sampled batches increases, the optimization leads to better power reductions. Therefore, it is advantageous to sample more batches to measure the sensitivity on more

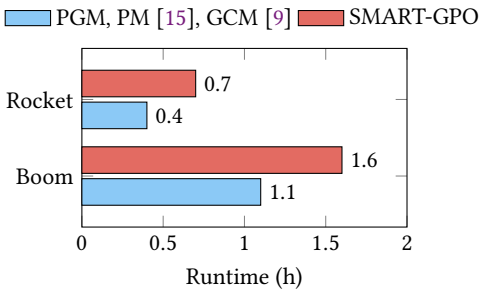

Figure 8: Comparison of runtime on BOOM and Rocket.

diverse gates for a better result whenever computational resources permit [32].

5.4 Overhead Analysis

Essentially, reducing glitch power by downsizing and increasing V_{th} inevitably incurs slack degradation. But both our work and previous works query the timing engine, and there is no timing violation after ECO.

In terms of the runtime, the runtime of baseline methods [9, 15] mainly involves an optimization process, in which the metrics are computed directly. In contrast, the proposed sensitivity-based optimization requires additional glitch power estimation steps, which may incur a longer runtime compared with baseline methods. In sensitivity measurement, optimizing sampled gates and running glitch power estimations in batches can be run in parallel. We only run power analysis with 0.1% cycles, which is accurate enough for sensitivity computation. The total runtime is about 18 minutes and 30 minutes longer than the baseline methods on Rocket and BOOM, respectively, as shown in Figure 8. In modern VLSI designs like BOOM (288K gates) and Rocket (114K gates), optimizing power is an increasingly challenging task, thus we believe this additional runtime is worthwhile. In the future, advanced techniques such as deep learning-based glitch power estimation can be utilized to eliminate the runtime issue, and we leave it as future work.

6 Conclusion

This paper presents SMART-GPO, a novel framework that efficiently and accurately estimates and optimizes glitch power at the gate level. Our framework performs power analysis on sampled cycles. It optimizes glitch power by measuring the sensitivity of gates w.r.t glitch power and sizing the most promising ones. The results show that our method can estimate glitch power more than 10 times faster than traditional methods while keeping the MAPE below 2% and Kendall's τ higher than 0.96 when we only use 1% cycles. Furthermore, with accurate glitch power estimation, we measure the sensitivity of gates and obtain better glitch power reductions, which reaches 9% and 17% glitch power reduction on BOOM and Rocket, respectively.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 62204066, No. 62304192, No. 62304197) and the Guangzhou Municipal Science and Technology Project (Municipal Key Laboratory Construction Project, Grant No.2023A03J0013).

References

[1] J. M. Rabaey, *Digital integrated circuits a design perspective*, 1999.
[2] "What is glitch power?" https://www.synopsys.com/ai/what-is-glitch-power.html.
[3] F. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE TCAD*, Feb. 1993.
[4] H. Mehta, "Accurate estimation of combinational circuit activity," in *Proc. DAC*, 1995.
[5] Yong Je Lim and M. Soma, "Statistical estimation of delay-dependent switching activities in embedded CMOS combinational circuits," *IEEE TVLSI*, 1997.
[6] "Primepower: Rtl to signoff power analysis datasheet," https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/primepower-ds.pdf.
[7] D. Rabe and W. Nebel, "New approach in gate-level glitch modelling," in *European Design Automation Conference with EURO-VHDL'96 and Exhibition*, 1996.
[8] A. Sayed and H. Al-Asaad, "A New Statistical Approach for Glitch Estimation in Combinational Circuits," in *Proc. ISCAS*, 2007.
[9] S. Bathla, R. M. Rao, and N. Chandrachoodan, "A Simulation-Based Metric to Guide Glitch Power Reduction in Digital Circuits," *IEEE TVLSI*, 2019.
[10] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanović, and B. Nikolić, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *Proc. MICRO*, 2020.
[11] K. Muthamizh Vithagan, V. Sundaresha, and J. Viraraghavan, "Geometric Programming Approach to Glitch Minimization via Gate Sizing," *IEEE TCAD*, 2023.
[12] J. Singh, V. Nookala, Z.-Q. Luo, and S. Sapatnekar, "Robust gate sizing by geometric programming," in *Proc. DAC*, 2005.
[13] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, 2007.
[14] M. Slimani, P. Matherat, and Y. Mathieu, "A dual threshold voltage technique for glitch minimization," in *Proc. ICECS*, 2012.
[15] L. Wang, M. Olbrich, E. Barke, T. Büchner, M. Bühler, and P. Panitz, "A gate sizing method for glitch power reduction," in *Proc. SOCC*, 2011.
[16] J. Daga and D. Auvergne, "A comprehensive delay macro modeling for submicrometer cmos logics," *IEEE Journal of Solid-State Circuits*, 1999.
[17] Z. Xie, X. Xu, M. Walker, J. Knebel, K. Palaniswamy, N. Hebert, J. Hu, H. Yang, Y. Chen, and S. Das, "Apollo: An automated power modeling framework for runtime power introspection in high-volume commercial microprocessors," in *Proc. MICRO*, 2021.
[18] D. Kim, J. Zhao, J. Bachrach, and K. Asanović, "Simmani: Runtime power modeling for arbitrary rtl with automatic signal selection," in *Proc. MICRO*, 2019.
[19] W. Fang, Y. Lu, S. Liu, Q. Zhang, C. Xu, L. W. Wills, H. Zhang, and Z. Xie, "Masterrtl: A pre-synthesis ppa estimation framework for any rtl design," in *Proc. ICCAD*, 2023.
[20] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu, "Cross-layer optimization for high speed adders: A pareto driven machine learning approach," *IEEE TCAD*, 2019.
[21] S. Roy, Y. Ma, J. Miao, and B. Yu, "A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders," in *Proc. ISLPED*, 2017.
[22] Y. Zhang, H. Ren, and B. Khailany, "Grannite: Graph neural network inference for transferable power estimation," in *Proc. DAC*, 2020.
[23] M. Rakesh, P. Das, A. Terkar, and A. Acharyya, "Graspe: Accurate post-synthesis power estimation from rtl using graph representation learning," in *Proc. ISCAS*, 2023.
[24] D. Kim, A. Izraelevitz, C. Celio, H. Kim, B. Zimmer, Y. Lee, J. Bachrach, and K. Asanovicc, "Strober: Fast and accurate sample-based energy simulation for arbitrary rtl," in *Proc. ISCA*, 2016.
[25] R. Wunderlich, T. Wenisch, B. Falsafi, and J. Hoe, "Smarts: accelerating microarchitecture simulation via rigorous statistical sampling," in *Proc. ISCA*, 2003.
[26] T. Wenisch, R. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. Hoe, "Simflex: Statistical sampling of computer system simulation," *Proc. MICRO*, 2006.
[27] C.-T. Hsieh, J. Cong, Z. Zhang, and S.-C. Chang, "Behavioral synthesis with activating unused flip-flops for reducing glitch power in fpga," in *Proc. ASPDAC*, 2008.
[28] Q. Dinh, D. Chen, and M. D. Wong, "A routing approach to reduce glitches in low power fpgas," in *Proc. ISPD*, 2009.
[29] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. CVPR*, June 2019.
[30] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American Statistical Association*, 1949.
[31] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih, "Monte carlo gradient estimation in machine learning," *Journal of Machine Learning Research*, 2020.
[32] P. Dellaportas and I. Kontoyiannis, "Control Variates for Estimation Based on Reversible Markov Chain Monte Carlo Samplers," *Journal of the Royal Statistical Society: Series B*, 2011.
[33] W. R. Knight, "A computer method for calculating kendall's tau with ungrouped data," *Journal of the American Statistical Association*, 1966.