GLOW: Glitch Optimization with Retiming for Low Power

Jiantao Liu
ETH Zurich
Zurich, Switzerland

Carmine Rizzi
ETH Zurich
Zurich, Switzerland

Pravriti Jaipuriyar
Cadence Design Systems
Munich, Germany

Ulrike Schorr
Cadence Design Systems
Munich, Germany

Sascha Richter
Cadence Design Systems
Munich, Germany

Ankush Sood
Cadence Design Systems
San Jose, USA

Lana Josipović
ETH Zurich
Zurich, Switzerland

ABSTRACT

With continuous technology scaling and increasingly compute-intensive workloads, power consumption has become a critical metric in digital design. While different EDA stages are increasingly addressing power, *retiming*—a widely-used logic synthesis optimization—has largely overlooked this aspect. In this paper, we introduce GLOW, an open-source retiming framework that simultaneously regulates timing and minimizes power by analyzing and reducing the circuit's switching activity. GLOW achieves power reductions of 8.01% and 20.37% compared to classic retiming and existing power-oriented retiming, respectively, under identical timing constraints, thus advancing logic synthesis power optimization.

1 INTRODUCTION

With technology scaling, power consumption has risen to the forefront of electronic design automation [16, 28], competing with performance and area as a primary design objective [3, 18, 31, 45, 46]. Retiming, one of the most widely used logic synthesis optimizations [4, 9, 38, 39, 42], traditionally aims to enhance performance by repositioning registers to meet the target clock period [12, 30]. Yet, register positions also impact power consumption by affecting the circuit's switching and load capacitances [32]. Standard retiming algorithms [30] entirely overlook power; a few efforts that consider it [13, 19, 24, 25, 29, 32] rely on local heuristics and fail to eliminate *glitches*—unwanted switches that cause power dissipation—thus missing wider-range power reduction opportunities.

In this work, we introduce GLOW, a retiming strategy for simultaneous performance and power optimization. The key idea of GLOW is to tackle the root cause of glitches, and prevent their appearance and propagation by strategic register retiming. By recongizing the conceptual similarity between delay propagation and glitch propagation, we formulate GLOW as an exact Mixed-Integer Linear Programming (MILP) problem that simultaneously accounts for the effects of register retiming on glitching, load capacitances, cirucit frequency, and area. We show that GLOW achieves 29.7% more power savings than a previous power-oriented retiming strategy [32] and 4.0% more power savings than state-of-the-art, power-agnostic retiming, while consistently meeting the same timing constraints. This demonstrates that GLOW comprehensively addresses the rising importance of power in digital circuit design.

In the rest of this paper, Section 2 presents the background and related works. Section 3 illustrates the limitations of existing power-oriented retiming approaches, and Section 4 discusses how GLOW overcomes them. Section 5 details our approach, and Section 6 experimentally demonstrates its superiority over existing techniques.

2 PRELIMINARIES

This section reviews the background and related work on retiming-based power optimization techniques.

2.1 Power Dissipation

Power dissipation in digital CMOS circuits is classified as [34]: (1) *static (or leakage) power*, dissipated when a logic circuit is powered but not active (i.e., the transistor is not switching), and (2) *dynamic power*, dissipated due to the switching activity of the transistors. Although technology scaling has amplified leakage concerns, dynamic power remains the predominant contributor to overall power [34, 45], making it the primary focus of GLOW.

The dynamic power at node x is expressed as $\mathrm{Pow}_{\mathrm{dyn}}(x) = \mathrm{Pow}_{\mathrm{external}}(x) + \mathrm{Pow}_{\mathrm{internal}}(x)$, where $\mathrm{Pow}_{\mathrm{external}}(x)$ refers to the dynamic power dissipated across the wires, and $\mathrm{Pow}_{\mathrm{internal}}(x)$ represents any dynamic power dissipated within the node:

$$\mathrm{Pow}_{\mathrm{external}}(x) = V_{DD}^2 \cdot f \cdot C^{\mathrm{x}}_{\mathrm{load}} \cdot \mathrm{switch}_{\mathrm{out}}(\mathrm{x}), \tag{1}$$

$$\mathrm{Pow}_{\mathrm{internal}}(x) = V_{DD}^2 \cdot f \cdot C^{\mathrm{x}}_{\mathrm{PD}} \cdot \mathrm{switch}_{\mathrm{in}}(\mathrm{x}), \tag{2}$$

where $\mathrm{switch}_{\mathrm{out}}(\mathrm{x})$ and $\mathrm{switch}_{\mathrm{in}}(\mathrm{x})$ are the *switching activity* of the node's output and input signals, respectively, expressed in units of switches per clock cycle, f is the design's operating frequency, $C^{\mathrm{x}}_{\mathrm{load}}$ is the *load capacitance* at the output of node x, $C^{\mathrm{x}}_{\mathrm{PD}}$ is the dynamic power dissipation capacitance of a node [37], and V_{DD} is the supply voltage [2, 35, 37]. In classic retiming [30], the goal is to meet a user-specified clock frequency, effectively keeping f constant. V_{DD} is also a constant determined by the technology node. Consequently, to reduce dynamic power at this stage, the primary goal is to decrease the load capacitances and switching activity.

2.2 Classic Retiming

Classic retiming [30] repositions registers in a sequential circuit to meet a specified clock period without altering the circuit's functionality. The circuit is modeled as a directed acyclic graph $G = (V, E, w, \delta)$, where V is the set of logic nodes, E is the set of

directed edges, $w(e)$ is the register count on each edge e, and $\delta(n)$ is the combinational delay of node n. A node's *fanins* are its direct input nodes and its *fanouts* are its direct outputs; a primary input (PI) has no fanins, whereas a primary output (PO) has no fanouts [13].

Retiming assigns an integer label $r(n)$ to each node $n \in V$, indicating how many registers move from the node's output edges to its input edges [30]. In the retimed design $G_r = (V, E, w_r, \delta)$, the new weight $w_r(e_{u,v})$ of any edge $e_{u,v} \in E$ is calculated as

$$w_r(e_{u,v}) = w(e_{u,v}) + r(v) - r(u), \forall e_{u,v} \in E. \quad (3)$$

To preserve the circuit's original functionality, these updated edge weights must remain nonnegative:

$$w_r(e_{u,v}) \geq 0, \forall e_{u,v} \in E. \quad (4)$$

To meet the target clock period $\mathrm{T_{clk}}$, retiming enforces:

$$r(v) - r(u) \geq 1 - W(u,v),\ \forall u, v \in V \mid D(u,v) > \mathrm{T_{clk}} \quad (5)$$

where $W(u,v)$ and $D(u,v)$ are precomputed matrices of path weights and delays in the original circuit. A path is a sequence of nodes connected by directed edges, and its total delay or weight is the sum of its nodes' delays or weights. The constraint above ensures that any combinational path has a delay below $\mathrm{T_{clk}}$ after retiming [30].

2.3 Retiming for Low Power

Classic retiming moves registers to meet a target clock period. However, changes in register location can affect the metrics from Equations 1 and 2: (1) Inserting a register at the node's output reduces its load capacitance ($C^{\mathrm{x}}_{\mathrm{load}}$), as the load capacitance of the register is typically smaller than that of the node's fanouts [32, 33, 43]. (2) Registers can unbalance input signal arrival times, which increases signal glitching and, consequently, the node's switching activity ($\mathrm{switch_{out}(x)}$ and $\mathrm{switch_{in}(x)}$).

Classic retiming targets only timing constraints and overlooks the impact of these factors on dynamic power. Several retiming methods aim to address power concerns [13, 19, 24, 25, 29, 32], but they are either scenario-specific or rely on iterative heuristics that lack global optimality. Jeddi et al. [25] consider rewiring and retiming but do not account for glitches. Fischer et al. [19] show manual examples of retiming's impact on glitches but lack automation. Yu-Lung et al. [24] detect glitches but do not quantify them. Chabini et al. [13] reduce supply voltage, incurring a more complicated backend process (i.e., IR-drop analysis). Lalgudi et al. [29] perform iterative latch-based retiming similar to Monteiro et al. [32].

Monteiro et al. [32] proposed an iterative method for power-aware retiming. The idea is to identify nodes with high power consumption due to high glitching and load capacitance (we refer to these nodes as *power hotspots*) and to retime registers to their outputs, thus preventing glitch propagation and lowering the dynamic power. To this end, they define the node's *glitch power reduction* as:

$$\mathrm{Pow_{red}}(x) = E_{\mathrm{glitch}}(x) \cdot (C^{\mathrm{x}}_{\mathrm{load}} + \sum_{\mathrm{y} \in \mathrm{fanouts(x)}} s_{\mathrm{x,y}} \cdot C^{\mathrm{y}}_{\mathrm{load}}), \quad (6)$$

where $E_{glitch}(x)$ is the expected number of glitches per clock cycle at node x's output edges, computed from the difference between the switching activity obtained from functional and timing-based simulation. The sensitivity of node y relative to its fanin node x, $s_{x,y}$, represents the probability of having a switch on the output of node y caused by a switch on the output of node x [32]:

$$s_{x,y} = \frac{P(x \updownarrow \ \wedge\ y \updownarrow)}{P(x \updownarrow)}. \quad (7)$$

Here, $P(x \updownarrow)$ is the probability of having a switch at node x, and $P(x \updownarrow \wedge y \updownarrow)$ is the probability of having a switch at y caused by a switch at x [20]. Thus, $\mathrm{Pow_{red}}(x)$ is the sum of the power dissipation at a node due to glitching (i.e., a portion of the external dynamic power from Equation (1) caused by glitches) and the consequent probabilistically-determined power dissipation of its fanouts; a high $\mathrm{Pow_{red}}(x)$ indicates that x is a power hotspot and that retiming a register to its output could reduce power.

To reduce dynamic power, Monteiro et al. [32] iteratively perform the following steps: (1) Compute $E_{glitch}(x)$ and $\mathrm{Pow_{red}}(x)$ for each node using Equation (6). (2) Identify a set of nodes, selecting at most one node from each PI-PO path, that maximizes the cumulative glitching power reduction. (3) Insert the registers of the pipeline stage at the PIs, retime them to the outputs of the selected nodes, and fix their positions. Finally, remaining registers are retimed to minimize register count and meet timing constraints.

Each iteration therefore reduces the power consumption of a set of hotspot nodes that dissipate power due to glitching. Although this ultimately lowers the circuit's total dynamic power, the iterative nature of the algorithm may prevent it from achieving the global optimum. Also, while targeting hotspots reduces their glitch propagation (and, consequently, their power), it does not eliminate the source of the glitch—other nodes may still be affected by it and dissipate power accordingly. We investigate this effect next.

3 IS ELIMINATING POWER HOTSPOTS THE WAY TO REDUCE POWER?

Figure 1(a) shows a six-node circuit with two PIs and three POs, where the clock period spans five *steps*—discrete time intervals within the clock period (referred to as *delta time steps* in discrete-time simulation [5]). The input capacitance of each node is indicated by its color. Assume that all node delays are equal to 1 step and a switch occurs at the registers at the PIs at step 0 of clock cycle 1; node D performs an XOR, while all other nodes simply forward their input values to the output. The waveform shows the signal through the circuit, and the table next to it summarizes the switching counts, load capacitances, and external dynamic power consumption.

The combinational paths to node D have an unbalanced combinational delay, and the switches of the registers at A and B arrive at D's inputs at different times, which causes a glitch to appear at D's output. This glitch propagates to all succeeding nodes; this is reflected in the external power consumption of 140. Any method that focuses on hotspot optimization (such as Monteiro et al. [32]) will identify E as the hotspot, as its power reduction potential is the largest; consequently, the registers will be retimed to E's outputs, as shown in Figure 1(b). As the waveform and table in the figure show, this prevents the glitch from propagating to E's successors and reduces E's load capacitance, thus reducing the external power to 36.

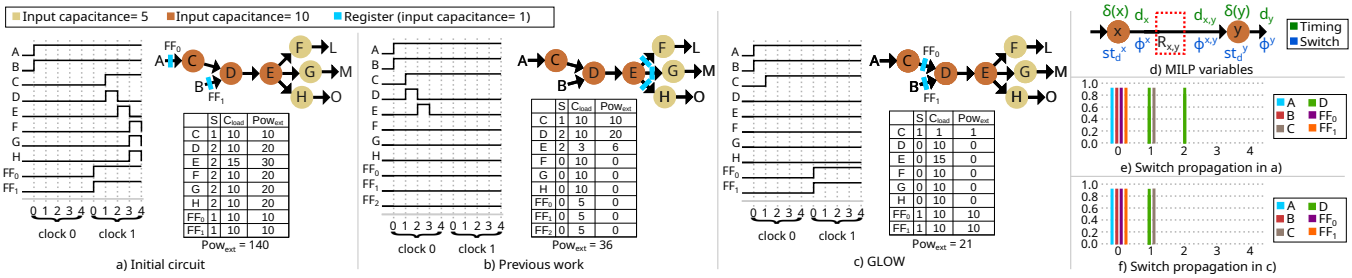


Figure 1: (a) Initial circuit with registers at the PIs; node D switches due to unbalanced combinational paths, and the external dynamic power is 140. (b) Monteiro et al.'s retiming [32] places registers after the highest-power node (E); yet, unaddressed switching at D still leads to an external power of 36. (c) GLOW (our approach) places registers at D's input; this eliminates excess switching and reduces the circuit's external power to 1 with fewer registers. (d) GLOW's MILP variables. (e) and (f) Switch propagation probabilities for a subset of the nodes in the circuits in (a) and (c).

Although this is an improvement, notice in the waveform that D and E still glitch and dissipate power accordingly. A superior solution is to retime the registers to D's inputs, as Figure 1(c) suggests, and eliminate the glitch by balancing the combinational delays to D; the entire circuit is now glitch-free, and its external power consumption is minimal. This solution is beyond the capabilities of local hotspot elimination and is exactly what we aim to achieve here.

4 GLOW: ADDRESSING THE POWER PROBLEM AT THE SOURCE

Hotspot elimination focuses on removing the effects of the glitch, rather than removing the glitch itself—in other words, it fixes the consequence of the problem, instead of addressing its root cause. Our goal is to address the power problem at the root: we seek to identify a retiming solution that prevents the appearance and propagation of power-expensive glitches. Although this goal is highly intuitive, reasoning about the effects of register retiming on circuit glitching is challenging—the register positions dictate the formation, propagation, and suppressing of glitches, and any register movement via retiming invalidates prior glitching assumptions. Retiming also affects other power, performance, and area metrics, such as load capacitances, combinational delays, and register count. Thus, we need a systematic way to capture the interactions of these aspects and reason about their changes due to retiming.

GLOW is a retiming approach based on mixed-integer linear programming that simultaneously tackles power, performance, and area: (1) GLOW reasons about circuit switching and balances signal arrival times on reconvergent paths to minimize glitch appearance and propagation. (2) GLOW accounts for load capacitance changes (and consequent dynamic power variations) caused by register retiming. (3) GLOW controls combinational delay propagation to meet the target clock period. (4) GLOW minimizes the number of registers required to achieve performant and power-efficient circuits, thus reducing the related area cost. By jointly addressing these aspects, GLOW identifies retiming solutions that minimize dynamic power consumption at a given clock period constraint. The rest of this paper formalizes GLOW and illustrates its benefits.

5 GLOW'S FORMULATION

The main idea behind GLOW's unified retiming for performance and power is that switch and combinational delay propagation are conceptually similar—both propagate through combinational circuit nodes and can be regulated by register placement. Thus, we can borrow well-established delay models to reason about switches as well. Yet, there are two fundamental differences: (1) In delay propagation, the *latest* signal arriving at a node's inputs is the one that dictates the node's output delay; in switch propagation, *all* signals arriving at a node's inputs dictate its output switching. (2) Delays *always* propagate through combinational nodes; switch propagation is *probabilistic* and dependent on the node functionalities. In this section, we first discuss the delay model that we employ in GLOW; we then build upon it to devise an analogous switch propagation model that accounts for the abovementioned differences. We then use these insights to combine classic delay-oriented retiming with dynamic power optimization in a single MILP problem. We employ the variables and constants shown in Table 1 and Figure 1(d).

5.1 Timing Regulation Constraints

GLOW employs standard retiming constraints (i.e., Equations (3) and (4)) to ensure that the retiming is legal. To regulate the circuit's timing, we devise a delay propagation model inspired by HLS scheduling [26, 36, 41]; as we will later see, our model serves as a foundation for an analogous reasoning about switch propagation.

To connect register retiming to our timing model (and, later, switch propagation), we introduce binary variables $R_{x,y}$ (defined in Table 1) to indicate the presence of at least one register on edge $x \rightarrow y$. Register presence relates to the retiming edge weights as:

$$R_{x,y} = \begin{cases} 1 & \text{if } w_r(e_{x,y}) > 0 \\ 0 & \text{else,} \end{cases} \tag{8}$$

where $w_r(e_{x,y})$ is the edge weight of the retimed circuit and can be calculated using Equation (3). We linearize this relationship as:

$$w_r(e_{x,y}) \leq M \cdot R_{x,y},$$
$$w_r(e_{x,y}) \geq R_{x,y}$$

where M is a sufficiently large positive constant integer (i.e., the number of the circuit's pipeline stages). In Figure 1(a), assuming 2 pipeline stages, the equations for the edge between D and E are:

Table 1: Variable and Notation Declarations.

Variables	Type	Description
$R_{x,y}$	$\{0, 1\}$	The presence of a register on edge $e_{x,y}$.
d_x	$\mathcal{R}_+$	Max. combinational path delay at node x's output.
$d_{x,y}$	$\mathcal{R}_+$	Combinational path delay at node y's input on edge $e_{x,y}$.
$r(x)$	$\mathcal{Z}$	The retiming variable for node x.
$w_r(e_{x,y})$	$\mathcal{Z}_{\geq 0}$	The weight of edge $e_{x,y}$.
ϕ^x	$\mathcal{R}_+$	The expected switches at node x's output.
ϕ^x_i	$\mathcal{R}_{[0,1]}$	The probability of node x's output switching at i-th time step.
$\phi^{x,y}_i$	$\mathcal{R}_{[0,1]}$	The probability of y's input on edge $e_{x,y}$ switching at i-th time step.
C^x_{load}	$\mathcal{R}_+$	The load capacitance of node x.
Constants		
$\delta(x)$	$\mathcal{R}_+$	The node delay of node x.
$s_{x,y}$	$\mathcal{R}_+$	Sensitivity of node y relative to node x.
C^x_{in}	$\mathcal{R}_+$	The input capacitance of node x.
$P(\text{FF} \updownarrow)$	$\mathcal{R}_{[0,1]}$	Probability that a FF generates a switch.
$P(\text{PI} \updownarrow)$	$\mathcal{R}_{[0,1]}$	Probability that PI generates a switch.
T_{clk}	$\mathcal{R}_+$	The desired clock period.
τ	$\mathcal{R}_+$	The granularity of a single step.
st^x_{d}	$\mathcal{Z}_{\geq 0}$	The first possible step for node x during switch propagation.
α	$\mathcal{R}_{[0,1]}$	Dynamic power coefficient in the objective function.
$w(e_{x,y})$	$\mathcal{Z}_{\geq 0}$	Weight of edge $e_{x,y}$. in the original circuit.
Notations		
$e_{x,y}$		An edge from x to y.

$w_r(e_{D,E}) \leq 2 \cdot R_{D,E}$, and $w_r(e_{D,E}) \geq R_{D,E}$. If the retiming weight is positive, the value of $R_{D,E}$ will be set to 1, indicating that one or more registers are placed on the edge; if the weight is zero, the equations will also set $R_{D,E}$ to zero, indicating register absence.

For each node y, we define the following constraints:

$$d_{x,y} \geq d_x - 2 \cdot \text{T}_{\text{clk}} \cdot R_{x,y}, \ \forall x \in fanins(y), \tag{9}$$

$$d_y \geq d_{x,y} + \delta(y), \ \forall x \in fanins(y), \tag{10}$$

$$\delta(y) \leq d_y \leq \text{T}_{\text{clk}}. \tag{11}$$

As shown in Figure 1(d), $d_{x,y}$ is the combinational path delay at y's input on edge $e_{x,y}$, d_y is the maximum combinational path delay at node y's output, $\delta(y)$ is the combinational delay of y, T_{clk} is the target clock period, $R_{x,y}$ is the presence of a register on edge $e_{x,y}$, and $fanins(y)$ is the set of all fanin nodes of y. When $R_{x,y} = 0$, Equation (9) propagates the combinational delay from x's output to y's input. Since $2 \cdot \text{T}_{\text{clk}}$ must be greater than the sum of the delays on the combinational path to honor the clock period constraint, the right-hand side of the equation will become negative when $R_{x,y} = 1$, thus disabling the delay propagation. Equation (10) propagates the path delays from the node's fanins to its output, while Equation (11) constrains the combinational delay at a node's output to be at least the delay of the node itself and at most the target clock period.

Assuming $\text{T}_{\text{clk}} = 5$ and $\delta(E) = 1$, the timing regulation constraints for node E in the example in Figure 1(a) are: $d_{D,E} \geq d_D - 2 \cdot 5 \cdot R_{D,E}$, $d_E \geq d_{D,E} + 1$, and $1 \leq d_E \leq 5$. The first two equations dictate the combinational delay propagation from D's output to E's output. The third equation specifies that the minimal combinational delay at E's output is the delay of E itself, and the maximal delay is the clock period target of 5. In case the propagation delay from node D (d_D) makes the delay at E's output (d_E) larger than the target (T_{clk}), $R_{D,E}$ would be set to 1 to stop the delay propagation (the first equation). The delay at E's output would be set to its own delay of 1 by the third equation.

GLOW supports finer-grain timing models by simply substituting per-node delays in Equations (9) to (11) with pin-to-pin delays.

5.2 Switch Propagation Constraints

In delay propagation, a node's output delay is determined by the largest input arrival time. The situation with switches is different: *all* switches arriving at different combinational times to a node *might* propagate to the node's output (as determined by the node's functionality and input signal values). This was the case for node D in Figure 1a: since the signals of FF1 and C arrived in consecutive time steps (defined in Section 3), D first switched from 0 to 1 and then back to 0 within the same clock cycle.

To capture such switch propagations, our MILP includes the following: (1) We reason about signal propagations in $\text{Step}_{\max}$ steps within the clock period; $\text{Step}_{\max}$ is an input constant to our problem whose value impacts the model's granularity and precision. (2) We define ϕ^x_i as the probability that a switch occurs at a node's output at the i-th time step of the clock period; this enables us to capture all possible switches of x within a single clock cycle and to characterize them by their individual occurrence probabilities.

We depict the propagation of a switch between nodes as:

$$\phi^{x,y}_i \geq \phi^x_i - R_{x,y}, \forall x \in fanins(y), \forall i \in [0, \text{Step}_{\max}), \tag{12}$$

where ϕ^x_i is defined above and $\phi^{x,y}_i$ is the probability that the input of y on the input edge $e_{x,y}$ switches at the i-th time step. This equation describes the propagation of a switch from node x to node y in the i-th time step and is conceptually similar to delay propagation in Equation (9), as shown in Figure 1(d): If no register is placed on the edge between x and y (i.e., $R_{x,y} = 0$), the switch from x propagates to y; thus, the probability of a switch occurring at the input of y is set to the probability of a switch occurring at the output of its fanin x. Otherwise, if $R_{x,y} = 1$, the switch propagation is blocked by the register and the constraint is disabled, indicating that the switching of y's input is independent of the switching of x.

We depict the propagation of a switch through each node y as:

$$\phi^y_{i+\text{st}^y_{\text{d}}} \geq \phi^{x,y}_i \cdot s_{x,y}, \forall x \in fanins(y), \tag{13}$$

where $s_{x,y}$ is the sensitivity of node y to its input on edge $e_{x,y}$ (see Equation (7)), st^y_{d} is the number of time steps necessary for a switch to traverse node y, and $i \in [0, \text{Step}_{\max} - \text{st}^y_{\text{d}})$. This equation captures how switches arriving at y via its fanin edges affect the switching at y's output and is conceptually similar to Equation (10). Like any other signal, the time it takes a switch to propagate through a node is dictated by a node's delay; thus, the switch probabilities of the fanin edges propagate through node y after st^y_{d} steps, where st^y_{d} is directly obtainable from the y's combinational delay and the step duration. The probability that the input switch appears at y's output is dictated by its functionality and reflected by its sensitivity.

If multiple input switch probabilities propagate to the output in different steps, they will be captured by different step-wise output probabilities, indicating the possibility of having multiple output switches. If multiple input switch probabilities propagate to the output in the same step, the output probability of that step will be set to the largest among the sensitivity-weighted input probabilities, reflecting the highest chance of observing an output switch and, consequently, the highest power impact. Other probability propagation schemes are possible and can be analogously supported.

We describe the initiation of a switch at registers and PIs as:

$$\phi^{x,y}_0 \geq R_{x,y} \cdot P(\text{FF} \updownarrow), \forall y \in V, \forall x \in fanins(y), \tag{14}$$

$$\phi_0^y \geq P(\text{PI} \updownarrow), \forall y \in \text{PIs}. \quad (15)$$

The first equation describes a switch initiation with probability $P(\text{FF} \updownarrow)$ at a register, reaching y at step $i = 0$; the second defines a switch initiation with a probability $P(\text{PI} \updownarrow)$ at a PI at step $i = 0$.

In the rest of this section, we apply the switch propagation equations to the example of Figure 1(a) and show how we can exploit them to reduce switching (and, consequently, power).

To describe the switch propagation from PI A to node C, GLOW employs Equation (12) and (14) as follows: $\phi_0^{A,C} \geq \phi_0^A - R_{A,C}$ and $\phi_0^{A,C} \geq R_{A,C} \cdot P(\text{FF} \updownarrow)$. These equations describe the switching probability at C's input: If no register is placed on the input edge of C (i.e., $R_{A,C} = 0$), the switch from PI A will propagate to C via the first constraint, whereas the second constraint will be disabled. Otherwise, if $R_{A,C} = 1$, the first constraint will be disabled, and C's input switching will be dictated by the register's switching. The input switching of C at i-th time step may propagate to its output, as described by Equation (13): $\phi^C_{i+\text{st}^\text{C}_\text{d}} \geq \phi_i^{A,C} \cdot s_{A,C}$. Here, $\text{st}^\text{C}_\text{d}$ corresponds to C's delay expressed in steps, and $s_{A,C}$ is C's sensitivity to a switch at its input. Analogous constraints can be expressed for all edges and nodes of the circuit in Figure 1(a).

We will now compare the switching of the circuits in Figure 1(a) and (c), assuming that all nodes have a delay of 1 step, all PIs have a switching probability of 1, all registers have a switching probability of 0.9, and the sensitivities of the considered nodes are all 1.

In Figure 1(a), the registers are placed directly after the PIs, hence, $R_{A,C} = 1$, and $R_{B,D} = 1$. Due to the registers, the switching of A and B does not propagate to C and D; instead, their input switching is determined by the switching of the registers: $\phi_0^{A,C} \geq 1 \cdot 0.9$, $\phi_0^{B,D} \geq 1 \cdot 0.9$. The input switch will propagate through C one step later, with a probability determined by C's sensitivity and the corresponding path and node delay as $\phi_1^C \geq \phi_0^{A,C} \cdot 1$, and then analogously through D in step 2 as $\phi_2^D \geq \phi_1^{C,D} \cdot 1$. The switch from the register after B will propagate through D with a probability $\phi_1^D \geq \phi_0^{B,D} \cdot 1$. The switching probability at D is 0 in all other steps.

Figure 1(e) shows the switch probabilities of each edge in different time steps: D's output might switch in two consecutive time steps due to the different arrival times of the switches at its inputs, which may have a detrimental effect on its power consumption.

Consider now the circuit in Figure 1(c), with $R_{C,D} = 1$, and $R_{B,D} = 1$. In this case, both inputs of D can switch only in step 0, as $\phi_0^{C,D}$ and $\phi_0^{B,D}$ are the only positive probabilities at D's inputs. Hence, the switch propagations through D are: $\phi_1^D \geq \phi_0^{C,D} \cdot 1$ and $\phi_1^D \geq \phi_0^{B,D} \cdot 1$. This indicates that, due to the identical arrival times of the switches at D's inputs, D can switch only once in time step 1, as shown in Figure 1(f). Consequently, the dynamic power consumption of D will be lower than in the circuit of Figure 1(a), as discussed in Section 3. GLOW's switch propagation equations identify redundant switches as in Figure 1(e) and eliminate them by placing registers to equalize input signal arrival times, resulting in low-switch and low-power circuits as in Figure 1(c).

5.3 Dynamic Power Consumption

The previous section devised a way to analyze switch propagation and to correlate it with register placement. We now build upon these insights to calculate the circuit's dynamic power, addressing first the contributions from combinational gates and then from sequential elements.

Dynamic Power of Combinational Nodes. We first define the load capacitance of node x as follows:

$$C^\text{x}_\text{load} = \sum_{y \in fanouts(x)} (C^\text{y}_\text{in} \cdot (1 - R_{x,y}) + C^\text{FF}_\text{in} \cdot R_{x,y})), \quad (16)$$

where $fanouts(x)$ is the set of fanouts of x, C^y_in is the input capacitance of node y, and C^FF_in is the input capacitance of a register on the edge from x to y. If a register is placed on the edge (i.e., $R_{x,y} = 1$), the edge's load capacitance will be set to the capacitance of the register; otherwise, it will be set to the capacitance of the succeeding node y. Wire capacitance can be incorporated as an additional capacitance. In Figure 1(a), the load capacitance of D is the following: $C^\text{D}_\text{load} = (C^\text{E}_\text{in} \cdot (1 - R_{D,E}) + C^\text{FF}_\text{in} \cdot R_{D,E})$. If $R_{D,E}$ is set to 1, the total load capacitance of D becomes: $C^\text{D}_\text{load} = C^\text{FF}_\text{in}$.

We approximate the node's output switching as a sum of its switching probabilities across all time steps in a clock cycle:

$$\phi^x = \textstyle\sum_{i=0}^{\text{Step}_\text{max}} \phi_i^x. \quad (17)$$

The above-calculated load capacitance and this switch count approximation allow us to calculate $\text{Pow}_\text{external}(x)$ using Equation (1). Analogously, by approximating the node's input switching as a sum of its input switching probabilities $\phi_i^{x,y}$ and using the technology-defined constant C^x_PD, we calculate the internal power using Equation (2). We can then compute the circuit's total power consumption generated by combinational gates:

$$\text{Pow}_\text{comb} = \textstyle\sum_{x \in V} (\text{Pow}_\text{external}(x) + \text{Pow}_\text{internal}(x)). \quad (18)$$

We illustrate the external dynamic power computations of nodes C and D of the circuit in Figure 1 (the procedure for internal dynamic power is similar), based on the switching analysis from the previous section and assuming the same switch probabilities and sensitivities as before. With the register configuration in Figure 1(a), ϕ_1^C is the only switch probability bigger than 0 for node C. For node D, there are two non-zero switch probabilities, ϕ_1^D and ϕ_2^D. Hence, $\text{Pow}_\text{external}(C) + \text{Pow}_\text{external}(D) = \phi_1^C \cdot C^\text{C}_\text{load} \cdot f + (\phi_1^D + \phi_2^D) \cdot C^\text{D}_\text{load} \cdot f = 0.9 \cdot 10 \cdot f + (0.9 + 0.9) \cdot 10 \cdot f = 27 \cdot f$. In contrast, with the register configuration in Figure 1(c), the register on the edge between C and D reduces the number of switches at D to a single switch in step 1. Hence, $\text{Pow}_\text{external}(C) + \text{Pow}_\text{external}(D) = \phi_1^C \cdot C^\text{C}_\text{load} \cdot f + \phi_1^D \cdot C^\text{D}_\text{load} \cdot f = 0.9 \cdot 1 \cdot f + 0.9 \cdot 10 \cdot f = 9.9 \cdot f$. Considering only nodes C and D, the dynamic power of circuit (c) is lower than that of (a); as the goal of GLOW is to minimize dynamic power, it will retime the registers to the configuration of circuit (c).

Dynamic Power of Sequential Nodes. Registers constitute another significant source of dynamic power [40]. Like for combinational nodes, their power consumption can be decomposed into two components, internal and external power. For a register $FF_{x,y}$ on an edge between node x and y, we compute the external power using Equation (1), where the load capacitance is C^y_in, and the corresponding output switching activity is $P(\text{FF} \updownarrow)$. Similarly, we employ Equation (2) for internal power, where the internal capacitance of a register is C^FF_PD, which is a technology-defined constant.

The circuit's total sequential dynamic power is then:

$$\mathrm{Pow_{seq}} = \sum_{e_{x,y} \in E} R_{x,y} \cdot (\mathrm{Pow_{external}(FF_{x,y}) + Pow_{internal}(FF_{x,y})}) \quad (19)$$

where $R_{y,x}$ ensures that the power contribution is counted only on edges occupied by a register. For instance, the dynamic power of register $\mathrm{FF_0}$ of Figure 1(c) is equal to $\mathrm{Pow_{external}(FF_0)} + \mathrm{Pow_{internal}(FF_0)} = C_{\mathrm{in}}^{\mathrm{D}} \cdot P(\mathrm{FF}\ \updownarrow) \cdot f + C_{\mathrm{PD}}^{\mathrm{FF}} \cdot f = 10 \cdot 0.9 \cdot \mathrm{f} + 20 \cdot \mathrm{f} = 29 \cdot \mathrm{f}$.

This detailed power analysis shows GLOW's ability to accurately estimate the number and interactions of switches across the circuit, while also considering the varying capacitive loads; as a result, it can accurately estimate the circuit's power and optimize it by register retiming.

5.4 Objective Function

GLOW employs standard retiming constraints (Equations 3 and 4) for valid retiming, timing regulation constraints (Section 5.1) to meet the desired timing, and switch propagation constraints (Section 5.2) to calculate dynamic power (Section 5.3). GLOW's main optimization goal is to reduce the total dynamic power consumption with the minimal number of registers while honoring a timing constraint. Hence, the objective function of GLOW is the following:

$$\min(\alpha \cdot (\mathrm{P_{comb}} + \mathrm{P_{seq}}) + \beta \cdot \sum_{e \in E} w_r(e)), \quad (20)$$

where α and β are user-defined coefficients for tuning the relative importance of dynamic power versus the total register count. If α is set to 0, GLOW's formulation aligns with classic retiming that minimizes the register count under a timing constraint.

6 EVALUATION

This section shows that GLOW achieves timing-compliant, low-power circuits. We will open-source GLOW and our benchmarks.

6.1 Comparison of Retiming Formulations

We first compare GLOW with classic retiming, which minimizes register count, and the technique proposed by Monteiro et al. [32].

We evaluate a subset of the ITC'99 benchmarks [15] and the AES [23] benchmark; their gate counts are reported next to the benchmark names in Table 2. Our larger benchmarks are comparable to retiming partitions that industry tools typically retime in isolation [6, 7, 14, 17, 21]; thus, our benchmark scale is realistic and practically relevant. The number of pipeline stages and gates of each benchmark is consistent across all three approaches, and all target and satisfy the same timing constraint (specified in the table per benchmark). We run our experiments on a 75-core AMD EPYC-Milan machine with 581 GB memory. We use Genus v25.11 [9] for logic synthesis, then extract the mapped netlists, annotated with input and internal capacitances, combinational delays, and node sensitivities. We formulate classic retiming as an ILP, Monteiro's retiming as an iterative ILP, and GLOW as an MILP; we solve them using Gurobi v10.0.1 [22], setting a time limit of one hour. We measure the retimed circuits' power post-synthesis using Joules v25.11 [11], and use ModelSim v10.7b for simulating all design versions with the same randomly generated input test vectors. We run logical equivalence checking between the initial and the retimed

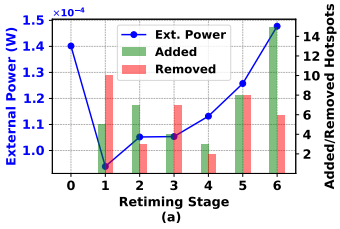


Figure 2: Monteiro's power consumption increases with retiming iterations.

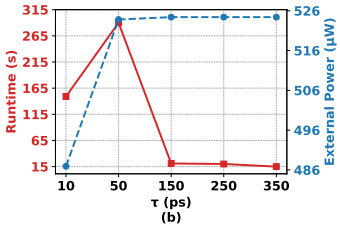


Figure 3: Tradeoff of MILP runtime and external power for different granularities τ in the AES benchmark.

circuits using Cadence Conformal [8] and confirm their functional equivalence. We use Nangate45PDK(v1.2) [1]. For each benchmark, we set a clock period target (T_{clk}) near the minimum achievable. We first set the granularity (τ) at 10 ps and then progressively increase it in 10 ps intervals until we find a τ value that yields a feasible solution within the timeout limit. We set $\alpha = 1$ and $\beta = 10^{-5}$.

Table 2 shows the achieved register count, internal and external power, and solver runtime for the three retiming methods. **GLOW consistently reduces internal and external dynamic power compared to both classical retiming and Monteiro's method**. This is because classic retiming ignores power, whereas Monteiro optimizes only power hotspots (see Section 3) and ignores internal power; in contrast, GLOW comprehensively analyzes and reduces the circuit's switching and, consequently, internal and external power. Surprisingly, Monteiro often yields worse power results even than classic retiming: while each of its iterations removes some hotspots, it also introduces new ones, thus increasing total power, as shown in Figure 2. GLOW's register count, generally, matches or marginally increases compared to classic retiming (whose register count is minimal for the given timing constraint), whereas Monteiro's iterative nature significantly worsens this metric. Naturally, GLOW's improvements come with an increased MILP complexity and runtime. Yet, the runtimes remain competitive and acceptable even for large benchmarks.

6.2 Granularity Selection

We examine the impact of the time step duration τ, dictating the number of time steps $Step_{max}$ (Section 5.2) and the MILP's granularity, on GLOW's runtime and external dynamic power.

We investigate the AES benchmark, whose smallest gate delay is around 20 ps, and the delay differences between gates are as small as

Table 2: Retiming comparison (post-synthesis results): classic retiming with register count minimization (C), Monteiro et al. [32] (M), and GLOW (G). GLOW achieves notable power improvements over both classic and Monteiro while, generally, also reducing the register count.

Bench. (#gates)	T_clk (ns)	$\tau(\mu s)$	Register Count					Internal Dynamic Power (μW)					External Dynamic Power (μW)					Total Dyn. Power		Retiming runtime (s)		
			C	M	G	C vs. G	M vs. G	C	M	G	C vs. G	M vs. G	C	M	G	C vs. G	M vs. G	C vs. G	M vs. G	C	M	G
AES (1084)	2.5	10.0	321	324	322	0.3%	-0.6%	2161.5	2327.7	2015.6	-6.8%	-13.4%	544.8	534.4	486.2	-10.8%	-9.0%	-7.6%	-12.6%	37.0	191.0	154.1
b03 (137)	1.0	10.0	30	53	30	0.00%	-43.40%	285.0	488.1	283.9	-0.39%	-41.84%	144.9	144.6	143.5	-0.97%	-0.76%	-0.58%	-32.45%	0.3	6.5	7.3
b04 (363)	1.0	10.0	78	78	78	0.00%	0.00%	521.1	521.2	514.1	-1.34%	-1.36%	103.6	103.6	101.0	-2.51%	-2.51%	-1.54%	-1.55%	3.0	6.4	30.1
b08 (1278)	2.0	40.0	62	147	63	1.61%	-57.14%	796.7	1173.4	781.4	-1.92%	-33.41%	710.1	901.7	688.9	-2.99%	-23.60%	-2.42%	-29.15%	11.4	207.6	375.0
b11 (3482)	2.0	50.0	286	288	286	0.00%	-0.69%	1362	1363.2	1315.6	-3.41%	-3.49%	797.1	798.2	746.5	-6.35%	-6.48%	-4.49%	-4.59%	16.7	79.6	3600.0
b14 (2701)	2.5	30.0	215	280	217	0.93%	-22.50%	1411.3	1506.7	1323.2	-6.24%	-12.18%	1182.5	1141.9	1066.8	-9.78%	-6.58%	-7.86%	-9.76%	22.1	221.2	697.9
b20 (5330)	4.0	30.0	428	553	429	0.23%	-22.42%	1532.6	1682.1	1285.5	-16.12%	-23.58%	1321.4	1208.6	928.9	-29.70%	-23.14%	-22.41%	-23.4%	124.8	1253.7	3600.0
b21 (5359)	4.0	30.0	428	769	430	0.47%	-44.08%	1562.5	2063.7	1360.0	-12.96%	-34.1%	1243.6	1167.3	987.1	-20.63%	-15.44%	-16.86%	-27.36%	112.6	2070.2	3556.8
b22 (7406)	4.0	40.0	610	1101	612	0.33%	-44.41%	1907.7	3392	1775.9	-6.91%	-47.64%	1431.6	1900.0	1267.7	-11.45%	-33.28%	-8.86%	-42.49%	194.6	2838.7	3600.0
Avg.						**0.43%**	**-26.14%**				**-6.23%**	**-23.45%**				**-10.57%**	**-13.42%**	**-8.01%**	**-20.37%**			

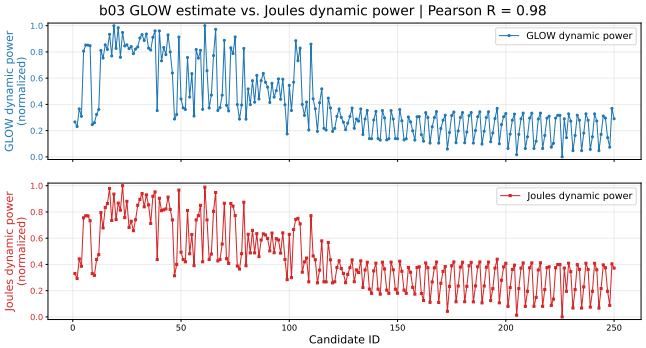
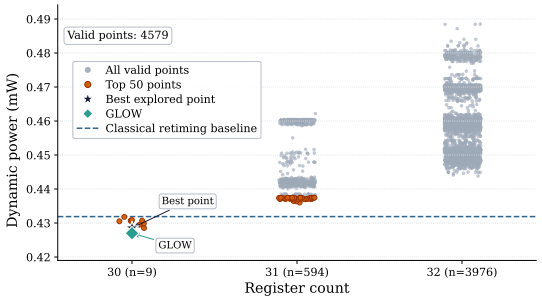
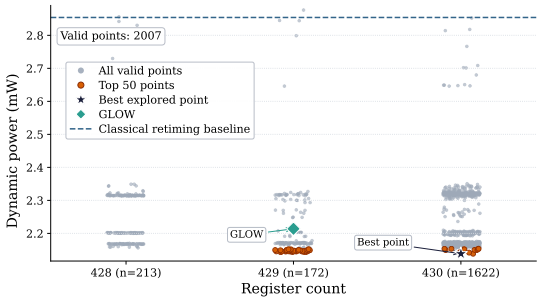


Figure 4: Correlation between GLOW's dynamic power estimate and Joules power analysis for 250 randomly generated valid retiming solutions (R = 0.98). Power values are normalized to the maximum power, respectively.



(a) b03 exploration: dynamic power vs. register count



(b) b20 exploration: dynamic power vs. register count

Figure 5: Time-bound exploration for 24 hours.

1 ps. Figure 3 shows GLOW's MILP runtime and external dynamic power after synthesis for different values of τ and a clock period target of 2.5 ns. As expected, decreasing the granularity (increasing τ) generally reduces the solver runtime due to the reduced number of MILP variables (there are occasional exceptions—for instance, the runtime at granularity 50 is higher than that at adjacent values; this is because MILP runtimes are not always perfectly proportional to the problem size but are also impacted by the computed values). However, this coarser modeling also leads to less accurate switching activity estimates, resulting in higher external power. We observe similar power-runtime tradeoffs with τ variations in other benchmarks as well. **This illustrates GLOW's flexibility in trading off optimization accuracy and computational runtime.**

6.3 Dynamic Power Model Correlation

We evaluate the correlation between the dynamic power estimated by GLOW and the actual dynamic power reported by Joules post-synthesis [11]. For this experimental evaluation, we use the *b03* benchmark from Table 2. Specifically, we consider 250 randomly generated valid retiming solutions for *b03* and inject each solution into the MILP formulation of GLOW to obtain the corresponding total dynamic power estimate. We then evaluate the same retiming solutions using Joules.

The results are shown in Figure 4. All power values are normalized to the maximum power value of each respective power computation method. As observed, there is a strong correlation between the estimates produced by GLOW and the values reported by

Table 3: Comparison of classical retiming (C) with GLOW (G) and with time-bound (24 hours) exploration search (ES).

Bench.	Internal Dyn. Power (μW)					External Dyn. Power (μW)					Runtime (h)	
	C	G	ES	C vs. G	C vs. ES	C	G	ES	C vs. G	C vs. ES	G	ES
b03	285.0	283.9	284.6	-0.39%	-0.2%	144.9	143.5	143.9	-0.97%	0.00%	0.0	24.0
b14	1411.3	1323.2	1345.1	-6.24%	-4.69%	1182.5	1066.8	1088.5	-9.78%	-7.94%	0.2	24.0
b20	1532.6	1285.5	1239.9	-16.12%	-19.09%	1321.4	928.9	901.8	-29.70%	-31.75%	1.0	24.0
b21	1562.5	1360.0	1368.3	-12.96%	-12.43%	1243.6	987.1	998.4	-20.63%	-19.72%	1.0	24.0

Joules, achieving a Pearson correlation coefficient (R) of 0.98, which indicates a very high degree of linear agreement between the two. **This demonstrates the high fidelity of GLOW in estimating dynamic power.**

6.4 Retiming Space Exploration

To highlight the complexity of identifying a post-synthesis power-optimal retiming solution and to demonstrate GLOW 's effectiveness in addressing this challenge, we compare its runtime and results against a time-bounded exploration of near-optimal classical

Table 4: Retiming comparison (post-synthesis results): classic retiming with register count minimization (C), and GLOW (G) for partitioned designs.

Bench. (#gates)	Register Count			Internal Dyn. Power (mW)			External Dyn. Power (mW)			Runtime (h)	
	C	G	C vs. G	C	G	C vs. G	C	G	C vs. G	C	G
Ariane (96971)	16004	16024	0.12%	41.87	40.48	-3.31%	24.3	23.0	-5.46%	0.7	7.4

Table 5: Retiming Performance Comparison Between the Original Industrial Flow and the GLOW-Integrated Flow.

Bench.	Register Count			Internal Dyn. Power (mW)			External Dyn. Power (mW)		
	I	G	I vs. G	I	G	I vs. G	I	G	I vs. G
b03	30	30	0.0%	0.270	0.272	0.59%	0.130	0.116	-10.69%
b14	215	215	0.0%	2.364	2.200	-6.96%	2.536	2.359	-6.99%
b20	428	429	0.23%	1.679	1.560	-7.10%	1.456	1.358	-6.73%
b21	428	429	0.23%	1.298	1.122	-13.56%	1.067	0.881	-17.43%

retiming solutions. We consider classical solutions whose register count is within two registers of the optimal retiming reported in Table 2, as this is the largest register difference between the solutions of GLOW and those of classical retiming for the benchmarks under exploration. We pick four benchmarks from Table 2: one small benchmark (*b03*) and three larger ones (*b14*, *b20*, and *b21*,). We run each one of these explorations on 12 cores for 24 hours, obtaining over 2000 different retiming solutions for each benchmark.

Table 3 reports our post-synthesis results, and Figure 5 visualizes the exploration results for benchmarks *b03* and *b20*. In the small benchmark *b03*, registers dominate the power; thus, as the register count increases, the power increases as well. Consequently, GLOW's solution coincides with that of classical retiming. Yet, the retiming space of the larger benchmark *b20* is more complex: power is no longer strictly proportional to register count, and solutions with the same register counts result in significant power differences. This implies that the power-optimal solution is nontrivial to identify and, as evident from the reported runtimes, performing explorations to find them is practically unfeasible (i.e., the exploration takes 24 hours even for our smallest benchmark). Even in such cases, GLOW achieves low-power designs, whose occasional deviation from the power optimum is due to the minor discrepancies between the power model and the measured power discussed in Section 6.3. Overall, **GLOW successfully navigates through such complex retiming solution spaces and consistently achieves power-efficient designs.**

6.5 Using GLOW for Processor Optimization

In this section, we evaluate GLOW's effectiveness on a 64-bit RISC-V processor Ariane [44]. From Table 2, it is evident that GLOW's runtime increases with design complexity. To make its runtime acceptable for designs of this scale (i.e., the evaluated processor has almost 97K gates), we use the same strategy as commercial retiming engines [9, 10, 38, 39]: we partition the design into parts to be retimed independently. To this end, we employ k-way partitioning with hMETIS [27] to create balanced partitions of up to 7K gates. To ensure the timing constraint for the entire circuit is satisfied, we annotate each partition's PIs and POs with delays representing its neighboring partitions. We then retime the partitions independently and sequentially, one after the other. We consistently apply both GLOW and classical retiming on the same partitions and evaluate the resulting retimed circuits.

Table 4 summarizes our post-synthesis results. At a negligible register count increase, GLOW improves both internal and external dynamic power. GLOW's runtime stays manageble thanks to our partitioning strategy; although its runtime exceeds that of classical retiming, this one-time overhead is justified by the notable power savings that can be consistently exploited throughout the processor's operational lifetime. Furthermore, the runtime could be amortized by running the partition retimings in parallel—a typical solution that we have yet to explore. In summary, our experiment shows that (1) GLOW's runtime and scalability can be managed via circuit partitioning, and (2) **GLOW achieves a reduction in total dynamic power for a production-ready processor.**

6.6 Comparison With a Commercial Flow

In this section, we integrate GLOW into a commercial logic synthesis tool to compare the effectiveness of GLOW against the commercial retiming engine (targeting the minimum number of registers, analogous to classical retiming). To use GLOW in the commercial tool, we export a circuit description, annotated with node delays, pin capacitances, and sensitivities, from the commercial synthesizer directly before retiming. We run GLOW on the exported circuit, and then inject the retimed circuit back into the commercial tool so that both approaches share the same downstream flow. As we do not have insight and control over the partitioning process in the commercial tool, we evaluate only benchmarks that are not partitioned by the tool.

Table 5 shows the post-synthesis results indicating that **GLOW is more effective in dynamic power optimization than the commercial retiming engine**; although not reported in the table, both retiming strategies meet the timing constraint. Overall, GLOW yields a power improvement of 10.44% for external dynamic power and 6.76% for internal power. **This indicates that GLOW is practically useful in an end-to-end commercial EDA flow.**

7 CONCLUSION

We presented GLOW, a retiming technique that reduces dynamic power consumption in sequential circuits while ensuring timing constraints and minimizing register count. Unlike prior methods that target only high-power nodes, GLOW leverages structural analysis to pinpoint and eliminate the root causes of glitches; it achieves near-optimal post-synthesis power solution without relying on iterative techniques limited to local optima. GLOW delivers lower power consumption with fewer registers in academic and production-ready benchmarks; it marks a significant advancement in power optimization and opens new possibilities for efficient low-power circuit design.

REFERENCES

[1] Nangate45pdk. Online: https://eda.ncsu.edu/freepdk/freepdk45/, Nov. 2011. NC State University.

[2] J. H. Anderson. *Power optimization and prediction techniques for FPGAs.* Ph.d. thesis, University of Toronto, Toronto, Canada, 2005.

[3] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, July 1999.

[4] R. Brayton and A. Mishchenko. ABC: an academic industrial-strength verification tool. In *Proceedings of the 22nd International Conference on Computer-Aided Verification*, page 24–40, Berlin, Heidelberg, July 2010.

[5] A. Buss and A. Al Rowaei. A comparison of the accuracy of discrete event and discrete time. In *Proceedings of the 43rd IEEE Winter Simulation Conference*, pages 1468–1477, Baltimore, MD, Dec. 2010.

[6] I. Bustany, G. Gasparyan, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang. An open-source constraints-driven general partitioning multi-tool for VLSI physical design. In *Proceedings of the 42nd IEEE/ACM International Conference on Computer Aided Design*, pages 1–9, San Francisco, CA, Oct. 2023.

[7] I. Bustany, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang. SpecPart: A supervised spectral framework for hypergraph partitioning solution improvement. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, San Diego, California, Dec. 2022.

[8] Cadence Design Systems. Conformal Technologies. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/conformal-technologies.html, 2025.

[9] Cadence Design Systems. Genus Synthesis Solution. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html, 2025.

[10] Cadence Design Systems. Innovus Implementaion System. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html, 2025.

[11] Cadence Design Systems. Joules RTL Power Solution. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/power-analysis/joules-rtl-power-solution.html, 2025.

[12] L. P. Carloni and A. L. Sangiovanni-Vincentelli. Combining retiming and recycling to optimize the performance of synchronous circuits. In *Proceedings of the 16th IEEE Symposium on Integrated Circuits and System Design*, pages 47–52, Sao Paolo, Brasil, Sept. 2003.

[13] N. Chabini, I. Chabini, E. M. Aboulhamid, and Y. Savaria. Unification of basic retiming and supply voltage scaling to minimize dynamic power consumption for synchronous digital designs. In *Proceedings of the 13th ACM Great Lakes Symposium on VLSI*, page 221–224, Washington, D. C., Apr. 2003.

[14] J. Cong, S. K. Lim, and C. Wu. Performance driven multi-level and multiway partitioning with retiming. In *Proceedings of the 37th ACM/IEEE Design Automation Conference*, page 274–279, Los Angeles, California, 2000.

[15] F. Corno, M. Reorda, and G. Squillero. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of Computers*, 17(3):44–53, Aug. 2000.

[16] A. Costamagna, X. Xu, G. De Micheli, and D. Ruic. Lazy man's resynthesis for glitching-aware power minimization. In *Proceedings of the 28th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 92–98, Lyon, France, May 2025.

[17] S. Dey, F. Brglez, and G. Kedem. Partitioning sequential circuits for logic optimization. In *Proceedings of the 9th IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 70–76, Cambridge, MA, Oct. 1991.

[18] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Power limitations and dark silicon challenge the future of multicore. *ACM Transaction on Computer Systems*, 30(3):1–27, Aug. 2012.

[19] R. Fischer, K. Buchenrieder, and U. Nageldinger. Reducing the power consumption of FPGAs through retiming. In *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 89–94, Greenbelt, MD, Apr. 2005.

[20] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pages 253–259, Anaheim, CA, June 1992.

[21] J. Gong, H. Li, and C. Wu. Simultaneous circuit partitioning/clustering with retiming for performance optimization. In *Proceedings of the 36th ACM/IEEE Design Automation Conference*, pages 460–465, New Orleans, LA, June 1999.

[22] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[23] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen. Design and implementation of low-area and low-power AES encryption hardware core. In *Proceedings of the 9th IEEE EUROMICRO Conference on Digital System Design*, pages 577–583, Cavtat, Croatia, Sept. 2006.

[24] Y.-L. Hsu and S.-J. Wang. Retiming-based logic synthesis for low-power. In *Proceedings of the 8th IEEE International Symposium on Low Power Electronics and Design*, pages 275–278, Monterey, CA, Aug. 2002.

[25] Z. Jeddi and E. Amini. Power optimization of sequential circuits by retiming and rewiring. In *Proceedings of the 49th IEEE International Midwest Symposium on Circuits and Systems*, pages 585–589, San Juan, PR, Aug. 2006.

[26] L. Josipović, S. Sheikhha, A. Guerrieri, P. Ienne, and J. Cortadella. Buffer placement and sizing for high-performance dataflow circuits. In *Proceedings of the 28th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 186–96, Seaside, CA, Feb. 2020.

[27] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: application in VLSI domain. In *Proceedings of the 34th ACM/IEEE Design Automation Conference*, page 526–529, Anaheim, CA, June 1997.

[28] J. Kawa. Low power and ower management for CMOS—An EDA perspective. *IEEE Transactions on Electron Devices*, 55(1):186–196, Jan. 2008.

[29] K. Lalgudi and M. Papaefthymiou. Fixed-phase retiming for low power design. In *Proceedings of 2nd IEEE International Symposium on Low Power Electronics and Design*, pages 259–264, Monterey, CA, Aug. 1996.

[30] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1-6):5–35, June 1991.

[31] J. Liu, M. Graczyk, A. Guerrieri, and L. Josipović. Fast switching activity estimation for HLS-produced dataflow circuits. In *Proceedings of the 34th IEEE International Conference on Field-Programmable Logic and Applications*, pages 118–125, Turin, Italy, Sept. 2024.

[32] J. Monteiro, S. Devadas, and A. Ghosh. Retiming sequential circuits for low power. In *Proceedings of 12th IEEE/ACM International Conference on Computer Aided Design*, pages 398–402, Santa Clara, CA, Nov. 1993.

[33] K. Muthamizh Vithagan, V. Sundaresha, and J. Viraraghavan. Geometric programming approach to glitch minimization via gate sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(6):1988–2001, June 2023.

[34] Y. Nasser, J. Lorandel, J.-C. Prévotet, and M. Hélard. RTL to transistor level power modeling and estimation techniques for FPGA and ASIC: A survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(3):479–493, Mar. 2020.

[35] M. Pedram. Power minimization in IC design: principles and applications. *ACM Transactions on Design Automation of Electronic Systems*, 1(1):3–56, Jan. 1996.

[36] C. Rizzi, A. Guerrieri, P. Ienne, and L. Josipović. A comprehensive timing model for accurate frequency tuning in dataflow circuits. In *Proceedings of the 22nd IEEE International Conference on Field-Programmable Logic and Applications*, pages 375–83, Belfast, UK, Aug. 2022.

[37] A. Sarwar. CMOS power consumption and CPD calculation. Application Report SCAA035B, Texas Instruments, June 1997. Revision B.

[38] Synopsys Inc. Fusion Compiler: RTL-to-GDSII Implementation Solution. Available: https://www.synopsys.com/implementation-and-signoff/physical-implementation/fusion-compiler.html, 2024.

[39] Synopsys Inc. Design compiler: Rtl synthesis solution. Available: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html, 2025.

[40] C. K. Teh, T. Fujita, H. Hara, and M. Hamada. A 77% energy-saving 22-transistor single-phase-clocking D-flip-flop with adaptive-coupling configuration in 40nm CMOS. In *Proceedings of the 58th IEEE International Solid-State Circuits Conference*, pages 338–340, San Francisco, CA, Feb. 2011.

[41] H. Wang, C. Rizzi, and L. Josipović. MapBuf: Simultaneous technology mapping and buffer insertion for HLS performance optimization. In *Proceedings of the 42nd IEEE/ACM Intl. Conference on Computer-Aided Design*, pages 1–9, San Francisco, CA, Oct. 2023.

[42] C. Wolf. Yosys open synthesis suite. Available: https://yosyshq.net/yosys/, 2025.

[43] J.-S. Yoon, J. Jeong, S. Lee, J. Lee, S. Lee, R.-H. Baek, and S. K. Lim. Performance, power, and area of standard cells in sub 3 nm node using buried power rail. *IEEE Transactions on Electron Devices*, 69(3):894–899, Mar. 2022.

[44] F. Zaruba and L. Benini. The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-GHz 64-bit RISC-V core in 22-nm FDSOI technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2629–2640, Nov. 2019.

[45] Z. Zhang, D. Chen, S. Dai, and K. Campbell. High-level synthesis for low-power design. *IPSJ Transactions on System LSI Design Methodology*, 8:12–25, Feb. 2015.

[46] S. Zou, J. Zhang, B. Shi, and G. Luo. PowerSyn: A logic synthesis framework with early power optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(1):203–216, Jan. 2024.