ScoutCell: Sensitivity-Guided Coupling Optimization for Timing-Driven Standard Cell Synthesis

Kairong Guo[1], Haoyi Zhang[1], Haichuan Liu[1], Chunyuan Zhao[1], Yibo Lin[1,2,3*]
[1]School of Integrated Circuits, Peking University, Beijing
[2]Institute of Electronic Design Automation, Peking University, Wuxi
[3]Beijing Advanced Innovation Center for Integrated Circuits
{krguo25, hy.zhang}@stu.pku.edu.cn, LHaiC@outlook.com, zhaochunyuan@stu.pku.edu.cn, yibolin@pku.edu.cn

Abstract

In advanced-node standard cell synthesis, timing optimization is increasingly limited by parasitic coupling effects that are not well captured by wirelength-centric heuristics alone. While recent timing-driven approaches improve delay by shortening critical paths, they often overlook the disproportionate impact of parasitic coupling on specific delay-critical net pairs. In this work, we propose ScoutCell, a sensitivity-guided standard-cell synthesis framework that follows a characterize-then-optimize flow. ScoutCell first extracts a reusable pairwise sensitivity prior from an anchor layout, and then uses this prior to guide coupling-aware placement candidate screening and routing refinement toward isolating delay-critical net-pair interactions. Experiments on the ASAP7 predictive process design kit show that ScoutCell consistently outperforms both the area/wirelength-driven baseline and a state-of-the-art timing-driven cell synthesis method, reducing the average cell delay by 0.6%/1.2% and 0.2%/0.5% on combinational/sequential cells, respectively. Under a full logic-synthesis and physical-implementation flow, ScoutCell further improves the average post-route achievable frequency by 8.5% and 4.7% over the two baselines, demonstrating that its cell-level timing benefits remain visible at the design level.

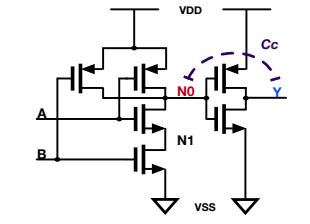
Keywords

Standard Cell Layout, Transistor Placement & Routing, Timing-Driven Layout Synthesis

ACM Reference Format:
Kairong Guo[1], Haoyi Zhang[1], Haichuan Liu[1], Chunyuan Zhao[1], Yibo Lin[1,2,3*] . 2026. ScoutCell: Sensitivity-Guided Coupling Optimization for Timing-Driven Standard Cell Synthesis . In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '26), November 08–12, 2026, San Jose, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3831252.3834054

1 Introduction

Standard cells are the fundamental building blocks of digital integrated circuits. As technology scales to 7nm and beyond, the exponential increase in design rules and layout complexity makes manual cell crafting prohibitively time-consuming and sub-optimal.



Metrics	*Layout 1*		*Layout 2*	
	tot.	Y&N0	tot.	Y&N0
WL (nm)	414	342	441 (+6.5%)	369 (+7.9%)
Cc (fF)	0.266	0.133	0.273 (+2.6%)	**0.129 (-3%)**
Delay (ps)	111.1		**110.5 (-0.54%)**	

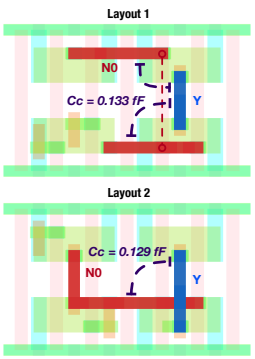


Figure 1: Comparison of two layout variants for AND2. *Layout 1* follows a conventional wirelength-centric optimization with smaller total coupling capacitance. In contrast, *Layout 2* achieves a better overall timing result (-0.54%) by sacrificing a small amount of wirelength (+6.5%) to specifically isolate the delay-dominant net pair (Y & N0).

Consequently, automated standard cell synthesis has emerged as a pivotal research frontier to accelerate design turnaround and enable aggressive Design Technology Co-Optimization (DTCO).

Traditionally, the primary objectives of standard cell synthesis frameworks have been minimizing cell area and maximizing routability. Extensive research has explored transistor placement and in-cell routing using Integer Linear Programming (ILP)[19][2], Satisfiability Modulo Theories (SMT)[13][9][1][6], heuristic algorithms and reinforcement learning[7][14] to ensure Design Rule Check (DRC) clean layouts within the smallest possible area. Other works have focused heavily on improving pin accessibility to ease block-level routing[8]. These methods typically use total wirelength (TWL) as a proxy for both routability and performance. However, this assumption breaks down in advanced nodes. To mitigate escalating wire resistance, foundries have significantly increased the aspect ratio of lower metal layers (e.g., M0, M1). This structural shift has caused sidewall coupling capacitance (C_c) to emerge as a dominant parasitic bottleneck for cell delay.

Driven by the need for performance optimization, recent research has pivoted towards timing-driven cell synthesis. Several studies[10][11][17][18] have leveraged machine-learning surrogates to evaluate standard cell layout performance and capture coupling effects. However, while these approaches excel at layout performance prediction, their complex models cannot easily be inverted into actionable optimization directives to actively guide the layout synthesis process. On the synthesis front, MinDCell[3] introduced a critical-path-driven framework that minimizes the topological span and wirelength of critical nets. While effectively

*Corresponding author.

ICCAD '26, San Jose, CA, USA

ACM ISBN 979-8-4007-2873-0/2026/11
https://doi.org/10.1145/3831252.3834054

reducing wire resistance, this purely wirelength-centric heuristic remains largely oblivious to parasitic coupling. As illustrated in Fig. 1, the coupling capacitance between specific net pairs can severely degrade cell delay. For instance, the interaction between nets Y and N0 in the AND2 cell acts as a strong Miller capacitance across the second-stage inverter, heavily penalizing the transition speed of the primary timing arc. This disparity reveals a critical optimization gap: to unlock the ultimate performance limits of advanced nodes, standard cell synthesis must move beyond simple wirelength minimization and proactively isolate these delay-dominant net-pair interactions.

To unlock the maximum performance potential within the tightly constrained routing resources of advanced cells, we must rethink the parasitic trade-offs. The delay of standard cells in advanced nodes exhibits significant skewness: it is overwhelmingly sensitive to the C_c of a select few critical nets, while variations in other parasitics yield marginal impacts. This insight unlocks a new optimization space. Rather than singularly minimizing topological span, we can proactively secure spatial isolation for highly sensitive nets to eliminate the dominant C_c penalty, thereby unlocking the true performance potential of the cell. Accordingly, we propose `ScoutCell`, a sensitivity-guided framework that optimizes the spatial isolation between critical nets to suppress coupling capacitance and enhance the timing performance of standard cells.

The contributions of this work are summarized as follows:

- We propose `ScoutCell`, a sensitivity-guided standard-cell synthesis framework that brings net-pair delay sensitivity into timing-driven cell optimization through a characterize-then-optimize flow.
- We develop a coupling-aware placement candidate exploration and screening flow that enumerates legal placement solutions and uses reusable net-pair sensitivity priors to favor topologies that better isolate delay-critical interactions.
- We formulate a coupling-aware routing optimization scheme that imposes isolation penalties on harmful net-pair interactions, thereby suppressing sidewall coupling on delay-dominant nets without relying on wirelength minimization alone.
- Experiments on ASAP7[4] show that `ScoutCell` consistently outperforms both the area/wirelength-driven baseline and the state-of-the-art[3] timing-driven cell synthesis method, reducing the average cell delay by 0.6%/1.2% and 0.2%/0.5% on combinational/sequential cells, respectively, over the two baselines, and improving the average post-route achievable frequency by 8.5% and 4.7% under a full logic-synthesis and physical-implementation flow evaluation.

The rest of the paper is organized as follows: Section 2 provides background on advanced-node parasitics and presents the key observations motivating our approach; Section 3 details the proposed method; Section 4 validates the method with experimental results; Section 5 concludes the paper.

2 Preliminaries

2.1 Parasitics and Delay under Advanced Nodes

In sub-7nm standard cell synthesis, the extreme scaling of intra-cell routing tracks changes the balance of parasitic effects. To keep local interconnect resistance acceptable, lower metal layers such as M0/M1 are typically implemented with a high aspect ratio ($AR = T/W > 2$)[4], making sidewall coupling increasingly prominent in the dense intra-cell routing environment.

For an internal net n, the wire resistance R_n and ground capacitance $C_{g,n}$ scale roughly with its wirelength L_n. In contrast, the coupling capacitance between two nets n and m depends on their relative geometry:

$$C_c(n,m) \propto \frac{T \cdot L_{\parallel}(n,m)}{\delta(n,m)}, \tag{1}$$

where $L_{\parallel}(n,m)$ is the parallel run-length and $\delta(n,m)$ is the spacing between the two nets. Accordingly, the delay contribution of a timing-critical stage driven by n can be approximated as

$$d_n \propto R_n \cdot \left(C_{g,n} + \sum_{m \neq n} \Delta_{n,m} \cdot C_c(n,m) \right), \tag{2}$$

where $\Delta_{n,m}$ is an effective Miller factor determined by the relative switching between the two nets.

Across legal layout variants of the same cell, wirelength-driven terms often vary within a limited range once the layout remains compact and routable. By contrast, local track assignment and adjacency can substantially change $L_{\parallel}(n,m)$ and $\delta(n,m)$ for a few nearby nets. Therefore, in the advanced-node standard-cell regime considered in this work, timing differences between candidate layouts are often governed more by the *distribution* of coupling capacitance than by small changes in wirelength alone.

2.2 Observations from Pairwise Coupling Sensitivity Analysis

The above analysis suggests that timing depends not only on the total amount of coupling capacitance, but also on how that coupling is distributed across different net pairs. To quantify the timing impact of this distribution, we define a perturbation-based *pairwise coupling sensitivity* under a common `typ` characterization condition. For a net pair (n,m), we uniformly reduce the extracted coupling capacitances between the two nets by 10% while keeping all other parasitics fixed, and then re-evaluate the corresponding cell delay using SPICE. The pairwise sensitivity is defined as the resulting delay reduction:

$$S(n,m) = \tau_{\text{base}} - \tau_{-10\%}(n,m), \tag{3}$$

where τ_{base} is the baseline delay and $\tau_{-10\%}(n,m)$ is the delay after reducing the coupling on pair (n,m) by 10%. Therefore, $S(n,m)$ is measured in ps, and a larger positive value indicates that the pair is more delay-critical. Using this perturbation-based pairwise sensitivity analysis, we observe two empirical regularities.

Observation 1 (Sparse and pair-structured sensitivity). Under this sensitivity probe, delay impact is highly concentrated on a small subset of net-pair interactions. Fig. 2 shows representative `OR5x1` and `DFFHQNx1` cases, where only a few pairwise hotspots dominate the positive delay response while most pairs remain weakly sensitive. This trend also persists across the full ASAP7 standard-cell library under the same characterization setup: the top-3 sensitive nets capture 76.4% of the total positive coupling sensitivity on average. As a result, cell delay is governed less by the total amount of coupling capacitance than by which nets, and in particular which net pairs, carry that coupling. This is consistent

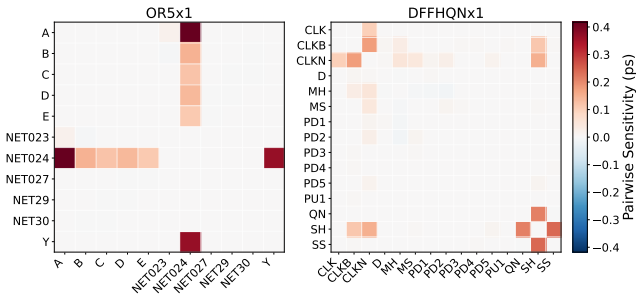


Figure 2: Representative pairwise sensitivity maps for `OR5x1` and `DFFHQNx1`, where each entry reports the delay improvement obtained by reducing the corresponding pair coupling by 10%. Across both combinational and sequential cells, sensitivity is highly sparse: most net pairs remain weakly sensitive, while only a small subset of pairwise interactions dominates the positive delay response.

with the motivating `AND2` example in Fig. 1, where the better-timing layout is obtained not by minimizing total coupling or wirelength, but by isolating the delay-dominant pair `Y<->N0`. ***Observation 1 therefore motivates explicitly accounting for delay-critical net-pair interactions during timing-driven cell optimization.***

Observation 2 (Reference-layout sensitivity is reusable). A practical question is whether a pairwise sensitivity profile extracted from one RC netlist remains informative after the layout changes. To evaluate this more comprehensively, we analyze over 60 ASAP7 cells spanning both combinational and sequential types, and generate multiple legal placement-and-routing variants for each cell, yielding over 1.8K parasitic-extracted layouts in total. Across within-cell layout variants, pairwise sensitivity rankings remain highly correlated. For each cell, we compute the average rank correlation across all $n(n-1)/2$ variant pairs. Averaging these per-cell metrics yields a mean Spearman rank correlation of 0.898. More importantly, when one variant of a cell is used to explain another, the reference top-10 sensitive pairs still retain strong explanatory power. For each cell, we evaluate the top-10 capture ratio over all ordered reference-target pairs and average over these $n(n-1)$ samples; averaging this per-cell capture score across cells gives 98.06% (25th/75th percentiles: 97.76%/99.95%) and 82.61% (73.09%/97.31%) for combinational and sequential cells, respectively, where only positive sensitivities are counted because downstream optimization retains only beneficial pairs. Therefore, although absolute sensitivities vary with detailed placement and routing, the dominant-pair structure extracted from a single reference layout remains sufficiently stable to serve as a reusable prior across legal variants of the same cell. ***Observation 2 thus suggests extracting a reusable sensitivity prior from a single reference layout and propagating it to later optimization stages.***

Taken together, these observations suggest that advanced-node timing optimization should not treat all parasitics uniformly. Instead, it should explicitly target a small set of dominant coupling pairs, leveraging a reusable sensitivity prior extracted from a reference layout.

2.3 Problem Formulation

Timing-Driven Standard Cell Synthesis: Given a transistor-level netlist and physical design rules, the objective is to determine the placement of transistors and routing topologies of nets on a 3D grid. The output layout must be LVS-correct and DRC-clean. Subject to these constraints, we define the optimization target as a unified cell-level timing metric, denoted as D_{cell}. To ensure consistency with industry-standard static timing analysis, all delay components are characterized via post-layout SPICE simulations with extracted parasitics and evaluated at a typical input slew and output load. Depending on the logic functionality, D_{cell} is formulated as follows:

- For **combinational cells**, D_{cell} is defined as the maximum propagation delay among all valid input-to-output timing arcs under the worst-case input pattern:

$$D_{cell} = \max_{\pi \in \Pi} d(\pi) \tag{4}$$

  where Π represents the set of all timing paths, and $d(\pi)$ is the path delay.
- For **sequential cells**, the formulation adapts to the specific memory element to capture its true data-path latency:
  - **Flip-Flops:** The delay is calculated as the weighted sum of the latencies for data-rise and data-fall transitions. Each transition latency comprises its respective setup time and clock-to-Q delay:

$$D_{cell} = w_r \cdot \left(t^{rise}_{setup} + t^{rise}_{c2q}\right) + w_f \cdot \left(t^{fall}_{setup} + t^{fall}_{c2q}\right) \tag{5}$$

    where w_r and w_f are user-defined weights (typically $w_r = w_f = 0.5$).
  - **Latches:** Unlike edge-triggered flip-flops, latches are level-sensitive and enable time borrowing. Consequently, the critical performance bottleneck during the transparent phase is the data-to-Q propagation delay. To maintain evaluation consistency, it is similarly defined as the weighted sum of the rise and fall transitions:

$$D_{cell} = w_r \cdot t^{rise}_{d2q} + w_f \cdot t^{fall}_{d2q} \tag{6}$$

This unified cell-level timing metric is later used as the paper's primary cell-quality target.

3 Methodology

3.1 Framework Overview

`ScoutCell` mitigates advanced-node parasitic coupling in standard-cell synthesis through a "characterize-then-optimize" flow. Instead of attempting full physical timing optimization over the entire search space, it extracts a reusable pairwise sensitivity prior from one anchor layout and reuses that prior in both placement selection and routing optimization. As illustrated in Fig. 3, the framework has three coupled stages: placement exploration, pairwise sensitivity extraction, and routing optimization.

3.1.1 ***Placement exploration.*** Given the input transistor netlist and physical design rules, `ScoutCell` first enumerates a library of legal placement candidates with a SAT/AllSAT engine. A lightweight proxy router then evaluates each candidate without invoking exact in-cell routing. This stage outputs both the candidate library and the coarse geometric features used for candidate re-ranking. After extracting the pairwise sensitivity prior $\widehat{W}(n,m)$ from the anchor layout, `ScoutCell` combines that prior with the proxy features to select the placement for final in-cell routing.

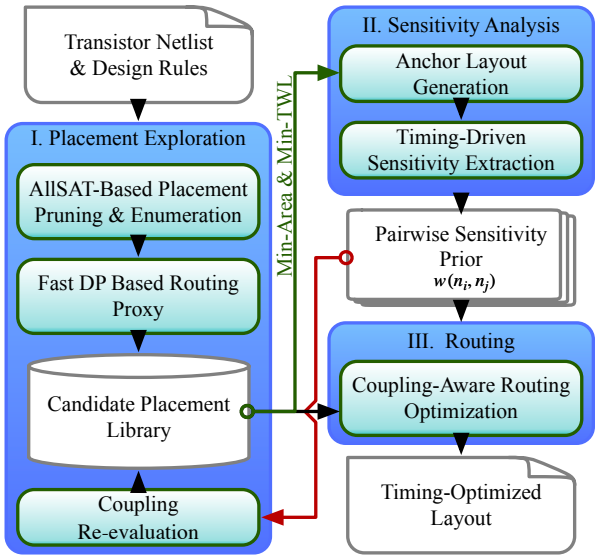


Figure 3: Framework overview of `ScoutCell`. A reusable pairwise sensitivity prior is extracted from one anchor layout and then injected into placement candidate re-ranking and routing optimization.

3.1.2 ***Pairwise sensitivity analysis.*** Running PEX+SPICE on every legal variant is prohibitively expensive. We therefore select the minimum-area and minimum-TWL placement candidate from the candidate library, apply wirelength-driven routing to generate its layout as the anchor layout, and run extracted-netlist perturbation analysis only on that layout. This stage outputs the pairwise sensitivity prior $\widehat{W}(n, m)$, which quantifies the delay benefit of reducing coupling on each net pair (n, m) and serves as the reusable timing prior for the later optimization stages.

3.1.3 ***Routing optimization.*** For the selected placement, the in-cell router first optimizes a wirelength-driven objective to obtain an initial routing solution and grid occupancy. It then runs a coupling-aware iterative rip-up and reroute (IRR) procedure under an explicit pairwise penalty objective guided by $\widehat{W}(n, m)$. To keep this optimization tractable, `ScoutCell` linearizes the quadratic pairwise penalty with precomputed cost lookup tables.

For clarity, the remainder of this section first presents the pairwise sensitivity analysis, and then describes how the extracted prior is injected into placement candidate selection and routing optimization.

3.2 Anchor-Based Pairwise Sensitivity Extraction

To guide placement re-ranking and routing optimization, `ScoutCell` must identify which intra-cell net pairs can return the largest timing improvement when their coupling is reduced. Given the severely constrained routing resources in standard cell synthesis, our objective is to maximize realizable delay recovery via discrete local perturbations, rather than computing infinitesimal parasitic gradients. We therefore use a simulation-based finite-perturbation procedure to quantify the pairwise delay sensitivity of each candidate pair.

Starting from this wirelength-driven anchor layout, we run PEX to obtain the baseline extracted parasitic netlist. We then identify the timing arc or latency term that determines the current D_{cell} value. To reduce the number of SPICE simulations, we first perturb the total extracted coupling capacitance associated with each net individually to screen timing-critical nets, and then perform pairwise perturbations only within this screened subset.

For each candidate pair (n, m), we uniformly reduce the extracted coupling capacitances between the two nets by a fixed ratio $\eta = 10\%$, keep all other parasitics unchanged, and re-evaluate D_{cell} in SPICE. We define the raw pairwise sensitivity score as the resulting timing improvement:

$$S_{\text{raw}}(n, m) = D_{\text{base}} - D_{-\eta}(n, m), \quad \eta = 10\% \tag{7}$$

where D_{base} is the baseline timing metric on the anchor layout and $D_{-\eta}(n, m)$ is the timing after reducing the coupling on pair (n, m) by η. Unlike a gradient-style quantity that divides by the perturbation magnitude, $S_{\text{raw}}(n, m)$ directly measures the delay recovery available from a realizable local perturbation.

Although pairwise perturbation is nominally $O(N^2)$, the characterization remains tractable because we only evaluate pairs formed within the screened timing-critical subset.

For downstream optimization, we discard non-positive scores and quantize the remaining ones into solver-friendly integer weights:

$$\widehat{W}(n, m) = \mathcal{Q}(\max(0, S_{\text{raw}}(n, m))) \tag{8}$$

where $\mathcal{Q}(\cdot)$ is a monotone bucketing function that maps raw delay improvements to a small set of integer priority levels. Larger $\widehat{W}(n, m)$ indicates that stronger spatial isolation should be enforced for pair (n, m). Because the perturbation is applied to the unordered pair as a whole, $\widehat{W}(n, m) = \widehat{W}(m, n)$ by construction. This integer-bucketed prior is the object injected into both placement candidate re-ranking and routing optimization.

3.3 Placement Candidate Exploration and Re-ranking

To generate routable layouts while suppressing coupling, we combine constraint-driven placement enumeration with a lightweight proxy evaluation engine. Unlike conventional flows that postpone routing information until after placement, `ScoutCell` materializes coarse routing topologies early and uses them to re-rank legal candidates with the pairwise prior.

3.3.1 Constraint-Driven Placement Enumeration. We enumerate placement candidates using an AllSAT formulation that combines legality, duplicate-solution pruning, congestion-aware routability screening, and channel connected components(CCC) based complementary pairing[15]. Let $T^m_{i,c,f}$ denote whether transistor i in row $m \in \{N, P\}$ is assigned to column c with flip state f.

$$\sum_{c,f} T^m_{i,c,f} = 1, \ \forall i, m, \qquad \sum_{i,f} T^m_{i,c,f} = 1, \ \forall c, m \tag{9}$$

The first constraint enforces one placement per transistor, while the second guarantees one occupant per row-column slot. We enforce diffusion-sharing legality by forbidding adjacent assignments

whose touching diffusion nets are incompatible:

$$T^m_{i,c,f} + T^m_{j,c+1,f'} \le 1, \quad \text{if } right(i,f) \nsim left(j,f'). \tag{10}$$

We further prune duplicate placements by disabling flips for source-drain-symmetric devices and enforcing a fixed order on identical transistors. To suppress placements that are legal but unlikely to route, we also encode compact routability constraints including metal-grid occupancy, access feasibility, detailed crossing, track crossing, and line-end checks. For complex and sequential cells, we use CCC labels to disallow same-column complementary pairings across unrelated channel-connected components.

To further remove mirrored duplicates, we apply a lightweight **canonical-device symmetry pruning** rule. For a placement with a total width of P columns, one deterministically selected canonical device $(i^\star, m^\star)$ is restricted to the left half of the grid:

$$\sum_{c>\lfloor (P-1)/2 \rfloor,\, f} T^{m^\star}_{i^\star,c,f} = 0. \tag{11}$$

Here $(i^\star, m^\star)$ is chosen using a fixed transistor-signature ordering so that symmetric solutions collapse to the same representative.

3.3.2 Fast Proxy Routing and Prior-Guided Geometry Evaluation. To compare a large number of placement candidates without invoking exact in-cell routing, ScoutCell builds a coarse but pessimistic projected-track topology for each candidate and uses it to estimate how well the candidate separates timing-sensitive net pairs. For a placement candidate π, each internal net n first induces a horizontal span $[l_n, r_n]$ from its device terminals. Let $\mathcal{T}$ denote the projected track set. The proxy router then assigns one routing track in $\mathcal{T}$ to each column in this span and materializes an explicit routed topology

$$\mathcal{R}_\pi(n) = (\mathcal{H}_\pi(n), \mathcal{V}_\pi(n)), \tag{12}$$

where $\mathcal{H}_\pi(n)$ is the set of horizontal segments and $\mathcal{V}_\pi(n)$ is the set of vertical segments.

To construct these topologies, the proxy router processes nets sequentially using a grid-based Dynamic Programming (DP) algorithm. However, a naive, single sequential legalization pass would be strongly order-sensitive because early nets could monopolize the cheapest tracks. ScoutCell therefore uses a mandatory two-pass negotiated DP router to mitigate this bias.

In the first warm-up pass, all nets are routed independently without conflict resolution to reveal provisional demand hotspots, which are recorded as a history penalty field $H_\pi(x,t)$. In the subsequent legalization pass, nets are processed in nonincreasing span width, and the DP incorporates both strict legality constraints and congestion awareness. Specifically, any state (x,t) that physically conflicts with an already legalized net is strictly blocked ($D_n(x,t) = \infty$). For all legally available states, each net minimizes the cumulative cost $D_n(x,t)$ at column x and track t:

$$\begin{aligned} D_n(x,t) = {} & c^n_{\text{acc}}(x,t) + \lambda_{\text{hist}} H_\pi(x,t) \\ & + \min_{t' \in \mathcal{T}} \Big(D_n(x-1,t') + \lambda_{\text{switch}} \mathbf{1}[t' \ne t] \Big), \end{aligned} \tag{13}$$

where $c^n_{\text{acc}}(x,t)$ is the terminal-access cost of connecting the terminals to track t, and $\mathbf{1}[t' \ne t]$ penalizes track switching. In this formulation, the hard infinite-cost constraints guarantee an overlap-free topology, while the soft history penalty $\lambda_{\text{hist}} H_\pi(x,t)$ proactively steers the routing away from systematically over-demanded tracks.

Upon completing the DP traversal, the proxy router backtraces the optimal path from the minimum-cost endpoint to instantiate the explicit horizontal and vertical routing segments, $\mathcal{H}_\pi(n)$ and $\mathcal{V}_\pi(n)$. The track-aware proxy wirelength, $\text{TWL}_{\text{proxy}}(\pi)$, is then defined strictly by summing the physical horizontal spans and the exact lengths of these materialized vertical segments across all nets.

After processing all nets, the proxy extracts pairwise geometry from the routed topology. Based on these proxy-routed segments, we define the geometry surrogate for each net pair (n,m) as

$$G_\pi(n,m) = G^{\text{hor}}_\pi(n,m) + G^{\text{ver}}_\pi(n,m), \tag{14}$$

$$\begin{aligned} G^{\text{hor}}_\pi(n,m) = \sum_{h \in \mathcal{H}_\pi(n)} \sum_{h' \in \mathcal{H}_\pi(m)} & \mathbf{1}[1 \le |t(h) - t(h')| \le W_t] \\ & \times \rho(|t(h) - t(h')|)\ \text{ov}_x(h,h'), \end{aligned} \tag{15}$$

$$\begin{aligned} G^{\text{ver}}_\pi(n,m) = \sum_{v \in \mathcal{V}_\pi(n)} \sum_{v' \in \mathcal{V}_\pi(m)} & \mathbf{1}[1 \le |x(v) - x(v')| \le W_x] \\ & \times \rho(|x(v) - x(v')|)\ \text{ov}_y(v,v'), \end{aligned} \tag{16}$$

where $\Delta t(h,h') = |t(h) - t(h')|$ is the track distance between two horizontal segments, $\Delta x(v,v') = |x(v) - x(v')|$ is the column distance between two vertical segments, ov_x and ov_y denote projected overlap lengths, and $\rho(\cdot)$ is distance-decay factor within local windows W_t and W_x.

The sensitivity of a net pair is injected only through the anchor-derived pairwise prior $\widehat{W}(n,m)$; the proxy router contributes only the candidate-specific geometry term $G_\pi(n,m)$. We therefore define the candidate-level proxy coupling score as

$$C_{\text{proxy}}(\pi) = \sum_{n<m} \widehat{W}(n,m)\, G_\pi(n,m), \tag{17}$$

and exhaustively re-rank placement candidates by

$$F(\pi) = \text{TWL}_{\text{proxy}}(\pi) + \lambda_{\text{proxy}} \cdot C_{\text{proxy}}(\pi). \tag{18}$$

ScoutCell therefore exhaustively re-ranks the legal placement library under $F(\pi)$ before invoking the final in-cell router. The selected placement is then handed to the final routing stage, where the same pairwise prior $\widehat{W}(n,m)$ is reused under an edge-level coupling objective instead of the proxy geometry score.

3.4 Coupling-Aware Routing Optimization

After placement selection, ScoutCell moves from the proxy geometry score used during candidate re-ranking to an edge-level coupling objective defined on the in-cell routing graph. The in-cell router first computes a wirelength-driven solution to establish legal connectivity and grid occupancy. Starting from this initialization, it performs a sensitivity-guided Iterative Rip-up and Reroute (IRR) procedure, where each iteration reroutes each net at a time under a linearized coupling-aware objective.

3.4.1 Formulation of the Global Coupling Penalty. Let $\mathcal{N}$ be the set of nets and E be the set of routing edges in the in-cell routing graph. We use a pseudo-Boolean decision variable $x_{e,n} \in \{0,1\}$ to indicate whether edge e is occupied by net n. To bound the constraint size, we evaluate coupling only within a local neighborhood. Two edges e_i and e_j form an adjacent parallel pair $(e_i, e_j) \in \mathcal{A}$ only if they lie

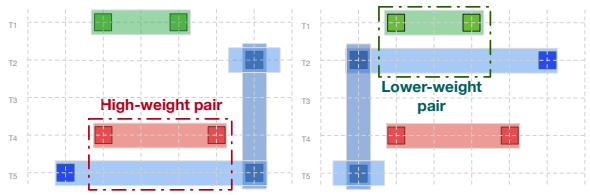


Figure 4: Sensitivity-guided routing optimization. Left: the wirelength-driven initialization creates a high-weight coupling interaction between the blue and red nets. Right: reroutes the blue net to reduce the weighted coupling penalty at the cost of a small detour.

Algorithm 1: Sensitivity-Guided Iterative Routing

Input: Netlist $\mathcal{N}$, routing graph $G = (V, E)$, max track gap K, weights $\widehat{W}$, objective coefficients α, β

$\mathcal{A} \leftarrow$ ExtractAdjacentPairs(E, K);
$\mathbf{M} \leftarrow$ InitialWirelengthDrivenRouting$(G, \mathcal{N})$;
for *iter* $\leftarrow 1$ **to** MAX_ITER **do**
  foreach *net* $n^* \in \mathcal{N}$ **do**
    $\mathbf{M}_{fixed} \leftarrow \mathbf{M} \setminus \{\text{routing of } n^*\}$;
    $\mathcal{T}^{n^*}_{cost} \leftarrow \mathbf{0}^{|E|}$;
    foreach *candidate edge* $e_i \in E$ **do**
      $c_i \leftarrow 0$;
      foreach *valid neighbor* $e_j \in \mathcal{A}(e_i)$ **do**
        if e_j *is occupied by net* m *in* $\mathbf{M}_{fixed}$ **then**
          $c_i \leftarrow c_i + \gamma_{i,j} \cdot \widehat{W}(n^*, m)$;
      $\mathcal{T}^{n^*}_{cost}[e_i] \leftarrow c_i$;
    Set objective to Minimize $\sum_{e_i \in E}(\alpha + \beta \cdot \mathcal{T}^{n^*}_{cost}[e_i]) \cdot x_{e_i,n^*}$;
    $\mathbf{M}_{n^*} \leftarrow$ RoutingSolver.Optimize();
    $\mathbf{M} \leftarrow \mathbf{M}_{fixed} \cup \mathbf{M}_{n^*}$;
return M

on the same metal layer, and their track gap is within a user-defined threshold. We assign a geometric coupling coefficient $\gamma_{i,j}$ to each such pair based on overlap length and distance decay.

We then define the global coupling penalty $\Phi_{C_c}(\mathbf{X})$ as the weighted sum of all valid physical couplings. Guided by the pairwise sensitivity prior $\widehat{W}(n, m)$ extracted from the anchor layout, the objective penalizes a pair only when two adjacent parallel edges are occupied by two different nets:

$$\Phi_{C_c}(\mathbf{X}) = \sum_{(e_i,e_j)\in\mathcal{A}} \gamma_{i,j} \sum_{\substack{n,m\in\mathcal{N} \\ n\neq m}} \widehat{W}(n, m)\, x_{e_i,n} x_{e_j,m} \tag{19}$$

This objective is motivated by the physical intuition that dominant coupling capacitance arises from adjacent parallel metal segments. Conceptually, this objective serves as the edge-level counterpart to the placement-stage proxy score $C_{\text{proxy}}(\pi)$: it retains the same pairwise prior $\widehat{W}(n, m)$, while replacing the proxy geometry surrogate with exact routed edge interactions. However, directly optimizing Eq. (19) is computationally intractable for SMT solvers, as it inherently introduces $\mathcal{O}(|E|^2 \cdot |\mathcal{N}|^2)$ quadratic pseudo-Boolean cross-terms.

3.4.2 Linearization via Iterative Rip-up and Reroute. To avoid this state-space explosion introduced by quadratic cross-terms, as shown in Fig. 4, ScoutCell applies IRR on top of the wirelength-driven initial solution. In each iteration, every net is visited once as the active net n^*, while all remaining nets are treated as fixed background routing. Fixing the background routing converts the quadratic interactions involving n^* into a linear per-edge penalty, which we precompute as a lookup table before invoking the solver. Let $\mathcal{A}(e_i)$ denote the subset of valid neighbors for a specific edge e_i:

$$\mathcal{T}^{n^*}_{cost}[e_i] = \sum_{e_j\in\mathcal{A}(e_i)} \gamma_{i,j} \cdot \sum_{m\neq n^*} \overline{x}_{e_j,m} \cdot \widehat{W}(n^*, m) \tag{20}$$

This transforms the active net's coupling penalty $\Phi^{(n^*)}_{C_c}$ into a dot product between its decision variables x_{e_i,n^*} and the precomputed static costs:

$$\Phi^{(n^*)}_{C_c}(\mathbf{x}_{n^*}) = \sum_{e_i\in E} x_{e_i,n^*} \cdot \mathcal{T}^{n^*}_{cost}[e_i] \tag{21}$$

The rerouting of n^* then solves a pseudo-Boolean objective that combines a base routing wirelength cost, weighted by α, and the coupling penalty, weighted by β:

$$\begin{aligned} \text{Minimize} \quad & \sum_{e_i\in E} \left(\alpha + \beta \cdot \mathcal{T}^{n^*}_{cost}[e_i]\right) \cdot x_{e_i,n^*} \\ \text{Subject to} \quad & \textit{Connectivity constraints}, \\ & \textit{Design rule constraints}. \end{aligned} \tag{22}$$

After optimizing n^*, we update the routing solution and move to the next active net. The complete algorithm process is shown in Algorithm 1. This iterative scheme eliminates the intractable quadratic cross-terms and relieves the constraint-solving burden on SMT solver, allowing ScoutCell to fully respect the sensitivity prior $\widehat{W}(n, m)$ without triggering a state-space explosion.

4 Experimental Results

The proposed ScoutCell framework is implemented in C++ and evaluated on an AMD EPYC 7543 workstation, utilizing the Z3 [5] solver (v4.8.5) for ALL-SAT and pseudo-boolean optimization tasks. All experiments are conducted on a modified ASAP7 [4] predictive PDK. We employ Calibre for parasitic extraction (PEX), Cadence Spectre for SPICE-level sensitivity perturbation simulations, and Cadence Liberate to generate accurate Liberty (.lib) files for cell-level characterization. For block-level full-flow evaluation, logic synthesis and physical implementation are performed using Cadence Genus and Innovus, respectively, on a set of open-source hardware designs sourced from OpenCores [12]. The following experiments include both cell-level timing results and block-design-level full-flow results.

4.1 Cell-Level Timing Improvement

We first compare three libraries under the same PDK and characterization flow: (1) *Min-Area & Min-WL*, which is generated by applying wirelength-driven routing to the minimum-area and minimum-wirelength placement candidate (serving as the anchor layout) for each cell; (2) *MinDCell*, reproduced following the minimum-delay synthesis methodology of [3] on top of AutoCellGen [16]; and (3) ScoutCell, generated by the proposed flow. Tables 1 and 2

Table 1: Combinational-cell results. Sensitive C_c is the summed coupling capacitance of the *top-2* sensitive net pairs in fF; D_{cell} and power are reported in ps and μW, respectively. ScoutCell achieves the best average D_{cell}.

Cell Type	#C	Min-Area & Min-WL					MinDCell [3]					ScoutCell				
		CPP	WL	Sensitive C_c	Power	D_{cell}	CPP	WL	Sensitive C_c	Power	D_{cell}	CPP	WL	Sensitive C_c	Power	D_{cell}
AND/OR	5	7.20	109.350	0.160	0.376	112.638	8.20	101.750	0.158	0.372	112.207	8.20	109.650	0.156	0.362	**111.107**
AO/OA	15	9.07	148.850	0.147	0.440	112.363	9.07	141.950	0.141	0.436	111.952	9.07	149.900	0.140	0.431	**111.550**
AOI/OAI	9	7.22	138.333	0.074	0.329	91.009	7.44	120.778	0.074	0.321	**90.506**	7.33	134.694	0.074	0.322	90.510
BUF/INV	5	5.00	62.850	0.201	0.335	95.684	5.60	62.150	0.199	0.333	95.582	5.60	62.700	0.196	0.329	**95.389**
FA	1	14.00	311.000	0.115	36.505	240.470	14.00	414.250	0.098	36.568	239.826	14.00	336.250	0.094	36.647	**239.575**
NAND/NOR	8	8.13	140.563	0.174	0.395	101.866	8.13	129.250	0.168	0.389	101.547	8.13	136.156	0.166	0.389	**101.508**
XOR/XNOR	5	9.20	148.200	0.124	0.366	107.210	9.20	134.400	0.122	0.364	**106.624**	9.20	140.300	0.119	0.360	106.771
Avg./tot.	48	8.54	151.307	0.142	5.535	123.034	8.81	157.790	0.137	5.540	122.606	8.79	152.807	**0.136**	5.549	**122.344**
Norm.	-	1.000	1.000	1.000	1.000	1.000	1.030	1.043	0.965	1.001	0.997	1.029	1.010	**0.950**	1.002	**0.994**

Table 2: Sequential-cell results. Sensitive C_c is the summed coupling capacitance of the *top-5* sensitive net pairs in fF; D_{cell} and power are reported in ps and μW, respectively. ScoutCell achieves the best average D_{cell}.

Cell	Min-Area & Min-WL					MinDCell [3]					ScoutCell				
	CPP	WL	Sensitive C_c	Power	D_{cell}	CPP	WL	Sensitive C_c	Power	D_{cell}	CPP	WL	Sensitive C_c	Power	D_{cell}
DFFHQNx1	18	446.25	0.295	0.735	92.231	18	377.25	0.284	0.722	92.194	18	475.00	0.288	0.723	**91.723**
DFFHQNx2	19	504.25	0.506	0.835	97.549	19	467.25	0.519	0.818	97.403	19	500.50	0.513	0.830	**97.295**
DFFHQNx3	20	676.03	0.526	0.940	93.538	21	553.50	0.545	0.920	**92.187**	21	629.50	0.487	0.898	92.489
DFFHQx4	25	671.00	0.500	1.112	90.049	25	647.98	0.453	1.117	89.232	25	673.25	0.510	1.126	**88.926**
DFFLQNx1	18	506.50	0.400	0.736	93.735	18	647.50	0.443	0.771	93.170	18	494.25	0.385	0.724	**93.153**
DFFLQNx2	19	518.25	0.317	0.854	100.533	19	635.25	0.300	0.879	99.199	19	462.00	0.280	0.828	**98.034**
DFFLQNx3	20	689.65	0.471	0.890	97.911	20	761.00	0.550	0.936	95.903	20	666.00	0.481	0.872	**95.563**
DFFLQx4	25	819.59	0.407	1.205	110.988	25	936.09	0.417	1.180	110.224	25	901.00	0.401	1.204	**107.875**
DHLx1	13	284.65	0.195	0.652	70.266	13	256.00	0.218	0.659	70.276	13	326.25	0.201	0.640	**69.973**
DHLx2	13	246.25	0.267	0.712	74.632	13	229.75	0.239	0.706	**74.124**	13	224.75	0.238	0.672	74.128
DHLx3	14	307.00	0.404	0.769	69.692	14	311.75	0.391	0.780	**68.992**	14	276.00	0.390	0.762	69.122
DLLx1	13	255.00	0.194	0.611	70.274	13	252.50	0.206	0.615	70.176	13	252.25	0.203	0.610	**70.043**
DLLx2	13	234.25	0.245	0.722	74.343	13	256.75	0.263	0.693	74.048	13	248.50	0.255	0.681	**73.669**
DLLx3	14	266.80	0.318	0.765	69.781	14	280.63	0.318	0.820	69.704	14	387.75	0.317	0.760	**68.931**
Avg.	17.43	458.96	0.360	0.824	86.109	17.50	472.37	0.368	0.830	85.488	17.50	465.50	**0.353**	0.809	**85.066**
Norm.	1.000	1.000	1.000	1.000	1.000	1.004	1.029	1.020	1.007	0.993	1.004	1.014	**0.981**	0.982	**0.988**

summarize the cell-level results for 48 combinational cells and 14 sequential cells. Consistent with the timing-oriented goal of this study, we use D_{cell} as the primary metric, while WL, sensitive-pair coupling capacitance, and power are reported as supporting indicators.

4.1.1 ***Combinational cells.*** For combinational cells, optimization headroom is inherently limited because simple cells contain only a few internal nets and short critical paths, and delay reduction is primarily observed in multi-stage logic gates with cascaded internal nets. ScoutCell improves the average D_{cell} by 0.56% over Min-Area & Min-WL and by 0.21% over MinDCell, while delivering the best average D_{cell} in five out of seven logic classes. Notably, ScoutCell does not consistently minimize WL, but it achieves the lowest sensitive-pair C_c (0.950) while maintaining essentially unchanged normalized power (1.002). The clearest gains appear on multi-stage families such as AND/OR and AO/OA, where cascaded nets expose the delay-critical coupling that sensitivity guidance targets.

4.1.2 ***Sequential cells.*** For sequential cells, ScoutCell again achieves the best average D_{cell}, improving it by 1.21% over Min-Area & Min-WL and by 0.49% over MinDCell, and obtains the best D_{cell} on 11 out of 14 flip-flops and latches while achieving the lowest sensitive-pair C_c (0.981) and power (0.982). Representative cases such as DFFLQx4 show that the best timing does not necessarily coincide with the shortest WL: despite a longer WL, ScoutCell still improves D_{cell} by about 2.8% and 2.1% over the two baselines. Overall, pairwise sensitivity guidance yields more consistent delay reduction across this sequential-cell set than MinDCell[3].

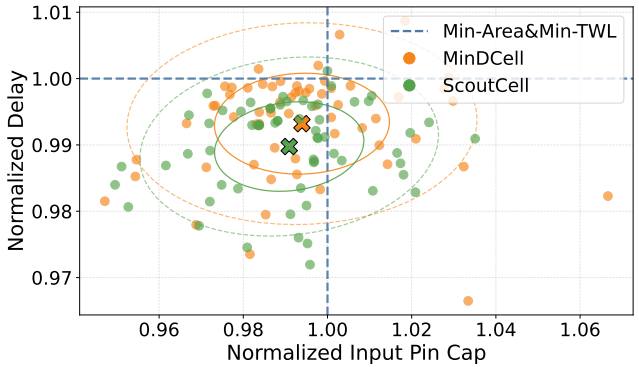


Figure 5: Scatter plot of 62 cells synthesized with *MinDCell* and *ScoutCell*. Each point corresponds to one cell. The x-axis is the normalized average input pin capacitance, and the y-axis is the normalized delay (D_{cell}), where both quantities are normalized to the *Min-Area&Min-TWL* result of the same cell. The ellipses summarize the distribution.

4.1.3 ***Input pincap comparison.*** To comprehensively evaluate a cell's timing contribution to the overall design, we must consider not only its intrinsic delay (D_{cell}) but also its input pin capacitance. Although input pin capacitance is a characterized equivalent metric, it is highly dependent on the cell's internal routing parasitics. As illustrated in Fig. 5, while both methods reduce the average input pin capacitance compared to the baseline (MinDCell by 0.87% and ScoutCell by 0.91%), MinDCell exhibits significant instability. The outliers in the bottom-right quadrant indicate that MinDCell struggles to bound parasitic degradation on input nets while minimizing local delay. In contrast, ScoutCell demonstrates a much tighter and more robust distribution. Because ScoutCell explicitly optimizes

Table 3: Post-route full-flow results under the common logic-synthesis and physical-implementation flow. Area is reported as final physical area in μm^2, WL is the total routed wirelength in μm, WNS and TNS are in ns, and the post-route achievable frequency $F_{\max}$ is reported in GHz and derived from post-route WNS with $F_{\max} = 1/(T_{\text{target}} - \text{WNS})$. `ScoutCell` achieves the best average WNS, TNS, and post-route achievable frequency across the designs.

Design	Area			WL			WNS			TNS			Fmax		
	Baseline	MinDCell [3]	ScoutCell	Baseline	MinDCell [3]	ScoutCell	Baseline	MinDCell [3]	ScoutCell	Baseline	MinDCell [3]	ScoutCell	Baseline	MinDCell [3]	ScoutCell
WB_DMA	9701	9211	9392	64430	60008	61598	-0.096	-0.067	**-0.045**	-20.202	-13.566	**-11.367**	2.983	3.265	**3.517**
MEM_CTRL	17704	18186	17992	85751	85806	84705	-0.076	-0.073	**-0.042**	-11.028	-8.099	**-7.297**	1.842	1.852	**1.965**
AC97_CTRL	28971	28897	28823	114431	115503	114964	-0.065	-0.046	**-0.043**	-18.765	-14.273	**-11.651**	3.556	3.814	**3.858**
USB_FUNC	23807	24164	24478	99267	98076	96887	-0.095	-0.065	**-0.032**	-9.041	-6.457	**-5.477**	2.113	2.256	**2.437**
DES3	25205	25756	24456	119838	124268	115466	-0.066	-0.062	**-0.048**	-7.604	-6.502	**-5.306**	1.836	1.849	**1.899**
WB_CONMAX	44352	43297	43117	710082	748536	736246	-0.086	-0.079	**-0.044**	-12.419	-5.847	**-1.602**	1.666	1.686	**1.792**
RS_DECODER	89668	91925	92187	343512	344527	351502	-0.086	-0.081	**-0.074**	-1.550	-1.637	-1.718	2.192	2.217	**2.252**
AES_128	413767	410717	413827	2864682	2797399	3074018	-0.081	-0.074	**-0.058**	-41.826	-38.862	**-31.828**	2.373	2.413	**2.510**
Norm.	1.000	1.001	0.997	1.000	1.000	1.003	1.000	0.845	**0.604**	1.000	0.774	**0.643**	1.000	1.036	**1.085**

internal coupling capacitance during the routing stage, it naturally keeps the dependent input pin capacitance well-controlled, effectively avoiding extreme high-pincap outliers.

4.1.4 ***Computational overhead.*** The computational overhead introduced by `ScoutCell` is manageable. Because the pairwise sensitivity prior is reusable across legal placement variants, the expensive SPICE-level characterization is incurred only once per cell on the anchor layout rather than during candidate exploration. Beyond this one-time extraction, the only additional overhead is the final sensitivity-guided coupling-aware IRR. For the complex sequential cell `DFFHQNx1`, anchor-layout sensitivity extraction takes 65.9 s using 8 parallel threads, candidate placement re-ranking takes 1.45 s, and the final coupling-aware optimization requires 333.54 s in total, including preprocessing and solving. Consequently, the absolute incremental runtime remains below 7 minutes for this sequential cell, and is typically reduced by one to two orders of magnitude for smaller combinational cells.

4.2 Block-Level Full-Flow Evaluation

To comprehensively evaluate whether the cell-level timing improvements translate into block-level benefits, we conduct a full logic synthesis and physical implementation flow. Following the evaluation methodology of [3], we configure three sets of cell libraries. The baseline library consists entirely of the *Min-Area & Min-WL* cells from Section 4.1. The three evaluation groups are constructed as follows: (1) **Baseline**, the pure Baseline library; (2) **MinDCell**, a mixed library of Baseline + MinDCell; and (3) **ScoutCell**, a mixed library of Baseline + ScoutCell. Providing these mixed libraries allows the synthesis and physical design tools to automatically exploit area-delay trade-offs during mapping and physical optimization stages.

To objectively evaluate the post-route achievable frequency, which we quantify as $F_{\max} = 1/(T_{\text{target}} - \text{WNS})$, determining an appropriate target clock period (T_{target}) is critical. Rather than treating $F_{\max}$ as a separately swept clock limit, we use this derived metric to compare how much frequency each library can support under the same calibrated timing target. Instead of applying a relaxed, arbitrary constraint, we employ a stringent constraint calibration methodology. Initially, logic synthesis is performed under progressively tightened clock periods to find a common T_{target} bound that avoids severe area and power overheads across all three libraries. During the subsequent physical design stage—with core utilization fixed at 0.6 to normalize routing congestion—we fine-tune this T_{target} to the exact threshold where all three groups successfully achieve a post-route WNS marginally exceeding -100 ps. This rigorous calibration ensures all libraries are evaluated under identical, near-limit timing stress, thereby guaranteeing a fair comparison of their true performance bounds.

Post-route results in Table 3 show that `ScoutCell` achieves the best average WNS, TNS, and post-route achievable frequency across the designs. On average, `ScoutCell` improves the post-route achievable frequency by 8.5% over the area/wirelength-driven baseline and 4.7% over the prior timing-driven baseline, indicating that the cell-level timing advantage remains visible after full logic synthesis and physical implementation. Furthermore, because `ScoutCell` inherently speeds up critical paths, it effectively relieves the burden on the physical design tool to rescue timing via aggressive buffering. Specifically, `ScoutCell` lowers the buffer-to-total-instance ratio to just 5.68%, compared to 8.66% in the baseline and 7.35% in MinDCell. It is this reduced reliance on buffer insertion that counterintuitively leads to a marginal reduction in the final post-route area (normalized to 0.997).

5 Conclusion

This paper presented `ScoutCell`, a sensitivity-guided framework for timing-driven standard cell synthesis that explicitly targets delay-critical coupling interactions in advanced nodes. The proposed characterize-then-optimize flow first extracts a reusable pairwise sensitivity prior from an anchor layout, then uses this prior to guide coupling-aware placement candidate screening and routing refinement through isolation penalties on harmful net-pair interactions. Experimental results on ASAP7 show that `ScoutCell` reduces the average cell delay by 0.6%/1.2% and 0.2%/0.5% on combinational/sequential cells over the area/wirelength-driven baseline and the state-of-the-art timing-driven synthesis method, respectively. These gains remain visible under a full logic-synthesis and physical-implementation flow, where `ScoutCell` further improves the average post-route achievable frequency by 8.5% and 4.7% over the two baselines.

Acknowledgement

This project is supported in part by Jiangsu National Science and Technology Major Project (SBG20250000204) and Beijing High Innovation Plan (202604841450).

References

[1] Chung-Kuan Cheng, Chia-Tung Ho, Daeyeal Lee, Bill Lin, and Dongwon Park. 2021. Complementary-FET (CFET) standard cell synthesis framework for design and system technology co-optimization using SMT. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 6 (2021), 1178–1191.

[2] Chung-Kuan Cheng, Andrew B Kahng, Byeonggon Kang, Seokhyeong Kang, Jakang Lee, and Bill Lin. 2025. SO3-Cell: Standard Cell Layout Automation Framework for Simultaneous Optimization of Topology, Placement, and Routing. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.

[3] Sehyeon Chung, Hyunbae Seo, and Taewhan Kim. 2025. Synthesis of Standard Cells of Minimum Delay. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–8.

[4] Lawrence T Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. ASAP7: A 7-nm finFET predictive process design kit. *Microelectronics Journal* 53 (2016), 105–115.

[5] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. 337–340.

[6] Kairong Guo, Haoran Lu, Rui Guo, Jiarui Wang, Chunyuan Zhao, Heng Wu, Runsheng Wang, and Yibo Lin. 2026. Standard Cell Layout Synthesis for Dual-Sided 3D-Stacked Transistors. In *2026 31st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 966–972.

[7] Chia-Tung Ho, Alvin Ho, Matthew Fojtik, Minsoo Kim, Shang Wei, Yaguang Li, Brucek Khailany, and Haoxing Ren. 2023. NVCell 2: Routability-Driven Standard Cell Layout in Advanced Nodes with Lattice Graph Routability Model. In *Proceedings of the 2023 International Symposium on Physical Design*. ACM, Virtual Event USA, 44–52.

[8] Byeonggon Kang, Ying Yuan, Yucheng Wang, Bill Lin, and Chung-Kuan Cheng. 2025. Cell-Flex Metrics for Designing Optimal Standard Cell Layout with Enhanced Cell Layout Flexibility. In *Proceedings of the 2025 International Symposium on Physical Design*. 164–172.

[9] Suwan Kim and Taewhan Kim. 2024. Optimal Transistor Folding and Placement for Synthesizing Standard Cells of Complementary FET Technology. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*. 1–6.

[10] Tai-Cheng Lee, Cheng-Yen Yang, and Yih-Lang Li. 2020. iTPlace: Machine Learning-Based Delay-Aware Transistor Placement for Standard Cell Synthesis. In *Proceedings of the 2020 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 159:1–159:8. doi:10.1145/3400302.3415613

[11] Zhiyuan Luo, Le Zhou, Zenghui Zhang, Jie Zhou, Zhien Li, Huiqing You, Feng Yao, and Zhenyu Zhao. 2025. ProtoCellLayout: Prototype-Guided Graph Learning for Accurate and Generalizable Standard Cell Layout PPA Estimation. In *Proceedings of the 2025 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9.

[12] OpenCores. [n. d.]. OpenCores Open Source Hardware IPs. https://opencores.org.

[13] Dongwon Park, Ilgweon Kang, Yeseong Kim, Sicun Gao, Bill Lin, and Chung-Kuan Cheng. 2019. ROAD: Routability Analysis and Diagnosis Framework Based on SAT Techniques. In *Proceedings of the 2019 International Symposium on Physical Design*. ACM, San Francisco CA USA, 65–72.

[14] Haoxing Ren and Matthew Fojtik. 2021. Standard Cell Routing with Reinforcement Learning and Genetic Algorithm in Advanced Technology Nodes. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. ACM, Tokyo Japan, 684–689.

[15] Yen-Ju Su, Jiun-Cheng Tsai, Hsuan-Ming Huang, Aaron C.-W. Liang, Han-Ya Tsai, Wei-Min Hsu, Jen-Hang Yang, and Charles H.-P. Wen. 2025. CoP&R: Co-Optimizing Place-and-Route for Standard Cell Layout via MCTS and AllSAT. In *Proceedings of the 2025 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9. doi:10.1109/ICCAD66269.2025.11240928

[16] The OpenROAD Project. 2024. AutoCellGen. https://github.com/The-OpenROAD-Project/AutoCellGen. Accessed: 2026-04-09.

[17] Jiun-Cheng Tsai, Wei-Min Hsu, Yun-Ting Hsieh, Yu-Ju Li, Wei Huang, CN Ho, Hsuan-Ming Huang, Jen-Hang Yang, Heng-Liang Huang, Aaron C-W Liang, et al. 2024. Maxcell: Ppa-directed multi-height cell layout routing optimization using anytime maxsat with constraint learning. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[18] Jiun-Cheng Tsai, Hsuan-Ming Huang, Wei-Min Hsu, Pei-Ting Lee, Jen-Hang Yang, Heng-Liang Huang, Yen-Ju Su, and Charles H-P Wen. 2025. Rescap: Fast-yet-accurate capacitance extraction for standard cell design by physics-guided machine learning. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*. 1243–1250.

[19] Pascal Van Cleeff, Stefan Hougardy, Jannik Silvanus, and Tobias Werner. 2020. BonnCell: Automatic Cell Layout in the 7-nm Era. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2020), 2872–2885.