A Region Adjacency Graph–Based Approach to Real-Time Wire Routing Synthesis with $\tau < 16$ms Constraints

Rabin Lamichhane
Computer Engineering
Pulchowk Campus
Lalitpur, Nepal
0009-0006-1215-3663

Abstract—**This paper presents a Hybrid Hierarchical Pathfinding system designed to achieve interactive latency ($\tau < 16$ms) for routing in obstacle-dense environments. Our method combines a high-level Region Adjacency Graph (RAG) abstraction with fine-grained grid A* search. Benchmarking demonstrated the system's success: at the highest complexity (100 obstacles), the Hybrid Router achieved a maximum path resolution time of 15.75ms, securing a maximum speedup of $20.6\times$ compared to the Baseline A* failure at 128.18ms. Critically, competitive algorithms like the Visibility Graph failed to scale, and suffered from poor aesthetics (TPL ≈ 0.051) and severe staircasing problems. The Hybrid Router maintains superior path quality (TPL ≈ 0.023) but confirms a necessary performance trade-off: average turn counts increased from 4.0 (Baseline A*) to 6.8 (Hybrid Router) due to RAG's Corridor Constrainment. The system successfully validates the strategy of prioritizing computational speed over path aesthetic optimality to enable responsive, interactive design in real-time systems.**

Index Terms—**Electronic Design Automation (EDA), Real-Time Routing, Hierarchical Pathfinding, Region Adjacency Graph (RAG), Maximal Empty Rectangle (MER), Rectilinear Routing, Congestion-Aware Routing, A* Search, Turns Per Unit Length (TPL), Interactive CAD Systems, Very Large Scale Integration (VLSI).**

I. Introduction

Modern electronic systems are becoming more complex, from high-density integrated circuits to multi-component control panels. This evolution calls for advanced tools for reliable and efficient physical circuit design. The primary task in synthesis is to connect discrete components (pins) via orthogonal paths (nets). The **Orthogonal Routing Problem** is the formal name for this fundamental issue in Electronic Design Automation (EDA).

Recent advancements in generative Artificial Intelligence (AI) have demonstrated impressive capabilities in conceptual circuit design. For instance, these models can generate functional **netlists** and suggest component values based on specifications. However, they inherently struggle with the subsequent and more demanding task: physical implementation and layout. The **wiring process** involves navigating a landscape of strict spatial and electrical constraints such as minimum spacing, wire gauge, and component placement within a limited 2D space. This non-local problem is often too distinct for generative AI to resolve directly. While AI succeeds at stating topological connectivity ("Pin A must join Pin B"), it typically fails to produce a geometric path that adheres to physical design rules without short circuits.

The central challenge addressed by this research is the performance gap between the algorithmic rigor required for pathfinding and the strict real-time constraints of an interactive system. Finding an optimal geometric path is resource-intensive; classic exhaustive search algorithms, such as maze routers, scale linearly with the grid size, exhibiting a time complexity of $O(W \times H)$ on a canvas of width W and height H [1]. Consequently, a generic grid-based search over a high-resolution canvas is infeasible for maintaining a 60 frames per second (FPS) interactive loop. Furthermore, the quality of the resulting path is paramount. To quantify path aesthetics, we introduce the metric **Turns Per Unit Length (TPL)**, defined as the ratio of the total number of 90-degree turns to the total wire length. Lower TPL values denote higher quality, smoother paths. Existing high-quality autorouting solutions often rely on iterative global optimization techniques, such as **rip-up and reroute**. While effective for offline compilation, these techniques require lengthy pre-computation or heavy server-side processing, thereby undermining the goal of responsive, integrated design [2]. While hierarchical pathfinding and Region Adjacency Graphs are common in robotics and game development, their **real-time** application to EDA requires unique adaptations. Specifically, it demands strictly **rectilinear enforcement**, precise handling of **electrical constraints** (such as T-junctions, net merging, and legal via placement), and the guarantee of a **manufacturable path**—conditions which standard, non-domain-specific RAG approaches do not natively support under $\tau < 16$ms constraints. This paper details the necessary modifications for industrial application.

To bridge this gap, a routing engine for **real-time**, quality-driven orthogonal routing under stringent time-budget constraints is presented in this paper. To address the $\mathcal{NP}$-hard nature of the problem, we propose a **Hybrid Hierarchical Pathfinding** approach that transforms the search space using a **low-entropy Region Adjacency Graph (RAG)** and **Maximal**

Empty Rectangle (MER) analysis. Our approach uses a two-tier search strategy and a capacity-aware cost model to ensure strictly legal, manufacturable paths within a single frame budget ($\tau < 16$ms). This engine effectively functions as the deterministic geometric kernel needed to accomplish connectivity derived from manual specifications as well as abstract generative models.

II. Related Work

A. Hierarchical Routing and Performance Constraints

Traditional routing for high-density designs is split into two sequential stages: **Global Routing** and **Detailed Routing** [3], [4]. In order to find a topological path, global routing abstracts the canvas into coarse tiles, or 'g-cells'. The detailed router then carries out segment-level geometric pathfinding, frequently using grid-based algorithms like **Lee's Algorithm** or **A* Search** [5]. However, achieving 100% routability in complex layouts often requires iterative relaxation techniques, such as **rip-up and reroute** [2]. Runtimes measured in seconds or minutes result from this process, making it inappropriate for interactive systems that need path resolution in milliseconds ($\tau < 16$ms). As a result, our work stands out for its dynamic, on-demand decomposition, which avoids drawn-out pre-computation and iterative cycles.

B. Spatial Decomposition and Graph Abstraction

Spatial abstraction is required to overcome the $O(W \times H)$ complexity of grid-based search. While techniques such as **Visibility Graphs** are widely used in general path planning, they are not well suited for the rectilinear constraints of VLSI. A more pertinent strategy is using **Maximal Empty Rectangles (MER)** [6], [7]. In rectilinear placement and routing, MERs divide the available space into the largest possible orthogonal rectangles. A **Region Adjacency Graph (RAG)** is created from this decomposition [7], where each MER is a node and edges link regions that share an orthogonal boundary of non-zero length. This RAG serves as a high-level, low-complexity pathfinding graph. Our system effectively generates a set of non-overlapping rectangular partitions suitable for real-time updates using a specialized recursive subtraction algorithm.

C. Obstacle-Avoiding Rectilinear Steiner Minimal Tree (OARSMT)

The routing of multi-terminal nets is governed by the **Rectilinear Steiner Minimal Tree (RSMT)** problem, which is $\mathcal{NP}$-hard [8]. The presence of obstacles introduces the **Obstacle-Avoiding RSMT (OARSMT)** problem, usually solved using heuristic maze routing kernels [9], although highly optimized approximations like **FLUTE** [10] exist for obstacle-free environments. Our methodology addresses this challenge by using geometric projection to dynamically evaluate the cost function, $f(n)$, as a constrained query from the current pin/segment to all available segments of the target net. This converts the computationally expensive global OARSMT search into a set of efficient point-to-segment pathfinding queries, offering a high-quality, real-time approximation that meets interactive performance requirements.

III. Proposed Hybrid Pathfinding Methodology

To achieve the $\tau < 16$ms interactive constraint while maintaining manufacturable path quality (low TPL), we propose a two-tier pathfinding architecture that drastically reduces the search space complexity from $O(W \times H)$ to a graph traversal problem $O(|V| + |E|)$. This methodology is based on the principle of separating the global, topological path from the local, detailed geometry.

A. Motivation: The State Space Explosion

Standard grid-based pathfinding algorithms, such as basic A* or Lee's maze router, operate on a discrete grid, where the state space size is proportional to the canvas dimensions ($W \times H$). For a high-resolution engineering canvas (e.g., 5000×5000 cells), the search space can exceed **25 million nodes**. Since the majority of this space is free, contiguous area (often referred to as *empty void*), a uniform grid search is computationally wasteful, treating open space with the same search granularity as complex obstacle clusters.

The solution is abstraction: grouping contiguous empty grid cells into larger geometric primitives. This dramatically reduces the search graph from millions of grid cells to a set of hundreds of abstract regions ($N_{\text{regions}} \ll N_{\text{pixels}}$), making real-time, high-level pathfinding feasible.

B. Theoretical Model: Non-Overlapping Empty Rectangles

We decompose the free space $\mathcal{F}$ into a finite set of axis-aligned, non-overlapping rectangles $\mathcal{R}$. This decomposition satisfies two requirements essential for a clean topological graph:

1) **Coverage:** The union of all rectangles precisely equals the free space: $\bigcup_{r \in \mathcal{R}} r = \mathcal{F}$.
2) **Disjointness:** No two rectangles overlap: $\forall r_i, r_j \in \mathcal{R}, i \neq j \implies r_i \cap r_j = \emptyset$.

Each rectangle $r \in \mathcal{R}$ forms the fundamental geometric primitive for the high-level pathfinding stage and is defined by its boundary coordinates $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$ **Figure 1**

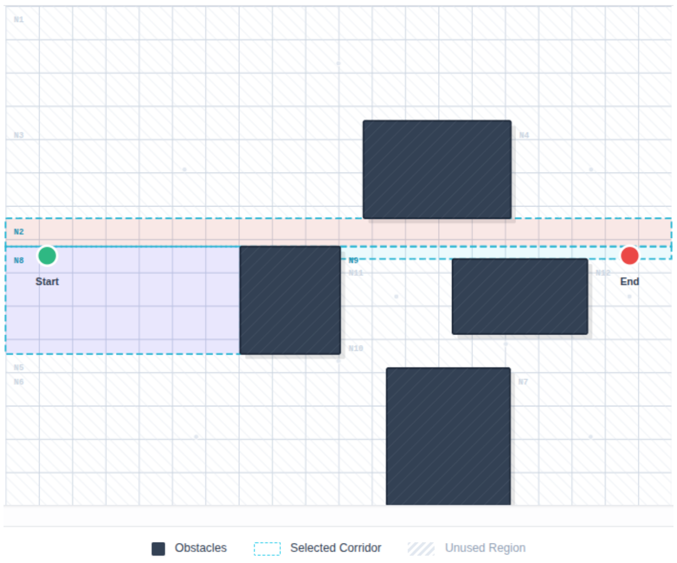

Fig. 2. **Hierarchical Pathfinding Corridor.** The result of the Global Routing phase on the Region Adjacency Graph (RAG), defining a high-level, collision-free channel for the subsequent Bounded A* search.

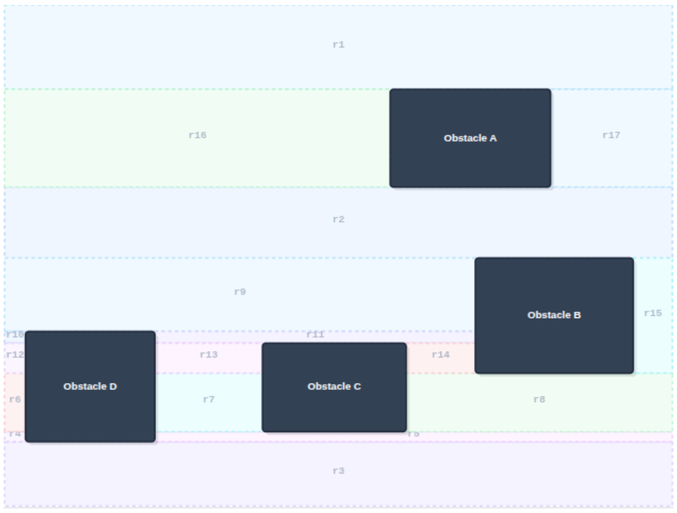

Fig. 1. Decomposition of the circuit routing free space $\mathcal{F}$ into a finite set of non-overlapping, axis-aligned rectangles $\mathcal{R}$. Obstacles (dark shaded areas) are excluded, and the free space is tiled by the resulting empty rectangles r_i to facilitate pathfinding.

C. Implementation: Recursive Subtraction Algorithm

The set $\mathcal{R}$ is generated dynamically within a localized bounding box using a **Recursive Subtraction Algorithm** [7]. Conceptually, this method starts with the largest empty rectangle (the local search area) and recursively subtracts obstacles until a complete, non-overlapping cover is achieved.

The recursive subtraction procedure produces a coarse-grained spatial abstraction suitable for high-level routing and global pathfinding, as illustrated in Fig. 2.

1) The Algorithm and Pivoting Strategy: The function Decompose(R, O) takes a current rectangle candidate R and the set of relevant obstacles O.

1) **Base Case:** If R contains no intersecting obstacles or existing routed wires, R is added to the set $\mathcal{R}$, and the recursion terminates.
2) **Pivoting and Splitting:** If R intersects an obstacle $o \in O$, the obstacle serves as a pivot, splitting R into a maximum of four new, non-overlapping sub-rectangles: $R_{\text{top}}, R_{\text{bottom}}, R_{\text{left}}, R_{\text{right}}$. To ensure the *Disjointness* property, the R_{left} and R_{right} sub-rectangles are **vertically constrained** to the height of o, preventing overlap with the upper and lower pieces.
3) **Recursion:** The Decompose function is recursively applied to each valid sub-rectangle.

This pivoting strategy guarantees a minimal, non-redundant coverage of the free space while keeping the decomposition complexity proportional to the number of local obstacles.

D. Region Adjacency Graph (RAG) Construction

The set of generated empty rectangles $\mathcal{R}$ forms the vertices of the **Region Adjacency Graph** (RAG), $G_{\text{RAG}} = (V, E)$, which encodes the topological connectivity of the free space.

1) Vertex Metadata: Each vertex $v_i \in V$ (corresponding to $r_i \in \mathcal{R}$) stores essential metadata:

- **Coordinates:** $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$.
- **ID:** A unique identifier used for graph traversal.
- **Density Score (ρ_i):** Represents the estimated local routing congestion.
- **State Flags:** Indicate whether the rectangle is fully traversable or contains a source/target pin.

2) Edge Definition: An edge $e_{ij} = (v_i, v_j) \in E$ is added if and only if two rectangles r_i and r_j share a contiguous, non-zero-length orthogonal boundary segment. Connections that share only a single corner vertex are ignored, as they represent topological pinch points that violate strict rectilinear routing assumptions and complicate cost computation.

E. Capacity Estimation via Sub-Gridding

A purely geometric abstraction, where a rectangle r_i is treated as a single, homogeneous node, is insufficient for modern routing. While a large, empty rectangle implies traversability, if it is already crossed by multiple existing wires, it represents a congested path. To model this capacity constraint, we introduce the **Density Score** (ρ_i).

To capture congestion effects within geometrically large but partially occupied regions, each rectangle is analyzed using a virtual sub-gridding scheme, as shown in Fig. 3.

1) The Density Score ρ_i: To estimate local routing density, each rectangle r_i is virtually subdivided into a grid of uniform **sub-squares** $S_i = \{s_{i,1}, \ldots, s_{i,N}\}$ based on a characteristic feature size (e.g., the wire pitch). The density score ρ_i is calculated as the ratio of sub-squares intersected by existing routed wire segments (W_{exist}) to the total number of sub-squares $N_i = |S_i|$:

$$\rho_i = \frac{1}{N_i} \sum_{k=1}^{N_i} \mathbb{1}(s_{i,k} \cap W_{\text{exist}} \neq \emptyset) \tag{1}$$

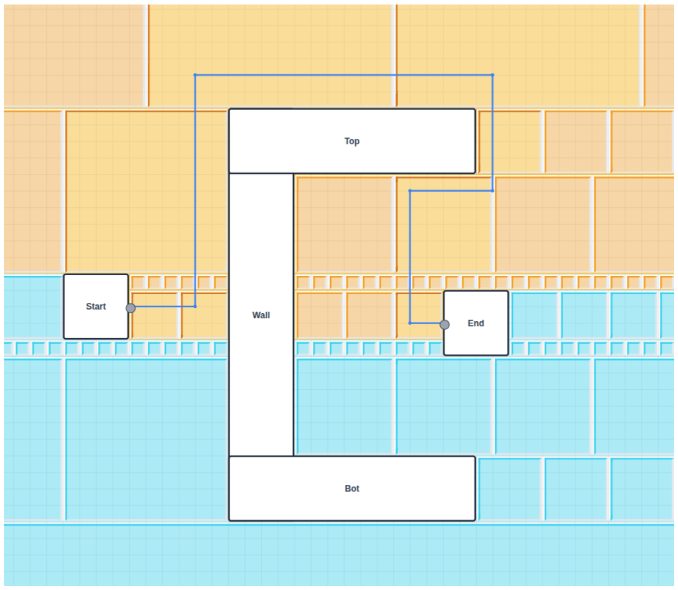

Fig. 3. Free space is partitioned into regions ($\mathcal{R}$) for global pathfinding. Density Score (ρ_i), calculated via a sub-grid, guides the congestion-aware path (blue line).

where $\mathbb{1}(\cdot)$ is the indicator function. The resulting score, $\rho_i \in [0, 1]$, allows the router to differentiate between an empty region ($\rho_i \approx 0$) and a congested bottleneck ($\rho_i \approx 1$).

2) RAG Cost Function: The cost of traversing an edge e_{ij} (moving from region r_i to r_j) is defined to prioritize short paths through uncongested regions. The cost function $c(e_{ij})$ combines the geometric distance with a penalty factor derived from the target region's density:

$$c(e_{ij}) = \text{Dist}_M(\mathbf{p}_i, \mathbf{p}_j) \cdot \left(1 + \alpha \cdot \rho_j^2\right) \tag{2}$$

where $\text{Dist}_M(\mathbf{p}_i, \mathbf{p}_j)$ denotes the Manhattan distance between the centroids (or entry/exit points) of the regions, and α is a tunable parameter (set to $\alpha = 10$ in our implementation). The quadratic term ρ_j^2 exponentially penalizes highly congested regions, forcing the high-level A* search over G_{RAG} to detour around bottlenecks before detailed micro-routing begins.

IV. Pathfinding Algorithm and Multi-Terminal Net Handling

The detailed pathfinding layer operates within the high-level routing corridor identified by the Region Adjacency Graph (G_{RAG}) and employs a highly optimized A* variant designed to meet the stringent real-time requirements of interactive schematic design.

A. Bi-Directional A* Search

To satisfy the target latency constraint ($\tau < 16\,\text{ms}$), we employ a **Bi-Directional A* Search** algorithm. This method simultaneously expands two search frontiers: F_s from the source and F_t from the target. By restricting the detailed search to the corridor produced during the global routing phase, fast convergence is achieved while preserving rectilinear routing constraints.

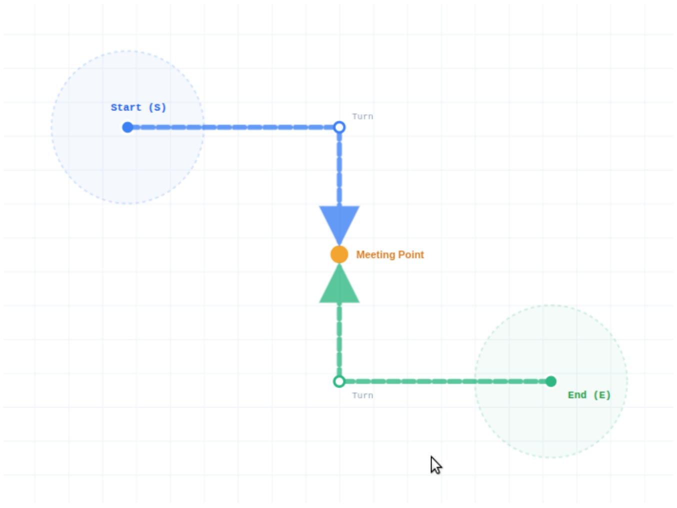

Fig. 4. **Bi-Directional A* Search.** Two-frontier expansion with a heavy turn penalty (C_{turn}) enforcing rectilinear aesthetics.

The dual-frontier expansion strategy used by the router is illustrated in Fig. 4, highlighting the influence of turn penalties on path geometry.

1) Complexity Reduction: In a conventional A* search, the explored area grows quadratically with path length (Area $\propto d^2$). By meeting in the middle, the maximum distance traversed by any single frontier is approximately halved ($\approx d/2$), resulting in a theoretical reduction of the combined search area:

$$A_{\text{Bi-Dir}} \approx 2 \cdot \left(\frac{d}{2}\right)^2 = \frac{d^2}{2} \tag{3}$$

This reduction is critical for maintaining interactive performance during long-distance routing across large canvases.

B. Aesthetic-Aware Cost Function

The detailed router is guided by the heuristic cost function $f(n) = g(n) + h(n)$, where $g(n)$ denotes the accumulated path cost from the source and $h(n)$ is the Manhattan-distance heuristic to the target. The accumulated cost function $g(n)$ is specialized to enforce high-quality rectilinear schematic aesthetics:

$$g(n) = g(\text{parent}) + C_{\text{base}} + C_{\text{turn}} + C_{\text{penalty}} \tag{4}$$

- $C_{\textbf{base}}$**:** The fundamental cost of traversing one grid unit.
- $C_{\textbf{turn}}$**:** A heavy penalty applied when the direction of travel changes. This cost is substantially larger than C_{base} and suppresses stair-casing artifacts, encouraging long, straight rectilinear segments (L- or Z-shaped paths).
- $C_{\textbf{penalty}}$**:** A moderate penalty applied when traversing a cell already occupied by an orthogonal wire, allowing necessary crossings while discouraging local congestion.

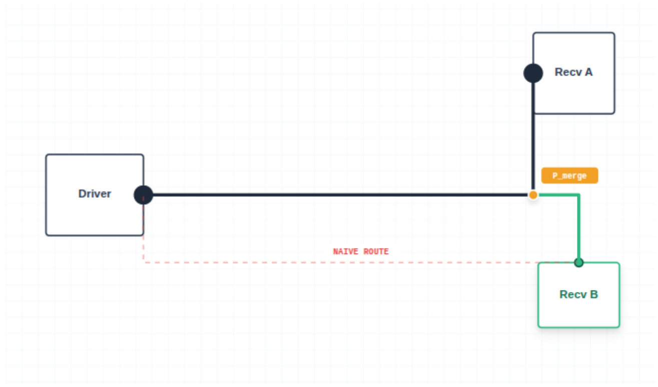

Fig. 5. **Conceptual Multi-Net Routing.** An illustration of the overall OARSMT approximation problem: connecting a new source pin S to an existing multi-terminal net $\mathcal{N}$ composed of pins A and B.

C. Multi-Terminal Net Approximation

For multi-terminal nets, the routing objective is to compute an efficient approximation of the **Obstacle-Avoiding Rectilinear Steiner Minimal Tree (OARSMT)**. When routing a source pin S to an existing net $\mathcal{N}$, the problem is reduced to a point-to-segment query that determines the optimal connection location, as conceptually illustrated in Fig. 5.

1) Geometric Projection and Merge Point: The router dynamically computes the optimal **merge point** (P_{merge}) on the existing net $\mathcal{N}$ that minimizes the connection cost, resulting in a valid T-junction. The geometric construction is illustrated in Fig. 6 and proceeds as follows:

1) **Segment Enumeration:** All linear segments L_i belonging to the target net $\mathcal{N}$ are retrieved.
2) **Projection:** The source point S is orthogonally projected onto the line supporting each segment L_i. The projection is clamped to the segment bounds, yielding a candidate point P_{proj}.
3) **Candidate Selection:** The optimal merge point is selected by minimizing the Manhattan distance to the source:

$$P_{\text{merge}} = \underset{P \in \{P_{\text{proj}}\}}{\arg\min} \left(|S.x - P.x| + |S.y - P.y|\right) \tag{5}$$

2) Aesthetic and Legal Checks: The computed merge point P_{merge} is validated for both geometric legality and schematic clarity. The point is checked against the Maximal Empty Rectangle (MER) structure to ensure it does not lie within an obstacle or an unroutable region. To improve visual clarity, P_{merge} is additionally constrained to be a minimum unit distance away from any component pin, ensuring a clearly distinguishable branched connection.

D. Port Directionality and Constraints

Component pins are modeled as **directed vectors** ($P_{\text{pos}}, \vec{V}_{\text{dir}}$) to enforce physical and aesthetic exit constraints. The detailed routing process is constrained at the source as follows:

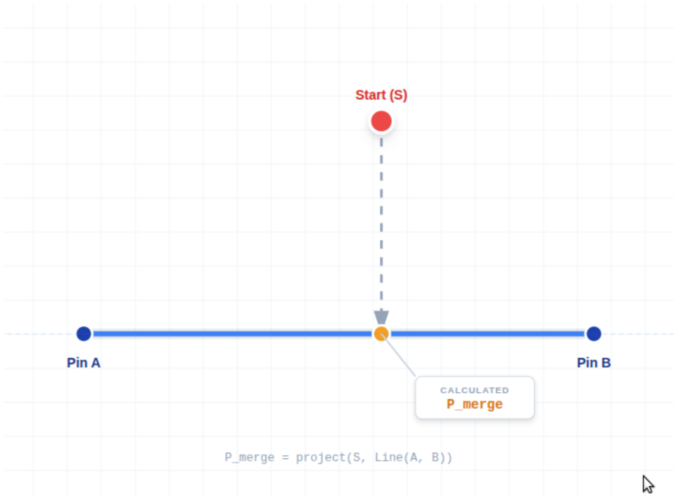

Fig. 6. **Geometric Projection for Multi-Terminal Routing.** Optimal T-junction merge point (P_{merge}) obtained by projecting the source pin (S) onto an existing net segment.

TABLE I
REFERENCE BENCHMARKING HARDWARE AND SOFTWARE ENVIRONMENT

Hardware Configuration (Lenovo LOQ 15IAX9E)	
Processor (CPU)	Intel Core i5-12450HX (12th Gen)
Memory (RAM)	12 GB DDR5
Software and Runtime Environment	
Implementation Language	TypeScript 5.x (transpiled to ES2022)
Runtime Engine	Google Chrome V8 (JIT compilation)
Operating System	Xubuntu 24.04.3 LTS
Kernel Version	Linux 6.14.0-37-generic
Target Latency (τ)	16 ms (60 FPS interactivity)

1) A mandatory stub segment is generated outward from the pin location P_{pos} in the component-defined direction $\vec{V}_{\text{dir}}$.
2) The A* search is initialized from the end of this stub (P_0).
3) Any immediate direction change at the pin location incurs a massive cost penalty ($C_{\text{enforcement}} \gg C_{\text{turn}}$), ensuring that the wire exits the component body cleanly before performing any turns.

V. Implementation and Performance Results

The proposed Hybrid Hierarchical Pathfinding engine was implemented in **TypeScript 5.x** and executed using the Google Chrome V8 JavaScript engine. All benchmarks were conducted in a controlled environment to evaluate runtime performance under interactive design constraints. The hardware specifications and runtime configuration used for evaluation are summarized in Table I.

A. Benchmark Results: Path Resolution Latency

To evaluate routing performance, we measured the **Path Resolution Latency** (PRL), defined as the average time re-

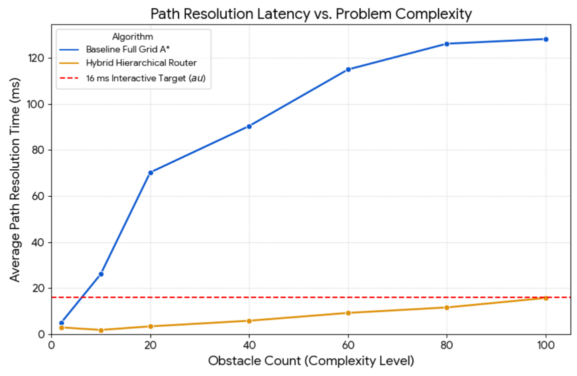

Fig. 7. **Path Resolution Latency vs. Problem Complexity.** Comparison of average routing time for the Baseline Full-Grid A* router and the Hybrid Hierarchical Router. The dashed line indicates the interactive latency target ($\tau = 16\,\text{ms}$).

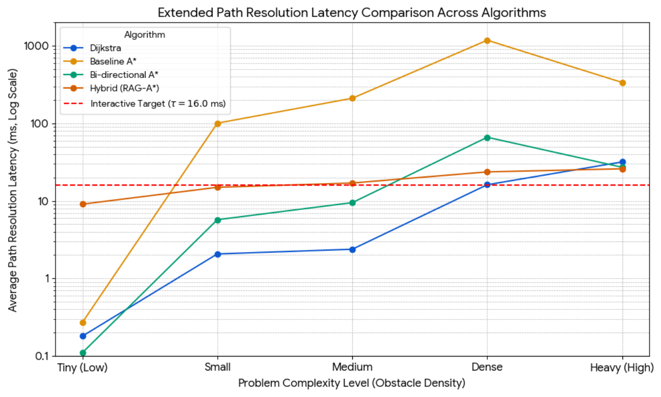

Fig. 8. **Extended Path Resolution Latency Comparison.** Comparison of average routing time for four algorithms across increasing problem complexity (obstacle density). The Y-axis is plotted on a logarithmic scale. The dashed red line indicates the interactive latency target ($\tau = 16\,\text{ms}$).

quired to compute a valid route between two terminals. Quantitative results comparing the Baseline Full-Grid A* router and the proposed Hybrid Hierarchical Router are reported in Table II and visualized in Fig. 7.

The performance trend across increasing obstacle density is illustrated in Fig. 7, where the dashed line indicates the target interactive latency threshold.

1) Latency Analysis: The Baseline A* router rapidly exceeds the interactive threshold due to state-space explosion, reaching a maximum latency of $128.18\,\text{ms}$. The Baseline A* is rooted in classic maze routing algorithms [5]. In contrast, the Hybrid Hierarchical Router consistently remains below the $16\,\text{ms}$ target, with a worst-case latency of $\mathbf{15.75}\,\text{ms}$. This improvement is enabled by the Region Adjacency Graph abstraction, based on the decomposition of free space into Maximal Empty Rectangles [6], [7], which dramatically reduces the effective search space and yields speedups ranging from approximately $8\times$ to $20\times$.

B. Extended Algorithmic Comparison

To contextualize the performance of the proposed Hybrid Hierarchical Router (labeled HPA* in the testing data), we evaluated it against classic and optimized pathfinding algorithms across various complexity levels. These include Dijkstra's algorithm (for optimal length without heuristic guidance), the Baseline A* (rooted in algorithms like Lee's [5]), and an optimized Bi-directional A* search. Table II presents the latency (Time in ms) and aesthetic quality (Turns Per Unit Length, TPL) for these algorithms under increasing density levels (Tiny to Heavy).

The latency trends from Table II are visualized in Fig. 8, plotted on a logarithmic scale to highlight the differential performance across the interactive threshold ($\tau = 16\,\text{ms}$).

1) Analysis of Alternative Approaches: The experimental data confirms the computational cost of global search. Classic maze routing approaches, such as the Baseline A* (rooted in algorithms like Lee's [5]) and Dijkstra's algorithm, experience massive latency spikes under dense conditions, exceeding the interactive threshold in nearly all but the lowest complexity levels.

The optimized Bi-directional A search showed promise, achieving faster routing times than the Baseline A and even occasionally the Hybrid Router in less complex scenarios ($\text{Small Time} : 5.72\,\text{ms}$ vs. $14.98\,\text{ms}$). However, its performance was highly **inconsistent** and it failed the interactive constraint at medium and high densities ($\text{Dense Time} : 66.22\,\text{ms}$). While this approach generally exhibited better aesthetics than the Hybrid Router, its inconsistency makes it unreliable for a real-time system.

Furthermore, early exploration of the **Visibility Graph** approach showed superior performance, even outperforming the Hybrid Router, in small routing fields (up to 1000 obstacles). However, this method proved to be inherently unstable and consistently failed to scale to massive complexity levels (10,000+ obstacles), where the Hybrid Router remained functional, albeit with increased latency (e.g., approximately $7000\,\text{ms}$ for $10,000$ obstacles, largely attributed to JavaScript runtime overhead). **Crucially, even when fast, the Visibility Graph suffers from poor aesthetic quality, exhibiting a higher number of turns per unit length (TPL** ≈ 0.051**) and leading to significant staircasing problems compared to the Hybrid Router's superior TPL** ≈ 0.023. This critical performance and aesthetic trade-off is clearly visualized in Figure 11.

C. Routing Quality and the Constrainment Trade-off

1) Wire Length Inflation: The average routed wire length produced by the Hybrid router was $1631\,\text{px}$, compared to the obstacle-free Manhattan minimum of $1309\,\text{px}$, resulting in an **length inflation factor of** $1.25\times$. This increase is expected, as the router must explicitly avoid physical obstacles and existing routes, unlike the theoretical lower bound.

2) Turn Count Trade-off and Corridor Constrainment: The average turn count for the Baseline router was $\text{avg}(\text{Turns}) = 4.0$ while the Hybrid router produced $\text{avg}(\text{Turns}) = 6.8$. This increase is a direct consequence of **Corridor Constrainment**:

TABLE II
EXTENDED ALGORITHMIC COMPARISON: LATENCY (TIME IN MS) AND AESTHETIC QUALITY (TPL)

Algorithm	**Tiny (Low)**		**Small**		**Medium**		**Dense**		**Heavy (High)**	
	Time (ms)	**TPL**	**Time (ms)**	**TPL**	**Time (ms)**	**TPL**	**Time (ms)**	**TPL**	**Time (ms)**	**TPL**
Dijkstra	0.18	0.571	2.07	0.217	2.38	0.08	16.17	0.278	31.85	0.28
Baseline A*	0.27	0.286	100.6	0.083	210.95	0.042	1182.76	0.02	336.72	0.023
Bi-directional A*	0.11	0.286	5.72	0.125	9.5	0.043	66.22	0.04	27.15	0.023
Hybrid (RAG-A*)	9.08	0.286	14.98	0.13	17.01	0.046	23.67	0.057	25.95	0.023

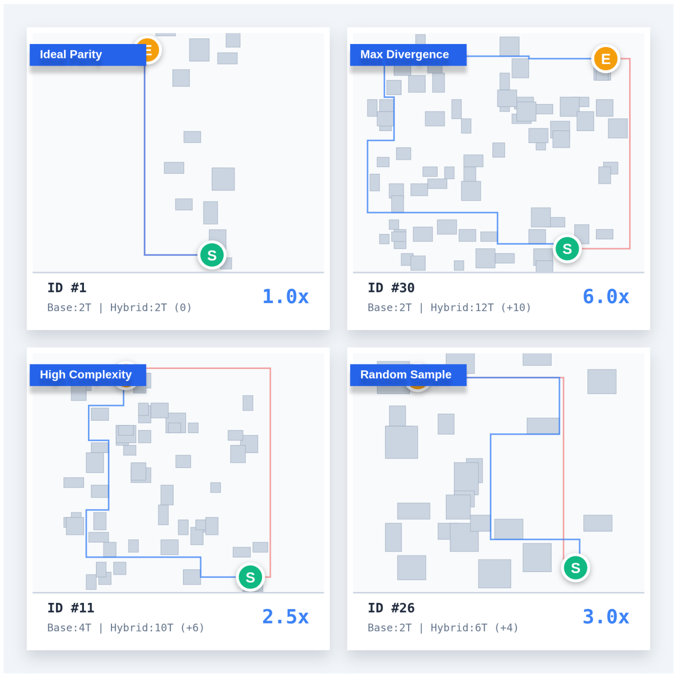

Fig. 9. **Routing Quality Trade-off Analysis.** Controlled visual examples showing the effects of corridor constraint on path optimality, including comparisons of turn counts (T) between the Baseline A* and the Hybrid Router under various obstacle densities.

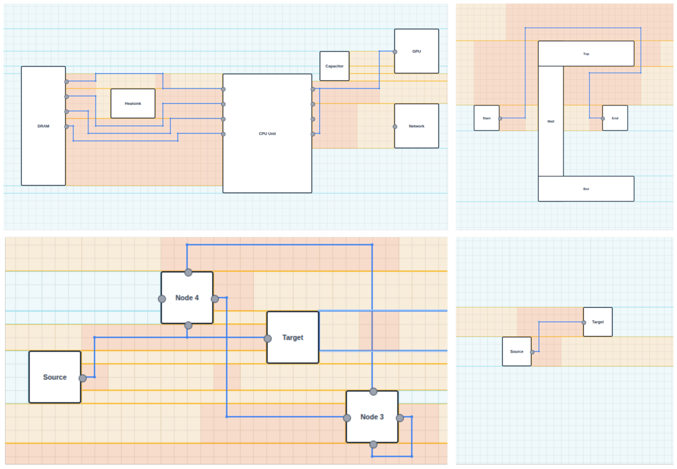

Fig. 10. **Final Implementation and Simulation Results.** A collage illustrating the Hybrid Router operating across different scales and complexities: multi-terminal routing on a complex chip-level design (top left) and varied single-net routing scenarios demonstrating obstacle avoidance and region-based pathfinding.

the Hybrid router commits early to a sequence of regions selected by the RAG search and restricts detailed routing to this corridor. While a global full-grid A* search can globally minimize turns, the Hybrid router may introduce local staircasing effects when transitioning between diagonally aligned regions, explicitly trading aesthetic optimality for interactive speed.

Representative high-complexity routing outcomes are shown in Fig. 9 and final simulation results are shown in Fig. 10.

3) Analysis of Alternative Approaches: The experimental data confirms the computational cost of global search. Classic maze routing approaches, such as the Baseline A* (rooted in algorithms like Lee's [5]) and Dijkstra's algorithm, quickly experience massive latency spikes under dense conditions, exceeding the interactive threshold.

The optimized Bi-directional A* search showed promise in low complexity but proved highly **inconsistent**, failing the τ constraint at medium and high densities (Dense Time : $66.22\,\mathrm{ms}$). Furthermore, the **Visibility Graph** approach, while initially fast in small routing fields, was unstable and failed to scale to massive complexity ($10,000+$ obstacles). Crucially, the Visibility Graph approach suffers from poor aesthetic quality (TPL ≈ 0.051) and severe **staircasing** compared to the Hybrid Router's superior TPL ≈ 0.023. This critical trade-off between scale performance and aesthetic quality is visualized in Fig. 11.

D. Discussion

The experimental results validate the core architecture of the proposed Hybrid Hierarchical Pathfinding system. The most significant outcome is the successful mitigation of search complexity, enabling the router to consistently satisfy the $\tau < 16\,\mathrm{ms}$ interactive performance target even under high obstacle density. This is achieved by transforming the routing problem from a fine-grained grid search into an efficient, coarse-grained search over the Region Adjacency Graph.

The observed trade-off in routing quality is both quantifiable and acceptable. The $1.25\times$ wire length inflation reflects unavoidable obstacle detours, while the increase in turn count highlights the aesthetic cost of corridor constrainment. These results confirm that the system intentionally prioritizes low latency over absolute path optimality, making it well suited for responsive, interactive schematic and circuit design environments.

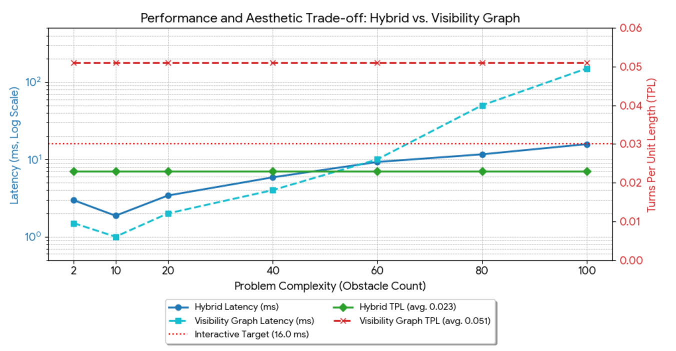

Fig. 11. **Performance and Aesthetic Trade-off: Hybrid Router vs. Visibility Graph.** A dual-axis plot comparing Latency (speed, Log Scale) and Turns Per Unit Length (TPL, aesthetic quality) against increasing complexity. Note the Visibility Graph's sharp latency increase and consistently higher TPL, highlighting its poor scalability and aesthetic issues.

VI. Conclusion and Future Work

A. Conclusion

This paper presented the design, implementation, and evaluation of a Hybrid Hierarchical Pathfinding system intended to meet the stringent interactive latency requirement of $\tau < 16$ms for complex routing problems.

The system validated its core hypothesis: using a high-level graph abstraction, specifically the Region Adjacency Graph (RAG), can effectively mitigate the state-space explosion inherent to full grid A* search. Performance results demonstrated the Hybrid Router's success, maintaining a Path Resolution Latency (PRL) of $\mathbf{15.75}$ms even at 100 obstacles, compared to the Baseline A* failure at 128.18ms. This hierarchical approach yielded a maximum speedup of $20.6\times$.

Comparative analysis against other optimized algorithms further confirmed the Hybrid Router's superiority for this domain:

- **Consistency:** The Bi-directional A* search proved inconsistent, failing the τ constraint at high complexity $(\text{Dense Time} : 66.22\text{ms})$.
- **Scalability and Aesthetics:** The Visibility Graph (VG) was initially fast, but fundamentally failed to scale to massive complexity $(10,000+$ obstacles). Furthermore, VG paths exhibited poor aesthetic quality (TPL ≈ 0.051) and severe **staircasing problems**, significantly worse than the Hybrid Router's TPL $\approx$ 0.023. This critical trade-off is clearly visualized in Figure 11.

Qualitatively, the Hybrid Router system exhibited two key behaviors:

- **Path Length:** The average wire length incurred an acceptable $1.25\times$ inflation compared to the theoretical minimum, successfully navigating obstacles while adhering to the cost function.
- **Aesthetics/Turns:** A substantial trade-off was observed; the average turn count for the Baseline A* was $\text{avg}(\text{Turns}) = 4.0$ while the Hybrid Router produced $\text{avg}(\text{Turns}) = 6.8$. This confirms the direct relationship between the massive speedup and the loss of path aesthetic optimality due to **Corridor Constrainment**.

In summary, the Hybrid Hierarchical Router successfully solves the primary challenge of achieving consistent interactive latency for complex routing, establishing itself as a viable foundation for a user-facing interactive design tool.

B. Future Work

Future efforts will focus on mitigating the turn count trade-off without compromising the achieved latency. This includes exploring techniques such as path post-processing (e.g., corner-cutting or path smoothing algorithms restricted to the RAG corridors) or dynamically adjusting the cost function's turn penalty based on the complexity of the local routing region.

References

[1] J. Cong and J. R. Shinnerl, *Global Routing in VLSI Design: Algorithms, Theory, and Implementation*. Boston, MA: Springer US, 2010, discusses the NP-hard nature and computational complexity of global routing.

[2] J. Chen, J. Liu, B. Li, Y. Wang, and L. Wen, "March: Maze routing under a concurrent and hierarchical scheme for buses," in *Proceedings of the 56th Annual Design Automation Conference (DAC)*. ACM/IEEE, 2019, pp. 1–6, details iterative routing schemes like rip-up and reroute for high completion rates.

[3] L. Yong, J. Zhang, and H. Fan, "A hybrid global and detailed routing framework for high-performance ic design," in *International Conference on ASIC (ASICON)*. IEEE, 2015, pp. 1–4, addresses hybrid approaches in VLSI context.

[4] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*, 3rd ed. Kluwer Academic Publishers, 1999, standard reference for VLSI routing and physical design.

[5] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, vol. 10, no. 3, pp. 346–365, 1961, the original paper on the maze routing algorithm (Lee's Algorithm).

[6] B. Chazelle, R. L. Drysdale, and D. T. Lee, "Computing the largest empty rectangle," *SIAM Journal on Computing*, vol. 15, no. 1, pp. 300–315, 1986, classic computational geometry work on maximal empty rectangles.

[7] M. Handa and R. Vemuri, "A fast algorithm for finding maximal empty rectangles for dynamic fpga placement," in *Proceedings of the conference on Design, Automation and Test in Europe*. IEEE, 2004, pp. 278–283.

[8] M. Hanan, "On steiner's problem with rectilinear distance," *SIAM Journal on Applied Mathematics*, vol. 14, no. 2, pp. 255–265, 1966, foundational paper establishing the Rectilinear Steiner Minimal Tree (RSMT) as a central routing problem.

[9] F. Y. Young and C.-Y. Liu, "Obstacle-avoiding rectilinear steiner tree construction," in *International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2008, pp. 20–25, describes maze-routing-based heuristics for OARSMT construction.

[10] C. Chu and Y.-C. Wong, "Fast and accurate rectilinear steiner minimal tree algorithm for vlsi design," in *International Conference on Computer Aided Design (ICCAD)*. IEEE, 2004, pp. 356–361, introduces FLUTE, a highly efficient RSMT approximation.