COMPARISON OF THE PERFORMANCE OF MAZE ROUTING AND RIP-UP & REROUTE ALGORITHMS IN VLSI ROUTING

Nguyen Thi Thanh Binh[1], **Nguyen Xuan Kien**[2*]
[1] Thai Nguyen University, [2] TNU - University of Information and Communication Technology

ARTICLE INFO

Received:	**19/02/2025**
Revised:	**07/5/2025**
Published:	**09/5/2025**

KEYWORDS

VLSI
Maze Routing
Rip-Up & Reroute
Circuit Routing
VLSI Simulation

ABSTRACT

This paper compares the performance of two routing algorithms, Maze Routing and Rip-Up & Reroute, in VLSI design, focusing on their ability to meet processing speed and routing quality requirements. Data collection and statistical analysis were conducted over 100 simulation runs on grid-based circuit layouts with two sizes (20×20 and 40×40) and two obstacle density levels (5% and 10%). In each simulation, a set of nets was randomly generated on cells free of obstacles. The evaluation metrics included execution time, path length, number of bends, routing success rate, and number of rip-ups. The results indicate that on the 20×20 grid, the execution time of Maze Routing (0.0032 ± 0.0051 seconds) in the scenario with 5% obstacles and 3 nets was about 10% of that of Rip-Up & Reroute (0.0336 ± 0.0522 seconds), while Maze Routing maintained a high routing success rate (88.7% compared to Rip-Up & Reroute’s 78.0%). On the 40×40 grid, Maze Routing continued to demonstrate fast processing times (from 0.0047 to 0.0075 seconds) and a low number of rip-ups. Overall, the study shows that Maze Routing excels in processing time and routing success rate, whereas Rip-Up & Reroute can achieve shorter path lengths in some scenarios but is hindered by higher computational costs and an increased number of rip-ups. This evaluation underscores the importance of balancing computational efficiency and path quality when selecting routing algorithms for VLSI applications.

SO SÁNH HIỆU NĂNG CỦA THUẬT TOÁN MAZE ROUTING VÀ RIP-UP & REROUTE TRONG ĐỊNH TUYẾN MẠCH VLSI

Nguyễn Thị Thanh Bình[1], **Nguyễn Xuân Kiên**[2*]
[1]Đại học Thái Nguyên, [2]Trường Đại học Công nghệ Thông tin và Truyền thông – ĐH Thái Nguyên

THÔNG TIN BÀI BÁO

Ngày nhận bài:	**19/02/2025**
Ngày hoàn thiện:	**07/5/2025**
Ngày đăng:	**09/5/2025**

TỪ KHÓA

VLSI
Maze Routing
Rip-Up & Reroute
Định tuyến mạch
Mô phỏng VLSI

TÓM TẮT

Bài báo thực hiện so sánh hiệu năng của hai thuật toán định tuyến Maze Routing và Rip-Up & Reroute trong thiết kế VLSI, về khả năng đáp ứng các yêu cầu về tốc độ xử lý và chất lượng đường đi. Thông qua thu thập dữ liệu và phân tích thống kê với 100 lần mô phỏng trên lưới vi mạch dạng lưới với hai kích thước (20×20 và 40×40) và hai mức mật độ vật cản (5% và 10%), các net được khởi tạo ngẫu nhiên trên các cell không chứa vật cản. Các chỉ số đánh giá bao gồm thời gian thực thi, độ dài đường đi, số lần uốn, tỷ lệ định tuyến thành công và số lần rip-up. Kết quả, trên lưới 20×20, thời gian thực thi của Maze Routing (0,0032 ± 0,0051 giây) ở kịch bản với 5% vật cản và 3 nets bằng 10% so với Rip-Up & Reroute (0,0336 ± 0,0522 giây), trong khi Maze Routing duy trì tỷ lệ định tuyến thành công cao (88,7% so với 78,0% của Rip-Up & Reroute). Trên lưới 40×40, Maze Routing vẫn có thời gian xử lý nhanh, (0,0047 đến 0,0075 giây) và số lần rip-up thấp. Nghiên cứu cho thấy Maze Routing có ưu thế về thời gian xử lý và tỷ lệ định tuyến thành công, trong khi Rip-Up & Reroute cho kết quả về độ dài đường đi ngắn hơn trong một số kịch bản nhưng bị hạn chế do chi phí tính toán cao và số lần rip-up tăng. Đánh giá này nhấn mạnh tầm quan trọng của việc cân bằng giữa hiệu quả tính toán và chất lượng trong việc lựa chọn thuật toán định tuyến cho các ứng dụng VLSI.

DOI: https://doi.org/10.34238/tnu-jst.12092

[*] Corresponding author. *Email: nxkien@ictu.edu.vn*

1. Introduction

In the context of the semiconductor industry's development, Very-Large-Scale Integration (VLSI) circuit design plays a crucial role in realizing complex electronic systems by integrating billions of components on a single chip. The VLSI design process requires the optimal placement of circuit elements and the resolution of routing issues to ensure that connections between components are efficient, adhere to physical and electrical rules, and optimize overall system performance [1].

Routing in VLSI design is critically important as it directly affects signal integrity, latency, power consumption, and noise immunity. As the number of elements increases and integration density grows, it becomes essential to design paths connecting functional blocks, manage spatial constraints, resolve routing conflicts, and address electromagnetic interference. Therefore, routing algorithms must be capable of finding optimal solutions that adapt to changing design requirements and the chip's physical conditions [2], [3]. Among the various approaches to solving the routing problem, the Rip-Up & Reroute (RR) and Maze Routing (MR) algorithms are prominent due to their effective handling of complex scenarios.

RR algorithm is based on an iterative mechanism: after initially establishing routing paths, the algorithm examines and "rips up" those paths that cause conflicts or fail to meet requirements, and then reroutes these connections in a more optimal manner. The strength of this method lies in its high flexibility and adaptability, enabling the system to adjust to achieve an optimal connectivity structure while minimizing network congestion and signal interference [4]. Various applications of RR in VLSI design have been demonstrated, such as an adaptive rip-up scheme that improved routing efficiency by reducing design rule violations by 69% and cutting runtime by 28% [5]. The integration of pin-access fine-tuning strategies further enhances compliance with strict design rules within RR processes [5]. Additionally, the RR technique plays a role in addressing security vulnerabilities and countering probing and routing attacks, utilizing frameworks such as Detour and Detour-RS [6], [7]. The application of reinforcement learning in RR processes has also been shown to help identify optimal networks for rerouting, effectively reducing short violations in global routing [8]. In reconfigurable system-on-chip designs, RR employs advanced algorithms to optimize connection routing by prioritizing efficient networks [9], among many other fields where RR is applied.

MR addresses the routing problem by traversing the circuit’s grid space to find the shortest path from the source to the destination. Its primary advantage is its ability to determine a path that satisfies design constraints with an optimal path length, thereby minimizing signal delay and improving overall circuit performance [10] – [12]. Notable implementations of MR include an efficient resource model and dynamic congestion sensitivity adjustment proposed for 3D maze routing, which optimizes resource distribution and the spatial components of the solution space, leading to improved routing quality and reduced runtime in 3D global routing [13]. The GAMER algorithm leverages GPU acceleration to enhance MR efficiency by decomposing the shortest path search into vertical and horizontal scanning operations, significantly reducing computational complexity and speeding up the system [14]. Convolutional neural networks have also been developed to predict RR quality by transforming the task into a binary classification problem, achieving high accuracy and speed [15], while other studies have exploited RR in various efficient routing applications.

Routing in VLSI circuit design is not merely an algorithmic challenge but a key factor determining the performance, reliability, and scalability of electronic systems. Both the RR and MR algorithms have their distinct advantages and contribute effectively to overcoming routing challenges, depending on the specific design requirements. These factors are critical for driving the VLSI industry toward sustainable and high-performance development in today's digital era. To assess routing capabilities in VLSI design, the authors compared and evaluated the performance of these two methods based on metrics such as routing runtime, path length, number of bends, routing success rate, and number of rip-ups. The simulation scenarios were repeated multiple times to collect a sufficiently large dataset for accurate statistical analysis. In each trial, both routing algorithms were executed independently on the same grid configuration and net list, from which the evaluation metrics were collected.

This paper presents a comprehensive quantitative evaluation of the MR and RR algorithms in VLSI routing. Through extensive simulations on grid-based layouts of 20×20 and 40×40 with varying obstacle densities, the study rigorously assesses key performance metrics including execution time, path length, number of bends, routing success rate, and rip-up count. The unique contribution of this work lies in its systematic comparison, which reveals that while the MR algorithm consistently achieves faster processing times and higher routing success rates. The RR algorithm can yield shorter paths in certain scenarios at the expense of increased computational costs and a higher frequency of rip-ups. By elucidating the trade-offs between computational efficiency and routing quality, this research not only deepens the understanding of these algorithms' operational characteristics but also provides valuable empirical insights for optimizing routing strategies in large-scale integrated circuit design systems.

2. Materials and Methods

2.1. Rip-Up & Reroute algorithm

The RR algorithm is an adaptive technique in VLSI design developed to resolve routing conflicts that occur when multiple nets must be routed simultaneously on the same grid, leading to interference or blockages. The RR mechanism is based on reassigning conflicting routes through an iterative process in which infeasible or suboptimal routes are ripped up and rerouted to improve the overall quality of the routing solution. The RR method is detailed as follows [4] – [9]:

Input: Consists of the following parameters:

- Routing Space: Represented as a graph *G = (V, E)*, where *V* is the set of grid points (nodes) and *E* is the set of edges connecting neighboring grid points. Each edge $e \in E$ has a weight *w(e)* representing the cost to traverse that edge.

- A set of nets $N = \{N_1, N_2, \ldots, N_k\}$ where each net $N_i = (s_i, t_i)$ is defined by a source point $s_i \in V$ and a target point $t_i \in V$.

- Each edge $e \in E$ has a capacity *c(e)*, indicating the maximum number of nets that can use that edge. The actual number of nets using edge *e* is denoted by *usage(e)*.

- When an edge *e* becomes congested, its weight is updated according to:

$$w(e) = w(e) + \alpha \cdot \text{of}(e) \tag{1}$$

where:

$$(e) = \max\{0, \text{usage}(e) - c(e)\} \tag{2}$$

and α is a tuning parameter to adjust the weight of congested edges.

Output: A set of feasible routes $P = \{P_1, P_2, \ldots, P_k\}$, where:

- Each route P_i connects the two points sisi and titi without violating the edge capacity constraints.

- The total cost of all routes is minimized:

$$C(P) = \sum_{i=1}^{k} \sum_{e \in P_i} w(e) \tag{3}$$

Step 1: Initialization

- Set the initial weights for all edges.

- Set the maximum number of iterations and initialize the iteration count.

- For each net, use a shortest path algorithm to find an initial route P_i, minimizing:

$$C(P_i) = \sum_{e \in P_i} w(e) \tag{4}$$

- Update *usage(e)* for all edges *e* in each route P_i.

Step 2: Rip-Up conflicting routes

- Identify congested edges, i.e., those for which *usage(e)>c(e)*.

- Remove all routes P_i that use any congested edge. For each removed route, update *usage(e)* to decrease the number of nets using that edge.

Step 3: Update edge weights

- For each congested edge, update its weight according to formulas (1) and (2).

- Edges that are not congested retain their weights.

Step 4: Reroute removed nets

- For each removed net, use the shortest path algorithm to find a new route P_i, minimizing the cost as in (3).

-Update *usage(e)* for all edges in the new route P_i .

Step 5: Check stopping condition

- If all nets are routed feasibly (i.e., no edge is congested) or if the maximum number of iterations is reached, stop the algorithm.

- Otherwise, increment the iteration count by 1 and return to Step 2.

2.2. Maze Routing algorithm

The MR algorithm is a routing technique used in VLSI design to optimally connect pins on a two-dimensional or three-dimensional grid. Its operation is based on finding the shortest path between two points in a grid space constrained by obstacles. MR relies on graph theory, where the grid is represented as a graph with grid points as vertices and edges representing feasible movement between neighboring points. The detailed implementation of the MR method is presented as follows [10] – [15]:

Input: Consists of the following parameters:

- Routing grid: The routing space is represented as a graph $G = (V, E)$, where V is the set of grid points defined by coordinates (x,y), and E is the set of edges connecting neighboring points. Each edge $e \in E$ has a weight *w(e)* representing the cost to traverse that edge.

- Source and Target Points: The source point $s \in V$ with coordinates (x_s,y_s); the target point $t \in V$ with coordinates (x_t, y_t).

- Obstacles: Grid points or edges that are blocked are assigned a weight $w(e)=\infty$ and are not used in the routing process.

- Cost Function: A function *d(v)* that stores the shortest path cost from the source ss to each vertex $v \in V$. Initially, $d(s)=0$ and $d(v)=\infty$ for all $v\neq s$.

Output: A feasible path $P = \{v_1, v_2, \ldots, v_k\}$ where $v_1 = s$ and $v_k = t$.

- Every consecutive pair $(v_i,v_{i+1}) \in E$

- The total cost *C(P)* of the path is calculated as:

$$C(P) = \sum_{i=1}^{k-1} w(e_i), \quad \text{with } e_i = (v_i, v_{i+1}) \tag{5}$$

Step 1: Initialization

- Set $d(s)=0$; Set $d(v)=\infty$ for all $v\neq s$.

- Use a priority queue Q, initially containing the source s: $Q=\{s\}$.

Step 2: Wave Propagation

While the queue Q is not empty:

Dequeue the first vertex v from Q: $v = Dequeue(Q)$

For each neighbor u of v:

If *d(u) > d(v) + w(v, u)* then:
Update *d(u) = d(v) + w(v, u)*
Add u to the queue: *Q = Q ∪{u}*
If the target t is labeled (i.e., *d(t)* < ∞) then:
Stop the wave propagation
If the queue Q becomes empty while *d(t) = ∞:*
Then no feasible path exists.
The cost update during wave propagation is given by:

$$d(u) = \min\{d(u), d(v) + w(v,u)\}, \quad \forall u \in \text{neighbors}(v) \tag{6}$$

Step 3: Path Backtracking
- Start from the target *t*.
- Initialize the path *P={t}*.
- Backtrack: While the current vertex *v≠s*:
+ Select a neighbor *u∈neighbors(v)* such that *d(u)+w(u,v)=d(v)*.
+ Prepend *u* to the path *P: P={u}∪P*.
+ Update *v=u*.
- Result: Return the path *P*.
- The backtracking function is defined as:

$$\%Formula(5)v_{i-1} = \arg\min_{u \in \text{neighbors}(v_i)} \tag{7}$$

3. Results and Discussion

The simulation scenarios during the routing process when designing VLSI are based on a square grid with different sizes (20×20 and 40×40) and varying obstacle densities (5% and 10%). Obstacles are randomly generated on the grid and marked with a value of -1. The list of nets is randomly initialized, with the source and target points located on cells that do not contain obstacles. The experiments are repeated 100 times to collect a sufficiently large dataset for statistical analysis. In each run, both routing algorithms are executed independently on the same grid configuration and net list, thereby collecting metrics on execution time, path length, number of bends, and success rate. The research objective is to evaluate the performance of the Rip-Up & Reroute (RR) and Maze Routing (MR) methods based on the following metrics:

- Execution Time: The computation time required to route all nets.
- Path Length: The number of cells in the discovered path.
- Number of Bends: The number of turns in the path, which may relate to the physical cost in circuit design.
- Routing Success Rate: The percentage of nets successfully routed.
- Rip-Up Count: The number of times nets have to be ripped up in the RR method.

The simulation program was implemented in MATLAB R2023b. This software was installed on a computer equipped with an Intel® Core™ i3-4130 CPU @ 3.40 GHz (4 CPUs) running the Windows 10 Pro 64-bit operating system. The simulation was carried out for both the RR and MR algorithms, and the results are presented in Table 1, Table 2, Figure 1, and Figure 2.

The data tables summarize the performance metrics of the MR and RR algorithms across various simulation scenarios. Table 1 presents the results for the 20×20 grid, while Table 2 focuses on the 40×40 grid. In Table 1, which provides the simulation results on a 20×20 grid, the execution time of MR is only a fraction of that of RR in all scenarios, especially when the number of nets is low. This is due to MR's breadth-first search strategy, which quickly identifies a path, whereas RR undergoes an iterative process that incurs higher computational costs. Although MR's paths are generally longer than those of RR, MR achieves a higher routing success rate. Specifically, MR maintains success rates ranging from 81.8% to 88.7%, compared to RR's 69.4% to 78.0%.

Table 1. *VLSI routing with 20x20 grid*

Metric	Scenario 1 (5% obstacles, 3 nets)	Scenario 2 (5% obstacles, 5 nets)	Scenario 3 (10% obstacles, 3 nets)
Maze Execution Time (s)	0.0032 ± 0.0051 (CI: 0.0022–0.0042)	0.0018 ± 0.0008 (CI: 0.0017–0.0020)	0.0014 ± 0.0010 (CI: 0.0012–0.0016)
Rip-Up Execution Time (s)	0.0336 ± 0.0522 (CI: 0.0232–0.0439)	0.0379 ± 0.0423 (CI: 0.0295–0.0463)	0.0224 ± 0.0335 (CI: 0.0157–0.0290)
Maze Path Length	13.82 ± 4.34 (CI: 12.96–14.68)	12.54 ± 3.62 (CI: 11.82–13.26)	13.99 ± 5.37 (CI: 12.92–15.06)
Rip-Up Path Length	12.14 ± 5.84 (CI: 10.98–13.30)	10.46 ± 4.35 (CI: 9.59–11.32)	11.98 ± 6.38 (CI: 10.71–13.24)
Maze Number of Bends	2.10 ± 1.08 (CI: 1.88–2.31)	1.98 ± 0.88 (CI: 1.80–2.15)	3.02 ± 1.68 (CI: 2.69–3.35)
Rip-Up Number of Bends	1.89 ± 1.22 (CI: 1.65–2.14)	1.69 ± 0.98 (CI: 1.50–1.89)	2.61 ± 1.79 (CI: 2.26–2.97)
Maze Success Rate (%)	88.7	81.8	86.7
Rip-Up Success Rate (%)	78.0	69.4	76.3
Average Rip-Up Count	33.01 ± 47.25	81.51 ± 72.91	37.48 ± 49.40

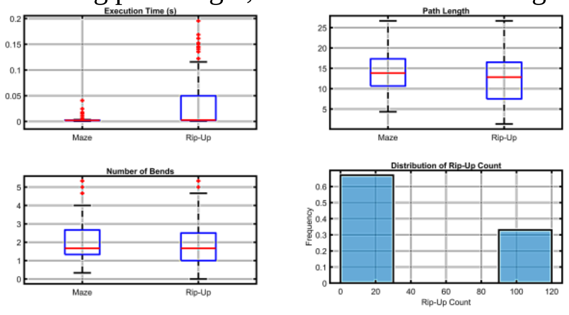
From the data in Table 2, for the 40×40 grid, the performance of both algorithms improves in terms of routing success rate. The execution time of MR remains low, between approximately 0.0047 and 0.0075 seconds, while RR exhibits greater variability. Although there is no statistically significant difference in path length and number of bends between the two methods on the 40×40 grid, MR still achieves a higher success rate and a lower rip-up count. The number of rip-ups for MR is consistently and significantly lower than for RR, indicating that MR's direct routing strategy effectively minimizes conflicts and re-routing.

Table 2. *VLSI routing with 40x40 grid*

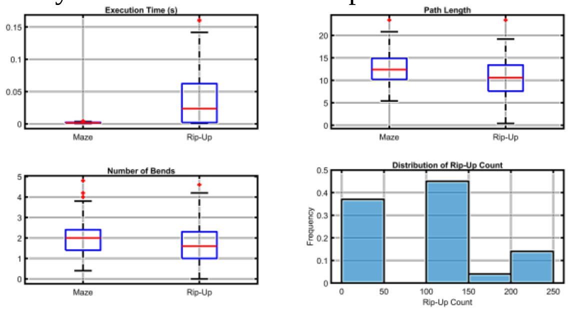
Metric	Scenario 5 (5% obstacles, 3 nets)	Scenario 6 (5% obstacles, 5 nets)	Scenario 7 (10% obstacles, 3 nets)
Maze Execution Time (s)	0.0047 ± 0.0024 (CI: 0.0043–0.0052)	0.0075 ± 0.0029 (CI: 0.0070–0.0081)	0.0048 ± 0.0018 (CI: 0.0044–0.0051)
Rip-Up Execution Time (s)	0.0334 ± 0.0780 (CI: 0.0180–0.0489)	0.0696 ± 0.1126 (CI: 0.0473–0.0920)	0.0328 ± 0.0840 (CI: 0.0161–0.0494)
Maze Path Length	27.83 ± 9.92 (CI: 25.86–29.80)	29.96 ± 9.29 (CI: 28.11–31.80)	29.37 ± 8.60 (CI: 27.66–31.08)
Rip-Up Path Length	25.97 ± 11.42 (CI: 23.71–28.24)	27.73 ± 10.57 (CI: 25.63–29.83)	27.81 ± 10.20 (CI: 25.79–29.83)
Maze Number of Bends	3.56 ± 1.85 (CI: 3.19–3.93)	4.17 ± 1.72 (CI: 3.83–4.51)	5.70 ± 2.36 (CI: 5.23–6.17)
Rip-Up Number of Bends	3.35 ± 1.95 (CI: 2.96–3.74)	4.02 ± 1.83 (CI: 3.65–4.38)	5.32 ± 2.38 (CI: 4.85–5.79)
Maze Success Rate (%)	95.3	90.6	95.7
Rip-Up Success Rate (%)	90.7	83.8	92.0
Average Rip-Up Count	14.00 ± 34.87	41.01 ± 55.21	12.01 ± 32.66

MR demonstrates a clear advantage in execution time, while RR can offer superior path quality (in terms of length and number of bends) in many scenarios, albeit struggling with increased conflicts as the number of connections rises. The charts presented in Figure 1 and Figure 2 provide a visual overview of the performance of MR and RR under different scenarios. Each figure includes four plots: Execution Time, Path Length, Number of Bends, and Distribution of Rip-Up Count. Through the analysis of these charts, we can draw insightful conclusions regarding the strengths and limitations of each algorithm.

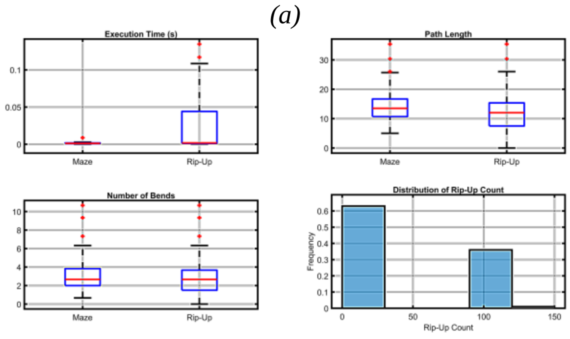
Figures 1illustrates the comparative results between the MR and RR algorithms in a 20×20 VLSI grid scenario with obstacle densities of 5% and 10%, and with 3 and 5 nets to be routed, respectively. In these scenarios, the execution time of MR is significantly lower than that of RR, especially when the number of nets increases or the obstacle density is higher (as observed in Figures 1c and Figure 1d). This is because MR does not require iterative conflict resolution steps, whereas RR incurs additional computational time due to its iterative process. However, RR produces shorter paths; specifically, at a 10% obstacle density, RR demonstrates an advantage in reducing path length, albeit at the cost of a significantly lower success rate compared to MR.



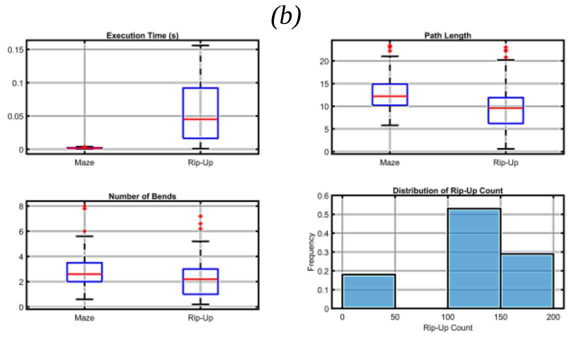
(a)



(b)

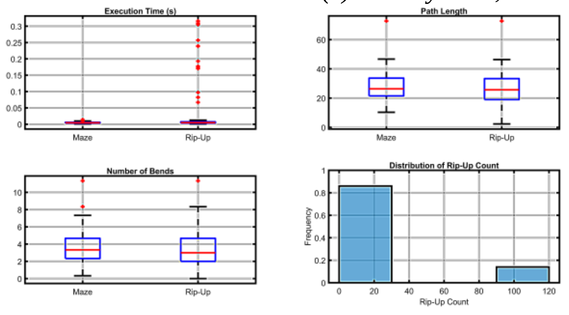


(c)

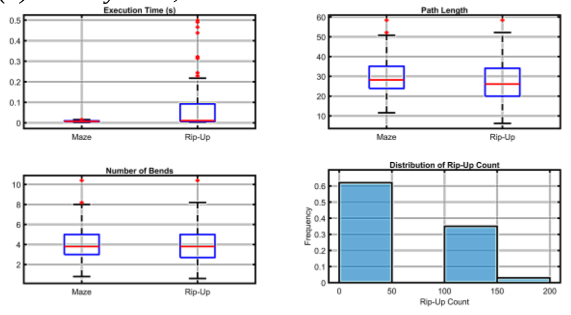


(d)

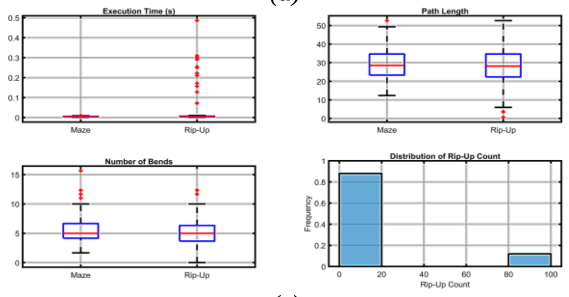
Figure 1. *Grid 20×20: (a) Density 5%, 3 nets; (b) Density 5%, 5 nets; (c) Density 10%, 3 nets; (d) Density 10%, 5 nets*



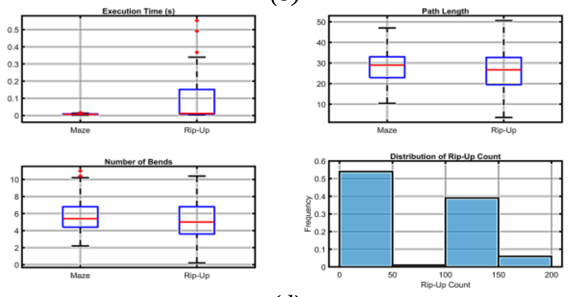
(a)



(b)



(c)



(d)

Figure 2. *Grid 40×40: (a) Density 5%, 3 nets; (b) Density 5%, 5 nets; (c) Density 10%, 3 nets; (d) Density 10%, 5 nets*

Figures 2 extends the analysis to a larger VLSI grid scenario with dimensions of 40×40, maintaining the same obstacle densities and net counts. MR continues to hold an advantage in execution time even as the grid size increases. However, the differences in path length between the two algorithms become less pronounced in scenarios with low obstacle density (Figures 2a and Figure 2b), likely due to the larger grid allowing MR to find feasible paths with fewer conflicts. Conversely, at higher obstacle densities (Figures 2c and Figure 2d), RR still optimizes path length effectively, but the significant increase in the number of rip-ups makes it less efficient when handling complex conflicts in larger grids.

In conclusion, the comparison of performance metrics indicates that the Maze Routing algorithm, with its fast processing time, high routing success rate, and low re-routing frequency, demonstrates overall superior performance compared to the Rip-Up & Reroute algorithm in the tested simulation scenarios. Conversely, although the Rip-Up & Reroute method may offer certain advantages in path length or reduced number of bends in specific cases, its high computational cost and unstable conflict resolution limit its practicality and effectiveness for VLSI applications that demand high speed and reliability. These results underscore the importance of balancing computational efficiency and solution quality when selecting a routing algorithm for large-scale integrated circuit design systems.

4. Conclusion

In this study, we compared and evaluated the performance of the MR and RR algorithms in VLSI design through a variety of simulation scenarios involving 20×20 and 40×40 grids, with obstacle densities of 5% and 10% respectively, and different net counts. The experimental results indicate that the MR algorithm processes significantly faster than RR, particularly in scenarios with fewer nets and lower obstacle densities. Although the paths generated by MR are generally longer than those produced by RR, MR achieves a higher routing success rate, demonstrating its stability and adaptability in complex routing environments. Conversely, while RR can optimize path length and minimize the number of bends in certain cases, it incurs higher computational costs due to the repeated ripping up and rerouting needed to resolve conflicts. This leads to a significant increase in the number of rip-ups as net count and obstacle density rise, thereby reducing the overall efficiency of RR in scenarios demanding high speed and reliability.

In summary, the paper emphasizes the importance of balancing computational efficiency and path quality when selecting routing algorithms for large-scale integrated circuit design systems. Its main contribution is the provision of a detailed experimental and statistical basis on the advantages and disadvantages of MR and RR, offering useful insights for applying these algorithms in practical VLSI design to optimize the performance and reliability of modern electronic systems.

REFERENCES

[1] X. Chen, G. Liu, N. Xiong, Y. Su, and G. Chen, "A Survey of Swarm Intelligence Techniques in VLSI Routing Problems," *IEEE Access*, vol. 8, pp. 26266-26292, 2020.

[2] K. Lakshmanna, F. Shaik, V. K. Gunjan, N. Singh, G. Kumar, and R. M. Shafi, "Perimeter Degree Technique for the Reduction of Routing Congestion during Placement in Physical Design of VLSI Circuits," *Complexity*, vol. 2022, no. 1, 2021, Art. no. 8658770.

[3] R. A. Solovyev *et al.*, "PAGR: Accelerating Global Routing for VLSI Design Flow," *IEEE Access*, vol. 13, pp. 6440-6450, 2025.

[4] M. B. Raith, “Rip-Up and Reroute Strategies,” in *Advanced Routing of Electronic Modules*, CRC Press, 2024, pp. 347-368.

[5] Z. Qi, J. Zhang, G.-L. Chen, and H. You, "Effective and Efficient Detailed Routing with Adaptive Rip-up Scheme and Pin Access Refinement," in *Proceedings of the Great Lakes Symposium on VLSI*, 2022, pp. 165-168.

[6] M. Gao and D. Forte, "Detour: Layout-aware Reroute Attack Vulnerability Assessment and Analysis," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, San Jose, CA, USA, 2023, pp. 122-132.

[7] M. Gao, L. K. Biswas, N. Asadi, and D. Forte, "Detour-RS: Reroute Attack Vulnerability Assessment with Awareness of the Layout and Resource," *Cryptography*, vol. 8, no. 2, 2024, Art. no. 13.

[8] U. Gandhi, E. Aghaeekiasaraee, P. Mousavi, I. S. K. Bustany, and L. Behjat, "Applying Reinforcement Learning to Learn Best Net to Rip and Re-route in Global Routing," *ACM Transactions on Design Automation of Electronic Systems*, vol. 29, no. 4, 2024, Art. no. 69.

[9] M. A. Zapletina, D. A. Zheleznikov, and V. M. Khvatov, "The Rip-up and Reroute Technique Research for Island-Style Reconfigurable System-on-Chip," in *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Saint Petersburg and Moscow, Russia, 2019, pp. 1593-1596.

[10] S. Yao, X. Yang, Z. Song, X. Yang, D. Duan and H. Yang, "Maze Routing: An Information Privacy-aware Secure Routing in Internet of Things for Smart Grid," *2022 7th International Conference on Communication, Image and Signal Processing (CCISP)*, Chengdu, China, 2022, pp. 461-465 .

[11] X. Jiang *et al.*, "FPGA-Accelerated Maze Routing Kernel for VLSI Designs," in *27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2022, pp. 592-597.

[12] Z. Fu *et al.*, "An Efficient Maze Routing Algorithm for Fast Global Routing," in *Proceedings of the Great Lakes Symposium on VLSI*, 2022, pp. 169-172.

[13] Z. Li, J. Zhai, Z. Li, Z. Qi, and K. Zhao, "Effective Resource Model and Cost Scheme for Maze Routing in 3D Global Routing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Singapore, Singapore, 2024, pp. 1-5.

[14] S. Lin, J. Liu, E. F. Y. Young, and M. D. F. Wong, "GAMER: GPU-Accelerated Maze Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 583-593, Feb. 2023.

[15] K.-H. Chang, H.-H. Pan, T.-C. Wang, P.-Y. Chen, and C.-F. C. Shen, "On Predicting Solution Quality of Maze Routing Using Convolutional Neural Network," in *23rd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2022, pp. 1-6.