Advancing Routing-Awareness in Analog ICs Floorplanning

Davide Basso[1,2] **Luca Bortolussi**[1] **Mirjana Videnovic-Misic**[2] **Husni Habal**[3]
[1]University of Trieste [2]Infineon Technologies AT [3]Infineon Technologies AG
davide.basso@phd.units.it,lbortolussi@units.it
{mirjana.videnovic-misic, husni.habal}@infineon.com

Abstract

The adoption of machine learning-based techniques for analog integrated circuit layout, unlike its digital counterpart, has been limited by the stringent requirements imposed by electric and problem-specific constraints, along with the interdependence of floorplanning and routing steps. In this work, we address a prevalent concern among layout engineers regarding the need for readily available routing-aware floorplanning solutions. To this extent, we develop an automatic floorplanning engine based on reinforcement learning and relational graph convolutional neural network specifically tailored to condition the floorplan generation towards more routable outcomes. A combination of increased grid resolution and precise pin information integration, along with a dynamic routing resource estimation technique, allows balancing routing and area efficiency, eventually meeting industrial standards. When analyzing the place and route effectiveness in a simulated environment, the proposed approach achieves a 13.8% reduction in dead space, a 40.6% reduction in wirelength and a 73.4% increase in routing success when compared to past learning-based state-of-the-art techniques [2].

1 Introduction

Designing the layout of analog integrated circuits (ICs) is a crucial and complex task that requires significant expertise in order to be successfully completed. The susceptibility to noise and parasitics, combined with the need of strict topological requirements, pose significant challenges for layout engineers, often necessitating multiple iterations to achieve an optimal layout that meets performance and design constraints. The entire process can be subdivided in *floorplanning* and *routing* steps, i.e. placing and connecting the analog devices, respectively. These two phases are inherently interdependent, with the optimization objectives of one stage directly affecting the feasibility and quality of the other. For instance, a compact but unroutable floorplan makes the automation efforts ineffective, highlighting the importance of considering routing metrics early in the layout stage.

Fully automated layout engines, such as MAGICAL [4] and ALIGN [7], have proved to be effective in delivering analog circuit layouts by employing analytical or metaheuristic-based floorplanning engines, that connect all the components together typically through the A* [8] algorithm. Some efforts have been made to integrate learning-based solutions into these pipelines as well, targeting both the floorplanning [15, 16] and routing phases [33, 5]. However, these promising approaches have shown limited industrial adoption, highlighting the need for further exploration in this direction.

On the other hand, the combination of relational graph convolutional neural networks (R-GCNs) [24] and reinforcement learning (RL) [27] proposed in [2] surpasses meta-heuristic approaches in exploration efficiency and solution quality, demonstrating readiness for deployment in real-world industrial pipelines. For this reason, to increase the routing-awareness of such methodology, we opt to extend its capability with the following enhancements. First, we develop a higher resolution grid to enable more compact and precise device placement, ensuring finer granularity in the layout space. Then, we

ML For Systems workshop at Neural Information Processing Systems (NeurIPS 2025).

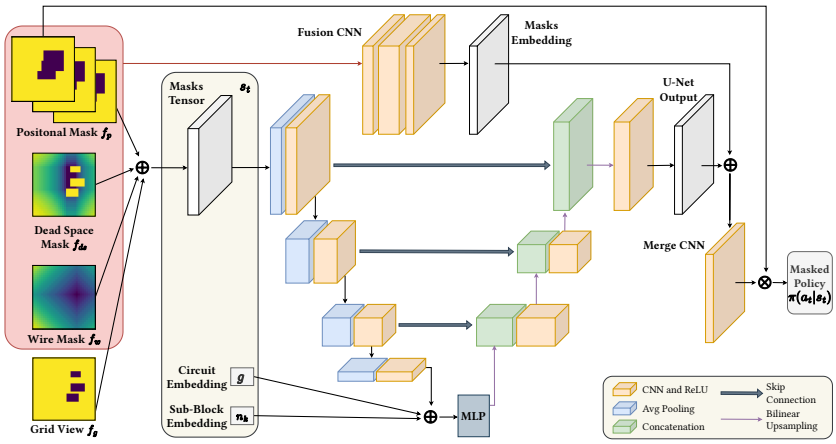

Figure 1: Overview of the RL framework, with a specific focus on the U-Net based policy network.

include detailed information from pins into both the graph representation and RL agent, accounting for more accurate half-perimeter wirelength (HPWL), computed as the half-perimeter of all nets' bounding boxes, and significantly improving the agent's understanding of the layout problem as a whole. Moreover, a dynamic routing resource estimation based on such new pin information is used to reserve enough space between floorplan elements, nicely accommodating the routing demands in a highly flexible manner. Furthermore, we introduce a revised reward scheme that actively prefers the generation of more routing-friendly floorplans by prioritizing HPWL optimization without sacrificing area efficiency, ensuring improved final outcomes. Finally, we devise a custom A* analog routing engine, inspired by [3, 31], to generate complete layouts within a simulated environment. The proposed framework outperforms past approaches both in terms of raw placement and routing-related metrics, delivering results which are consistently more routable.

2 Routing-Aware Floorplanning

Floorplanning involves placing transistor, resistor, and capacitor shapes on the IC layout canvas, with the goal of minimizing area and ensuring efficient routing. Modeled as a Markov Decision Process (MDP), RL has achieved state-of-the-art results in digital layout applications [10, 11, 29], leveraging increased capability in exploring large solution spaces compared to metaheuristic techniques.

An overview of the proposed *analog* routing-aware RL floorplanning engine is depicted in Figure 1. The state s_t captures both global and local problem information, comprising the R-GCN-generated circuit graph g and current instance n_k embeddings, along with the reward and current grid status masks. Action a_t consists in selecting one of three possible shapes for the current block b_t and placing its lower-left corner in a specific grid cell. The policy network $\pi(a_t|s_t)$ ensures valid actions by applying a mask to prevent device overlaps. A partial reward r_t is defined as the negative change in HPWL and dead space after action a_t, while the end of episode reward is defined as the negative weighted sum of floorplan area, HPWL, and the discrepancy from the target aspect ratio.

Pin-Enhanced Circuit Graph Circuit netlists can be naturally represented as a graph $G(V, E)$ [18], and for this reason we decide to employ R-GCNs to retain rich circuit and sub-block embeddings, to be then fed to the RL agent for enhancing its problem understanding and generalization capabilities [2, 1, 20, 12]. R-GCNs can, by construction, differentiate between diverse type of edges $e_i \in E$ and nodes $v_i \in V$. Electrical connections through a net or topological constraints that hold between components are embedded as edges, while nodes include functional sub-blocks, devices, pins and nets. An example of the proposed circuit graph representation is shown in Figure 3. The node feature vectors, edge relationship, and the R-GCN training procedure details are presented in Appendix A.

Policy Architecture Improvement The granularity of the floorplanning grid in [2] is increased from 32×32 to a 256×256, enabling higher precision for the analog-specific alignment and symmetry constraints. To sustain the larger action and state space, we transition from a simple convolutional neural network (CNN) [13] to a more sophisticated U-Net [23] for the RL policy architecture, similar

to the approach in *EfficientPlace* [9]. This update allows for efficient information extraction from visual grid masks, which is combined with R-GCN embeddings to reconstruct action probabilities, as in Figure 1, all while maintaining the scalability of the RL agent. Appendix B.1 details the U-Net design.

Dynamic Routing Resource Allocation To avoid routing congestion, we devise a dynamic routing resource (DRR) allocation mechanism, first proposed by [21]. The routing space $\lambda_{i,[\mathrm{h,v}]}$ is symmetrically padded around each block based on the number and direction of pins, as well as routing grid constraints. Additional details are provided in Appendix B.2.

Routing-Driven Reward Design The agent's reward at the final time step T is defined as:

$$r_T = -\left(\alpha \frac{F_{\text{area}}}{\sum_{i=1}^{m} A_i} + \beta \frac{\text{HPWL}}{\text{HPWL}_{\text{min}}} + \gamma(\text{Ar}^* - \text{Ar})^2\right), \tag{1}$$

where F_{area} is the final floorplan area, A_i denotes the area of the i^{th} device, HPWL_{min} is the minimum HPWL value obtained through metaheuristic-based simulations, and Ar^* and Ar are the target and current floorplan aspect ratios, respectively. To prioritize HPWL optimization, a linear lexicographical weighting system is introduced. Instead of solving individual minimization problems sequentially, a computationally efficient alternative assigns weights such that $\alpha_1 \gg \alpha_2 \gg \cdots \gg \alpha_m$. Following this idea, the reward weights α, β, and γ in Equation 1 are set to 10, 100, and 5, respectively. A penalty of -1000 is further applied to discourage violations of predefined constraints.

Updated Dataset and RL Training Specifics The RL agent is trained using a hybrid curriculum learning approach [30], starting with smaller circuits and periodically sampling new circuit and constraints. The training dataset includes 10 circuits based on the Infineon 130nm Commercial Technology node that vary in nature, topology, and functionality. These encompass operational transconductance amplifiers (OTAs), biasing circuits, clock resynchronization circuits, sense amplifiers with RS latches, delay elements, band-gap voltage references with low-power operational amplifiers, and level shifters. Circuit complexity varies, with 3 to 10 functional sub-blocks and 9 to 26 individual devices.

2.1 A* Routing Engine

To close the loop between floorplanning and routing and validate the results obtained with the new routing-aware RL agent in terms of routability, we develop an A*-powered rip-up and reroute routing engine, inspired by the promising methodologies employed by both state-of-the-art MAGICAL router [3] and SAGERoute [31, 32]. The prototype, despite being at early stages of its development, allows us to effectively assess the routing awareness of the proposed floorplans, avoiding DRC errors by construction, and supporting multiple metal layers routing. The insights regarding the routing grid, rule checking and routing scheme are presented in Appendix C.

3 Experimental Results

We compare the proposed routing-aware floorplanning engine with the state-of-the-art *R-GCN RL* methodology from [2], using Proximal Policy Optimization [26] to train the RL agent. We first compare raw placement performance between the proposed approach and both the vanilla and fine-tuned versions of the baseline. Then, we evaluate routing-specific metrics to assess the generation of routing-ready floorplans. The routing-specific results are presented below, while placement-related metrics are detailed in Appendix D.

Routing Metrics Assessment To evaluate routing effectiveness, we applied the A* routing engine to the floorplans generated by both the new routing-aware method and the previous R-GCN RL approach. Key metrics include total routing wirelength, routing failure rate, average iterations for a valid layout, and final via count. The A* algorithm was limited at 35 rip-up and reroute iterations. As the proposed method does not use routing engine feedback during training, both RL training and 6 additional circuits, presented in Appendix D.1, were tested to ensure comprehensive evaluation.

Table 1 presents the interquartile mean (IQM) and interquartile range (IQR) for routing metrics, calculated on successfully routed circuits. Given that the routing-aware system successfully routes all available circuits, unlike the other methods, the comparison mainly focuses on metrics derived from circuits routed by all three approaches. The best results are highlighted in bold. The routing-aware approach outperforms prior methods, achieving a 73.4% reduction in routing failure rate due to

Table 1: Routing-related metrics comparison only successfully routed floorplans. The failure rate is instead computed on all available circuits.

Algorithm	Subset	Routing-Aware RL	R-GCN RL 0-shot	R-GCN RL 1000-shot
Failure Rate (%)	All	**9.56**	79.0	83.0
Wirelength (µm)	All	6496.76 ± 448.58	8732.97 ± 881.36	5637.36 ± 731.86
	Common	**5742.17 ± 212.71**	7107.96 ± 666.75	6264.70 ± 813.04
Vias	All	144.16 ± 7.98	140.58 ± 6.86	107.75 ± 7.53
	Common	**113.48 ± 5.29**	117.51 ± 4.50	113.74 ± 7.88
Routing Iterations	All	3.94 ± 4.06	10.01 ± 2.46	3.64 ± 2.78
	Common	**3.15 ± 2.96**	10.26 ± 1.96	3.57 ± 2.83

Table 2: Performance Comparison On Delay Line Circuit.

Algorithm	Dead Space (%)	HPWL (µm)	Wirelength (µm)	Vias	Failure Rate (%)	Runtime (s)
Routing-Aware RL	62.41 ± 4.82	755.54 ± 60.91	14243.0 ± 2153.25	363.04 ± 33.25	24.0	48.7
R-GCN RL 0-shot	78.29 ± 4.10	57035.68 ± 100092.32	15943.62 ± 15546.75	286.12 ± 298.75	100.0	-
R-GCN RL 1000-shot	77.71 ± 5.13	1026.53 ± 101.0	17715.42 ± 5243.25	297.18 ± 122.25	100.0	-

precise pin information and HPWL optimization. For circuits routed by all methods, it reduces total wirelength and routing iterations by 8.34% and 11.8%, respectively, compared to the R-GCN RL 1000-shot finetuned method, and by 19.2% and 69.3%, respectively, compared to the 0-shot baseline.

Figure 2 shows the complete layout of a 24-block delay line, the largest and most complex circuit in the test set, used in analog core signal timing applications. The routing process benefits significantly from the spacing introduced by the DRR allocation procedure, which, combined with an optimized HPWL during floorplanning, ensures optimal routing paths while adhering to the defined DRC rules. As reported in Table 2, previous approaches fail to produce any fully working routing solution. In contrast, the routing-aware solution achieves successful routing in 76% of the cases in just 48.7s on average.

4 Conclusion

In this paper, we tackled the analog IC layout problem as a whole by enhancing the R-GCN RL floorplanning system with detailed pin information, HPWL-prioritized optimization and a flexible routing resource allocation strategy to reduce congestion and DRC violations. Moreover, placement precision was improved by increasing the placement grid resolution. Additionally, an A*-based routing engine prototype was developed to ensure DRC-compliant layouts suitable for import into commercial EDA tools. Experiments show the superiority of the routing-aware system, achieving significantly lower routing failure rates and more compact floorplans than previous learning-based methods. Future work will involve improving the routing engine and integrating it into a closed feedback loop with the RL placement agent, as well as performing extensive ablation studies to isolate the impact of each enhancement in the proposed R-GCN–RL floorplanning framework.

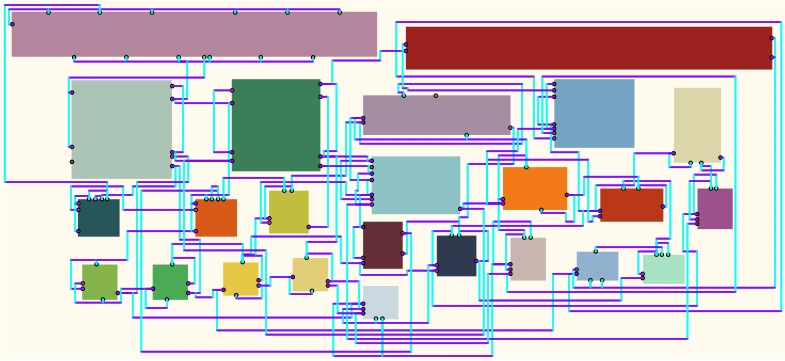

Figure 2: Complete Layout of The Delay Line for Analog Core Signal Timing Circuit.

References

[1] M. Amini, Z. Zhang, S. Penmetsa, Y. Zhang, J. Hao, and W. Liu. Generalizable Floorplanner through Corner Block List Representation and Hypergraph Embedding. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2692–2702, New York, NY, USA, Aug. 2022. ACM. ISBN 978-1-4503-9385-0. doi: 10.1145/3534678.3539220. URL https://dl.acm.org/doi/10.1145/3534678.3539220.

[2] D. Basso, L. Bortolussi, M. Videnovic-Misic, and H. Habal. Effective Analog ICs Floorplanning with Relational Graph Neural Networks and Reinforcement Learning. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pages 1–7, 2025. doi: 10.23919/DATE64628.2025.10992888. URL https://doi.org/10.23919/DATE64628.2025.10992888.

[3] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan. Toward Silicon-Proven Detailed Routing for Analog and Mixed-Signal Circuits. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 11 2020. doi: 10.1145/3400302.3415660. URL https://doi.org/10.1145/3400302.3415660.

[4] H. Chen, M. Liu, B. Xu, K. Zhu, X. Tang, S. Li, Y. Lin, N. Sun, and D. Z. Pan. MAGICAL: An Open-Source Fully Automated Analog IC Layout System from Netlist to GDSII. *IEEE Design & Test*, 38(2):19–26, 4 2021. ISSN 2168-2356, 2168-2364. doi: 10.1109/MDAT.2020.3024153. URL https://ieeexplore.ieee.org/document/9195880/.

[5] H. Chen, K.-C. Hsu, W. J. Turner, P.-H. Wei, K. Zhu, D. Z. Pan, and H. Ren. Reinforcement Learning Guided Detailed Routing for Custom Circuits. In *Proceedings of the 2023 International Symposium on Physical Design*, pages 26–34, Virtual Event, USA, 3 2023. ACM. ISBN 978-1-4503-9978-4. doi: 10.1145/3569052.3571874. URL https://dl.acm.org/doi/10.1145/3569052.3571874.

[6] J. Chen and N. R. Sturtevant. Necessary and Sufficient Conditions for Avoiding Reopenings in Best First Suboptimal Search with General Bounding Functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):3688–3696, 5 2021. doi: 10.1609/aaai.v35i5.16485. URL https://ojs.aaai.org/index.php/AAAI/article/view/16485.

[7] T. Dhar, K. Kunal, Y. Li, M. Madhusudan, J. Poojary, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, P. Mukherjee, S. Yaldiz, and S. S. Sapatnekar. ALIGN: A System for Automating Analog Layout. *IEEE Design & Test*, 38(2):8–18, 2021. doi: 10.1109/MDAT.2020.3042177. URL https://doi.org/10.1109/MDAT.2020.3042177.

[8] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan. A Systematic Literature Review of A* Pathfinding. *Procedia Computer Science*, 179:507–514, 2021. ISSN 1877-0509. doi: 10.1016/j.procs.2021.01.034. URL https://www.sciencedirect.com/science/article/pii/S1877050921000399.

[9] Z. Geng, J. Wang, Z. Liu, S. Xu, Z. Tang, M. Yuan, J. Hao, Y. Zhang, and F. Wu. Reinforcement Learning within Tree Search for Fast Macro Placement. In *Proceedings of the 41st International Conference on Machine Learning*, ICML '24. JMLR.org, 2024. URL https://openreview.net/forum?id=AJGwSx0RUV.

[10] Y. Lai, Y. Mu, and P. Luo. MaskPlace: Fast Chip Placement via Reinforced Visual Representation Learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 24019–24030, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088. URL https://doi.org/10.5555/3600270.3602014.

[11] Y. Lai, J. Liu, Z. Tang, B. Wang, J. Hao, and P. Luo. ChiPFormer: Transferable Chip Placement via Offline Decision Transformer. In *Proceedings of the 40th International Conference on Machine Learning*, pages 18346–18364. JMLR.org, 2023. URL https://doi.org/10.5555/3618408.3619165.

[12] T. P. Le, H. T. Nguyen, S. Baek, T. Kim, J. Lee, S. Kim, H. Kim, M. Jung, D. Kim, S. Lee, and D. Choi. Toward Reinforcement Learning-based Rectilinear Macro Placement Under Human Constraints. In *Fast ML for Science Workshop*, 2023. URL https://fastmachinelearning.org/iccad2023/file/fastml-iccad-23-final7.pdf.

[13] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems 2 (NIPS Conference, Denver, Colorado, USA, November 27-30, 1989)*, pages 396–404, 1989. URL http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.

[14] M. Leitner-Ankerl. A fast & densely stored hashmap and hashset based on robin-hood backward shift deletion, 8 2025. URL https://github.com/martinus/unordered_dense.

[15] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, W. Xu, S. S. Sapatnekar, R. Harjani, and J. Hu. A Customized Graph Neural Network Model for Guiding Analog IC Placement. In *Proceedings of the 39th International Conference on Computer-Aided Design*, volume 2020-November, pages 1–9, New York, NY, USA, 11 2020. ACM. ISBN 978-1-4503-8026-3. doi: 10.1145/3400302.3415624. URL https://dl.acm.org/doi/10.1145/3400302.3415624.

[16] M. Liu, K. Zhu, J. Gu, L. Shen, X. Tang, N. Sun, and D. Z. Pan. Towards Decrypting the Art of Analog Layout: Placement Quality Prediction via Transfer Learning. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 496–501, 3 2020. doi: 10.23919/DATE48585.2020.9116330. URL https://ieeexplore.ieee.org/document/9116330/?arnumber=9116330.

[17] N. Lohmann. Json for modern c++, 4 2025. URL https://github.com/nlohmann.

[18] D. S. Lopera, L. Servadei, G. N. Kiprit, S. Hazra, R. Wille, and W. Ecker. A Survey of Graph Neural Networks for Electronic Design Automation. In *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6, Raleigh, NC, USA, 8 2021. IEEE. ISBN 978-1-66543-166-8. doi: 10.1109/MLCAD52597.2021.9531070. URL https://ieeexplore.ieee.org/document/9531070/.

[19] L. McMurchie and C. Ebeling. Pathfinder: A Negotiation-based Performance-Driven Router for FPGAs. In *Proceedings of the 1995 ACM Third International Symposium on Field-Programmable Gate Arrays*, FPGA '95, pages 111–117, New York, NY, USA, 1995. Association for Computing Machinery. doi: 10.1145/201310.201328. URL https://doi.org/10.1145/201310.201328.

[20] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nova, et al. A graph placement methodology for fast chip design. *Nature*, 594 (7862):207–212, 2021. doi: 10.1038/s41586-021-03544-w. URL https://doi.org/10.1038/s41586-021-03544-w.

[21] H.-C. Ou, H.-C. C. Chien, and Y.-W. Chang. Simultaneous Analog Placement and Routing with Current Flow and Current Density Considerations. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 5:1–5:6, New York, NY, USA, 2013. Association for Computing Machinery. doi: 10.1145/2463209.2488739. URL https://doi.org/10.1145/2463209.2488739.

[22] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[23] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. URL https://doi.org/10.1007/978-3-319-24574-4.

[24] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling Relational Data with Graph Convolutional Networks. In A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, editors, *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2018. doi: 10.1007/978-3-319-93417-4_38. URL https://doi.org/10.1007/978-3-319-93417-4_38.

[25] B. Schling. *The Boost C++ Libraries*. XML Press, 2011. ISBN 0982219199. URL https://theboostcpplibraries.com/.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *CoRR*, 2017. URL http://arxiv.org/abs/1707.06347.

[27] R. S. Sutton and A. Barto. *Reinforcement learning: an introduction*. The MIT Press, 2020. URL http://www.incompleteideas.net/book/the-book-2nd.html.

[28] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *CoRR*, 2020. URL https://arxiv.org/abs/1909.01315.

[29] B. Yang, Q. Xu, H. Geng, S. Chen, and Y. Kang. Miracle: Multi-Action Reinforcement Learning-Based Chip Floorplanning Reasoner. In *Proceedings of the 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2024. doi: 10.23919/DATE58400.2024.10546767. URL https://doi.org/10.23919/DATE58400.2024.10546767.

[30] W. Zaremba and I. Sutskever. Learning to Execute. *CoRR*, abs/1410.4615, 2014. doi: 10.48550/arXiv.1410.4615. URL http://arxiv.org/abs/1410.4615.

[31] H. Zhang, X. Gao, H. Luo, J. Song, X. Tang, J. Liu, Y. Lin, R. Wang, and R. Huang. SAGERoute: Synergistic Analog Routing Considering Geometric and Electrical Constraints with Manual Design Compatibility. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, Antwerp, Belgium, 4 2023. IEEE. doi: 10.23919/DATE56975.2023.10137296. URL https://ieeexplore.ieee.org/document/10137296/.

[32] H. Zhang, X. Gao, Z. Shen, J. Song, X. Cheng, X. Tang, Y. Lin, R. Wang, and R. Huang. SAGERoute 2.0: Hierarchical Analog and Mixed Signal Routing Considering Versatile Routing Scenarios. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 3 2024. doi: 10.23919/DATE58400.2024.10546542. URL https://ieeexplore.ieee.org/document/10546542/?arnumber=10546542.

[33] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan. GeniusRoute: A New Analog Routing Paradigm Using Generative Neural Network Guidance. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019. doi: 10.1109/ICCAD45719.2019.8942164. URL https://doi.org/10.1109/ICCAD45719.2019.8942164.

A R-GCN Insights

A.1 Circuit Graph Details

The feature vector of a device node in the circuit graph, $x_d \in X$, includes the block area, width, height, the number of pins, and a 28-dimensional one-hot encoding representing the block's functional structure. Similarly, the feature vector of a pin node, $x_p \in X$, is characterized by a series of numerical encodings of pin's direction, the type of net it is connected to (signal, ground, or power), and a one-hot encoding of both the bottom and top metal layers that the pin can accept, which varies according to the technology specifications. Finally, the feature vector of a net node, $x_n \in X$, is constructed by concatenating the number of connected devices and pins.

Connections between nodes are established based on the relationships among them. By construction, device nodes and pin nodes are connected through a *device has pin* relationship, while pin nodes are connected to net nodes via the *pin belongs to net* relation. This approach results in a near-tripartite graph structure. We refer to it as "near-tripartite" because, in addition to the device-pin-net relationships, we need to consider also the constraints between devices that are also represented as edges.

A.2 R-GCN Specifics

The R-GCN architecture consists of two R-GCN layers followed by a node mean aggregation block to generate the graph embedding, which is subsequently passed through two fully connected layers to predict the reward value.

The training dataset includes 91200 floorplans with corresponding reward values, optimized for area and proxy wirelength metrics using metaheuristic algorithms such as simulated annealing, genetic algorithms, and particle swarm optimization. Circuit sizes range from 3 up to 24 blocks and span a variety of designs, including OTAs, bias circuits, drivers, level shifters, clock synchronizers, comparators, and oscillators. To ensure robustness, the dataset is balanced between constrained and unconstrained floorplans.

The supervised learning task minimizes the mean squared error between the predicted and ground truth rewards for input circuit graphs. Training was performed on a single Nvidia A30 GPU, with a total runtime of 46 minutes. Additional R-GCN architecture and training hyperparameters are detailed in Table 3.

Table 3: R-GCN hyperparameters.

R-GCN Model	
R-GCN units	64
Learning rate	4.7e-3
Dropout	0.1
Optimizer	ADAM
Weight Decay	5.6e-5
β_1, β_2	0.9, 0.999
FC Neural Network	
Input normalization	tanh
Activation	ReLU
Hidden layer neurons	[64,32,1]
Loss	MSE

B Enhancing RL Capabilities for Better Routability

B.1 U-Net Policy Design

Originally designed for medical images segmentation, the U-Net is perfectly suited for efficiently processing visual information coming from the visual masks that are part of the agent's state, f_g, f_w, f_{ds}, f_p. In fact, thanks to its architecture design, which is detailed hereafter, it is possible to

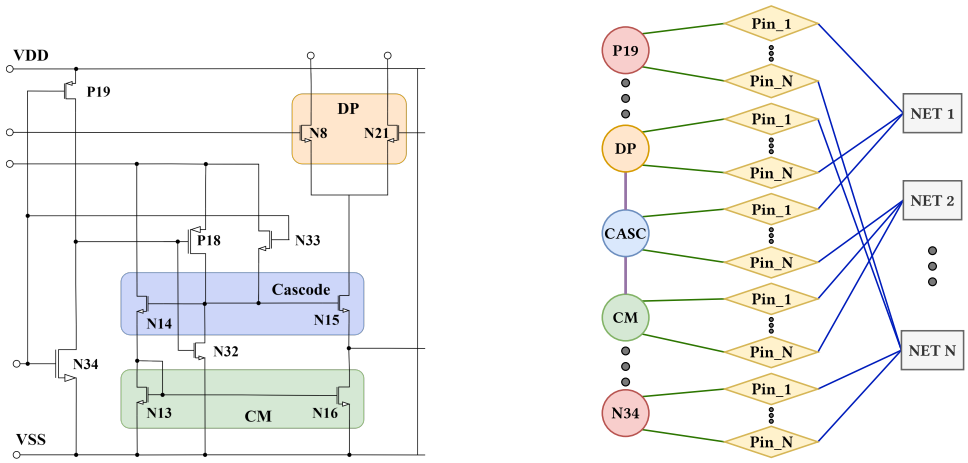

Figure 3: An 8-block OTA schematic (a) and the near-tripartite circuit graph (b). Violet edges represent the topological constraints applied between recognized structures, green ones denote the *device has pin* relationship, and blue edges connect pins and nets through the *pin belongs to net* relationship.

process high-dimensional input data without incurring in a significant computational overhead. As a result, despite the increased grid size and the large action space of $|\mathcal{A}| = 256 \times 256 \times 3 = 196608$, employing the U-Net enables effective training and inference.

The U-Net architecture adopts a U-shaped configuration, which can be conceptually divided into three main components: the encoding (or contracting) path, the bottleneck, and the decoding (or expansive) path.

The *contracting path*, located on the left side of the U-Net, serves as a feature extractor for the input image-shaped data while progressively reducing its spatial dimensions. This section typically consists of a sequence of convolutional layers, followed by ReLU activation functions and max-pooling operations.

The *bottleneck*, which forms the central portion of the U-Net, represents the most compact and low-dimensional encoding of the input data. In our setup, this bottleneck incorporates an MLP to efficiently combine not only the information extracted from the contracting path, but also the R-GCN generated circuit g, and single block n_k embeddings. Doing so, both global circuit and local block-specific details are integrated into the distilled encoding.

Eventually, the *expansive path*, i.e. the right side of the U-Net, reconstructs the spatial dimensions of the input data through a series of upsampling and convolutional layers. *Skip connections* are used to directly connect layers from the encoding path to their corresponding layers in the decoding section. These help to retain fine-grained spatial details lost during downsampling, enabling the network to reconstruct the spatial structure of the input with high precision.

In order to better guide the RL agent's decision-making process, as the final step of our policy, the state masks are fed to a CNN to eventually retrieve a learned feature vector of such combination. The output of this CNN is then integrated with the probability distribution produced by the U-Net architecture, using an additional CNN, to produce the final action probabilities for the RL agent. The value network instead follows the same rationale of the contracting path of the policy network, ultimately outputting the learned value function values through an MLP.

The hyperparameters used for the policy network from Figure 1 are listed in Table 4.

Table 4: RL Policy Model Hyperparameters.

Model Part	Layer ID	Kernel Size	N. Channels	Stride	Padding	Output Size
U-Net Encoder	E_CNN_1	3×3	8	1	1	(8, 128, 128)
	E_CNN_2	3×3	16	1	1	(16, 64, 64)
	E_CNN_3	3×3	32	1	1	(32, 16, 16)
	E_CNN_4	3×3	32	1	1	(32, 4, 4)
	Max_Pool_2	2×2	-	-	-	-
	Max_Pool_4	4×4	-	-	-	-
Bottleneck	FC_1	-	-	-	-	(640, 512)
	FC_2	-	-	-	-	(512, 512)
U-Net Decoder	D_CNN_1	3×3	16	1	1	(16, 16, 16)
	D_CNN_2	3×3	8	1	1	(8, 64, 64)
	D_CNN_3	1×1	3	1	0	(3, 256, 256)
	Bilinear_Up_2	-	-	-	-	(in $\times$ 2, in $\times$ 2)
	Bilinear_Up_4	-	-	-	-	(in $\times$ 4, in $\times$ 4)
Fusion CNN	F_CNN_1	1×1	8	1	0	(8, 256, 256)
	F_CNN_2	1×1	8	1	0	(8, 256, 256)
	F_CNN_3	1×1	3	1	0	(3, 256, 256)
Merge CNN	M_CNN_1	1×1	3	1	0	(3, 256, 256)

B.2 Dynamic Routing Resources Allocation Details

The routing space, $\lambda_{i,[\mathrm{h,v}]}$ required on either the horizontal or vertical sides of a block i, is defined as follows:

$$\lambda_{i,[\mathrm{h,v}]} = \zeta + |p_{\mathrm{i},[\mathrm{h,v}]}| \cdot \left(\max_{j \in Nets(i)} (\mathcal{W}_{j,\mathrm{spacing}}) + \max_{j \in Nets(i)} (\mathcal{W}_{j,\mathrm{width}}) \right), \tag{2}$$

where ζ is equal to the minimum chip canvas placement grid step, $|p_{\mathrm{i},[\mathrm{h,v}]}|$ is the number of vertical or horizontal pins of sub-block i, $\max_{j \in Nets(i)}(\mathcal{W}_{j,\mathrm{spacing}})$ is the maximum parallel run spacing among the nets connected to block i, $Nets(i)$, and $\max_{j \in Nets(i)}(\mathcal{W}_{j,\mathrm{width}})$ is the maximum wire width for each connection.

Since the output direction of the pins during placement is unknown, the routing space defined in Equation 2 must be padded symmetrically. More precisely, horizontal pins require padding on both the left and right sides, while vertical pins on the top and bottom ones. This approach inevitably results in an overestimation of the actual space needed to connect all components. However, this added room ensures very low routing failure rates, while still achieving well-optimized floorplans in terms of area utilization and HPWL.

C Analog-Compliant A* Routing Engine

C.1 Routing Grid Graph

The routing grid is modeled as a 3-D graph, $G_r(V_r, E_r)$, representing the entire routing space. V_r denotes the set of possible interconnection points, i.e. vertices, and E_r the set of edges. The edges $(u, v) \in E_r$ can either be horizontal or vertical, depending on the connection type. In fact, horizontal edges correspond to routing track segments within the same metal layer, while vertical ones represent connections between different metal layers, realized physically as vias.

In our setup, the grid step used to interleave two routing vertices is determined by the minimum valid spacing between two physical components specified by the circuit technology. To efficiently represent G_r, since the number of vertices is typically large, an adjacency list is implemented using a hashmap structure, enabling scalable storage and fast lookup of the graph's components. To further limit the increase in size of G_r we limit the possible metal layers to two options, one for vertical and the other for horizontal tracks.

C.2 DRC Rules and Constraints

Analog ICs routing is characterized by a multitude of rules that must be satisfied to meet industrial requirements and practical usability of the final layout. For this reason, we guarantee the compliance to the following fundamental DRC rules:

- *Minimum wire length*: Specifies the minimum length for a wire segment in G_r.
- *Minimum wire area*: Specifies the minimum area for a wire segment in G_r.
- *End of line spacing*: Defines the minimum required spacing, *eolSpace*, from any other wire or via at the end of a wire segment within the *eolWithin* distance beyond the end of the line. This rule applies only if a parallel wire segment exists within the parallel edge region on the side of the wire segment of interest, defined by the *parWidth* and *parSpace* values.
- *Parallel wire spacing*: Specifies the minimum parallel spacing from any other wire or via based on the parallel run length and the maximum width of the two wires. In our scenario, we set it to a minimum fixed valid value.

Obstacle avoidance is also considered in order to ensure that wires do not overlap with pins that are not part of the net of interest or intersect with devices. Moreover, given that analog circuit routing typically involves highly regular shapes to form interconnections between devices, a low-bending rule is introduced to discourage too tortuous paths, which would eventually lead to an increased number of vias and the possibility to incur in DRC violations.

The wire width is set to the minimum valid value to ensure compliance with DRC rules. However, the system is designed to accommodate varying wire widths for different metal layers, allowing flexibility based on specific routing requirements or technology constraints.

C.3 The Routing Scheme

The routing process is based on a negotiated rip-up and reroute scheme [19]. Whenever a net violates one of the aforementioned rules or constraints, a history cost associated to all the vertices that have been used to construct the failing wires is accredited before they are removed. Doing so, the A* engine can avoid to visit in successive routing iterations the same nodes that led to some source of errors in the past, incentivizing a different exploration of the grid space.

The net routing order is defined according to an heuristic that allows us to properly rank the nets based on their criticality and size. First of all, nets with more routing failures are prioritized. Then, among nets with the same number of routing failures, nets with higher HPWL are prioritized. If both routing failures and HPWL are identical, the net with more pins is favored. Doing so, not only the final quality of the layout is improved, but also the number of iterations to generate a valid routing.

The output direction of the pins to be connected through a net has to be defined at the very beginning of the routing stage. In fact, pins are typically positioned within a device layout and they need to be projected to one side of the block, either horizontally or vertically, depending on pin's characteristics. Since HPWL is an optimal estimator of the true circuit wirelength, the approach we follow is to select the block side for each pin that leads to the smallest increase in the HPWL value of the net to be routed. This is achieved by evaluating all the possible combinations of pin-to-block side assignments for the net of interest. While this naive approach increases slightly the computational effort and congestion, it keeps a strong correlation with the smallest estimated wirelength, which in the end is the key goal for an high quality routing solution. In such cases, the earlier padding overestimation around the blocks proves to be highly beneficial.

Once all the endpoints' positions are defined, we decompose each net into a two-pin one by using a MST decomposition.

The A* search follows a bidirectional scheme by alternating a forward and backward search, starting from the source and target pin respectively, in order to improve the speed efficiency of navigating into such a large grid space. Design rules and constraints are integrated into the node cost function $g(n)$ in a shallow manner. If n is a neighboring node of the current node k, the cost function can be expressed as:

$$D(n,k) = |x_n - x_k| + |y_n - y_k| + c_{\text{VIA}} \cdot |z_n - z_k| \tag{3}$$

$$g(n) = g(k) + D(n,k) + c_{\text{DRC}} \cdot \mathbb{1}(\text{violation}) + H(n), \tag{4}$$

where $D(n,k)$ is a Manhattan distance metric that accounts also for a via cost, c_{VIA}, to discourage excessive layer changes and jagged paths. Instead, $c_{\mathrm{DRC}} \cdot \mathbb{1}(\mathrm{violation})$ represents a penalty for DRC violations when visiting the neighboring node, while $H(n)$ is the accumulated history cost of n up to the current reroute iteration. Overall, this formulation ensures that the cost function penalizes undesirable routing behaviors while also promoting efficient and compliant routing paths. On the other hand, obstacles are directly removed from the routing grid, simplifying the routing process and ensuring that invalid paths are avoided by construction.

The heuristic $h(n)$ used to estimate the cost from the neighboring node to the target goal t is computed as $D(t,n)$. In order to gain some speedup in runtime we decided to introduce a *dynamic weighting* mechanism for the cost function as follows:

$$f(n) = \begin{cases} g(n) + h(n) & \text{if } g(n) < h(n), \\ \dfrac{g(n) + (2\kappa - 1) \cdot h(n)}{\kappa} & \text{otherwise.} \end{cases} \tag{5}$$

This idea is inspired by the work of Chen et al. in [6], which demonstrates how increasing the weight of the heuristic depending on κ as the routing path progresses can be used to exploit the proximity to the goal. In our implementation, we set $\kappa = 3$. Doing so, the algorithm prioritizes reaching the target rather than exploring the search space, which on the contrary is preferred during the early stages of the search.

Finally, a *tie-breaking* mechanism is introduced to prevent the A* router from exploring all paths of the same length instead of focusing on a single optimal path. This is achieved by multiplying $h(n)$ by $1 + q$, with $q = 0.0001$, which biases the algorithm toward expanding vertices closer to the goal and further reduces unnecessary exploration of the search grid. This approach, combined with the dynamic weighting from before, ensures reasonably fast execution runtimes for the whole pipeline when used both in testing and production environments.

D Raw Placement Metrics Assessment

The RL agent uses stable-baselines3 [22] and DGL [28] libraries. Due to the increased size of the training dataset, the training process requires slightly more time, taking approximately 3 days and 20 hours on a single Nvidia A30 GPU.

The A* routing engine, on the other hand, is implemented in C++, in favor of guaranteeing short runtimes and high computational efficiency. The ankerl [14] library is used for hashmap operations, the Boost [25] library for implementing an efficient R-tree spatial index, and the nlohmann [17] library for processing JSON input files.

D.1 Floorplanning Metrics Assessment

Performances are measured on a test set of 6 different designs. These include both circuits of similar nature to those in the training set, typically with greater complexity, and entirely different circuit types, such as comparators and oscillators. The number of functional blocks in the test circuits varies from 3 to 24, while the total number of individual devices ranges between 5 and 40, ensuring enough variety for validating our approach. No constraints are imposed on any circuit to ensure a fair comparison.

Table 5: Comparison of the Routing-Aware Approach Versus R-GCN RL

Algorithm	Finetune Shots	Runtime (s)	Dead Space (%)	HPWL (µm)	Reward
Routing-Aware	0	0.52 ± 0.29	*57.44 ± 2.68*	*797.29 ± 46.51*	*-2.61 ± 0.13*
Routing-Aware w/o DRR	0	0.49 ± 0.1	**37.28 ± 7.72**	**673.36 ± 27.37**	**-1.65 ± 0.23**
R-GCN RL	0	**0.32 ± 0.12**	59.78 ± 11.42	1006.75 ± 167.65	-7.87 ± 19.53
R-GCN RL	1000	788.82 ± 1505.63	*55.98 ± 9.06*	*749.92 ± 74.77*	*-1.75 ± 0.46*

Table 5 showcases the IQM and IQR values obtained on the test set circuits for our new floorplanning methodology runtime, resulting floorplan dead space, HPWL and corresponding reward. We also report the results obtained using the routing-aware RL system without DRR, in order to clearly showcase the enhancement provided by the increased grid size and the capability that our new

proposed solution can achieve. The best results are presented in bold, while the second and third best rewards are highlighted in italic underlined and underlined only, respectively.

As expected, the increased grid size delivers the best results across all floorplan quality metrics within less than half a second. More specifically, compared to the updated RL agent without DRR, the R-GCN RL 1000-shot approach shows increases in dead space and HPWL of 50.2% and 11.4%, respectively. For the R-GCN RL 0-shot case, these increases are even more evident, reaching 60.4% and 49.5% for the same metrics.

In contrast, the routing-aware system achieves results that nearly match the R-GCN RL 1000-shot produced floorplans, with dead space and HPWL larger by 2.6% and 6.3%, respectively. Nevertheless, the significant runtime improvement, a gain of 150769%, makes this new solution our preferred choice. Moreover, the lower IQR values for every routing-aware solution suggest much more stable performance compared to both previous approaches, making it particularly well-suited for generating fast and accurate floorplan drafts.