Simulated Annealing for VLSI Design

Shrivatsa Gururaj Kulkarni

This manuscript was compiled on August 29, 2025

Abstract

This project applies simulated annealing to the VLSI chip floor-planning problem, aiming to optimally place rectangular modules within a fixed boundary. The cost function combines Manhattan wire length and an overlap penalty. By refining placements step by step, while probabilistically tolerating cost-increasing moves, the algorithm reduces total wire length and eliminates overlaps. Tested on ten modules with fourteen nets, results show improved layouts and convergence visualizations confirm its effectiveness. This work demonstrates the value of metaheuristics in electronic design automation and provides a basis for future hybrid approaches.

Keywords: *chip floor planning, simulated annealing, wire length optimization, metaheuristics, electronic design automation*

1. Introduction

Very-Large-Scale Integration (VLSI)) refers to the process of integrating thousands to millions of transistors onto a single silicon chip, enabling the creation of highly complex and powerful integrated circuits (ICs) **Weste**. VLSI has been the driving force behind modern computing and communication systems, forming the foundation of microprocessors, memory devices, and application-specific integrated circuits (ASICs). With continuous advancements in semiconductor technology, the demand for faster, smaller, and more energy-efficient devices has grown, making efficient VLSI design methodologies increasingly critical **Sangiovanni**.

A fundamental step in the VLSI design flow is **chip floor-planning**, which determines the spatial arrangement of modules on the chip die. An optimized floor-plan ensures minimal wiring complexity, reduced signal delay, balanced area utilization, and improved thermal and power distribution **Sherwani**. Conversely, poor floor-planning can lead to excessive wire length, routing congestion, timing violations, and suboptimal chip performance. Due to the exponential growth of possible placements as the number of modules increases, floor-planning is categorized as an **NP-hard combinatorial optimization problem WongSA**. This complexity has driven the adoption of heuristic and metaheuristic approaches for practical solutions.

Among the various metaheuristics, **simulated annealing (SA)** has proven particularly effective in VLSI design **Kirkpatrick**. SA is inspired by the physical process of annealing in metallurgy, where a material is slowly cooled to achieve a low-energy crystalline state. The algorithm mirrors this process by probabilistically accepting not only better solutions but also, with decreasing probability, worse solutions. This property allows the search to escape local minima and approximate a global optimum. In the context of floor-planning, the cost function typically combines the *total wire length* of interconnections and *overlap penalties* between modules. By iteratively refining placements while controlling the acceptance of worse solutions through a cooling schedule, SA converges toward high-quality, non-overlapping layouts.

Therefore, simulated annealing provides a robust and practical framework for addressing the challenges of chip floor-planning in VLSI design, striking a balance between computational feasibility and solution quality.

2. Problem Formulation

The **chip floor-planning problem** can be formally described as arranging a set of rectangular modules within a fixed chip boundary while minimizing interconnection cost and avoiding overlaps **Sherwani**. Each module is defined by its width and height, and its placement is specified by the coordinates of its bottom-left corner on the chip. The design must satisfy geometric and functional constraints while optimizing for layout quality.

2.1. Modules and Chip Boundary

Let

$$M = \{m_1, m_2, \dots, m_n\}$$

be the set of n modules, where each module m_i has dimensions (w_i, h_i). The chip is modeled as a rectangular die of dimensions $W \times H$. A valid placement assigns each module to a position (x_i, y_i) such that:

$$0 \le x_i \le W - w_i, \quad 0 \le y_i \le H - h_i$$

ensuring all modules remain inside chip boundaries.

2.2. Nets and Interconnections

The communication requirements between modules are represented as a set of **nets**:

$$N = \{(m_i, m_j) \mid m_i, m_j \in M\}$$

Each net defines a direct electrical connection between two modules. The cost of a net is measured by the **Manhattan distance** between the centers of the connected modules:

$$d(m_i, m_j) = |(x_i + w_i/2) - (x_j + w_j/2)| + |(y_i + h_i/2) - (y_j + h_j/2)|$$

The **total wire length** is then the sum of all net distances:

$$WL = \sum_{(m_i, m_j) \in N} d(m_i, m_j)$$

2.3. Overlap Constraint

Modules must not overlap, as overlaps represent physically infeasible designs. To enforce this, an **overlap penalty** is added to the cost function whenever two modules intersect. A simple penalty model is:

$$P = \lambda \cdot \sum_{(m_i, m_j) \in M} O(m_i, m_j)$$

where $O(m_i, m_j)$ is 1 if modules m_i and m_j overlap and 0 otherwise, and λ is a penalty coefficient.

2.4. Objective Function

The final objective function to be minimized is:

$$\text{Cost} = WL + P$$

where WL is the total wire length and P is the overlap penalty.

This formulation transforms chip floor-planning into a **combinatorial optimization problem**, where the search space includes all possible placements of modules, and the task is to identify a layout with minimal cost.

3. Methodology: Simulated Annealing

3.1. Overview

Simulated Annealing (SA) is a probabilistic metaheuristic inspired by the annealing process in metallurgy **Kirkpatrick**. In physical annealing, a material is heated to a high temperature and then cooled gradually, allowing atoms to settle into a low-energy crystalline state. Similarly, SA explores the solution space of an optimization problem by probabilistically accepting both improving and, with a decreasing likelihood, deteriorating solutions. This prevents the algorithm from becoming trapped in local minima and allows it to approximate a global optimum.

3.2. State Representation

In the context of chip floor-planning, a **state** represents a complete placement of all modules within the chip boundary. Each module m_i is associated with a position (x_i, y_i) corresponding to the coordinates of its bottom-left corner. The state can be represented as:

$$S = \{(m_i, x_i, y_i) \mid m_i \in M\}$$

where M is the set of all modules.

3.3. Cost Function

The cost function evaluates the quality of a placement and guides the optimization process. It is defined as:

$$\text{Cost} = WL + P$$

where WL is the total wire length (Equation 1) and P is the overlap penalty (Equation 2). A lower cost corresponds to a more efficient and feasible floor-plan.

3.4. Neighborhood Function

To explore new solutions, SA uses a **neighborhood function**. A neighbor state is generated by selecting a random module and moving it to a new valid position within the chip boundary. This introduces diversity in the search process and enables gradual refinement of placements.

3.5. Acceptance Criterion

The acceptance of a new state S_{new} depends on the Metropolis criterion:

$$P(\text{accept}) = \begin{cases} 1 & \text{if } \Delta E \leq 0 \\ \exp\left(-\frac{\Delta E}{T}\right) & \text{if } \Delta E > 0 \end{cases}$$

where $\Delta E = \text{Cost}(S_{\text{new}}) - \text{Cost}(S_{\text{old}})$ and T is the current temperature. This criterion ensures that better solutions are always accepted, while worse solutions may be accepted with a probability that decreases as the temperature cools.

3.6. Cooling Schedule

The **temperature** T is gradually reduced according to a cooling schedule:

$$T_{k+1} = \alpha \cdot T_k$$

where α is the cooling coefficient ($0 < \alpha < 1$). A slower cooling rate (higher α) increases exploration, while a faster cooling rate (lower α) accelerates convergence but may lead to suboptimal solutions.

3.7. Algorithm Flow

The overall simulated annealing algorithm for VLSI floor-planning can be summarized as follows:

1. Initialize with a random placement S and compute its cost.
2. Set initial temperature $T = T_{\text{start}}$.
3. For each iteration:
   (a) Generate a neighbor state S_{new} by random movement of a module.
   (b) Compute the cost of S_{new}.
   (c) Accept or reject S_{new} based on the Metropolis criterion.
   (d) Update the best solution if an improved cost is found.
   (e) Reduce temperature using the cooling schedule.
4. Terminate after maximum iterations or when temperature reaches a threshold.

This methodology allows simulated annealing to effectively search the solution space, balancing exploration and exploitation to obtain a high-quality chip layout.

4. Implementation and Results

4.1. Implementation Setup

The simulated annealing algorithm for VLSI floor-planning was implemented in Python. The implementation makes use of the following key components:

- **Programming Language:** Python 3.13
- **Libraries:** `random` for stochastic moves, `math` for exponential probability functions, and `matplotlib` for visualization of cost convergence and final placements.
- **Modules:** Each module was represented as a rectangle defined by its width and height.
- **Nets:** Nets were modeled as pairs of modules with Manhattan distance used to measure interconnection cost.

4.2. Experimental Parameters

The following parameters were used in the simulated annealing experiments:

- Initial Temperature $T_{\text{start}} = 100$
- Cooling Coefficient $\alpha = 0.995$
- Maximum Iterations $= 50000$
- Overlap Penalty $\lambda = 10$
- Chip Boundary 20×20

These parameters were chosen empirically to balance exploration of the search space with convergence efficiency.

4.3. Cost Convergence

Figure 1 illustrates the cost history across iterations. The cost decreases rapidly in the early stages due to acceptance of beneficial moves, followed by gradual refinement as the temperature cools. This trend confirms that the simulated annealing algorithm effectively avoids poor local minima and converges toward a high-quality solution.

4.4. Final Placement

The final optimized placement obtained from the algorithm is shown in Figure 2. Each rectangle represents a module, annotated with its identifier. The placement is non-overlapping and achieves a low total cost while respecting chip boundaries. Nets connecting modules are minimized in length, which improves wirelength efficiency.

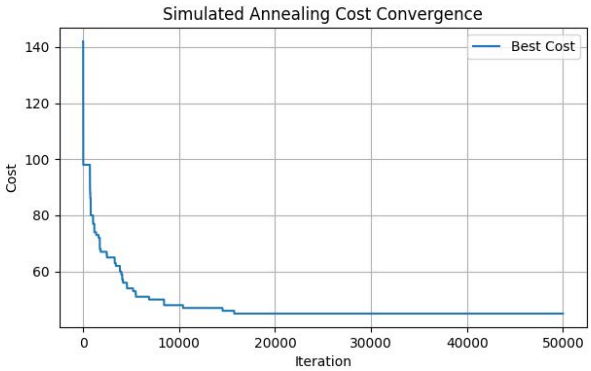

Figure 1. Cost convergence of simulated annealing across iterations.

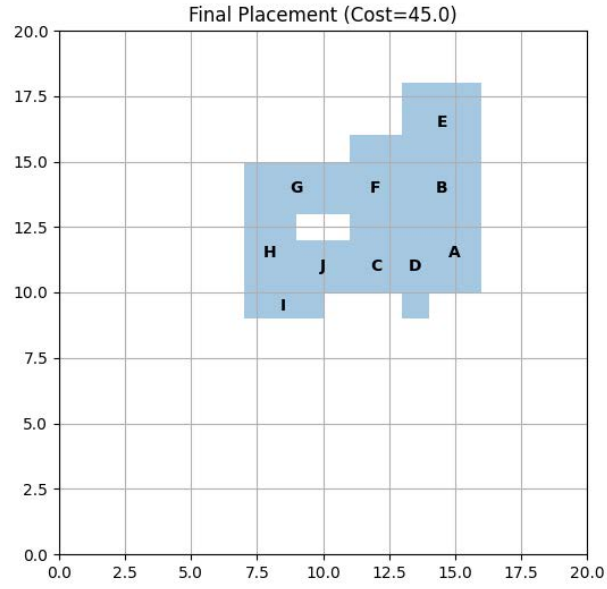

Figure 2. Final optimized placement of modules on the chip.

4.5. Discussion of Results

The experimental results demonstrate that simulated annealing provides an effective approach to solving the chip floor-planning problem. The cost function, which incorporates both wirelength and overlap penalties, guides the optimization process toward feasible, high-quality layouts.

Notably:

- The algorithm successfully eliminates module overlaps by penalizing infeasible placements.
- The final wire length is significantly reduced compared to random initialization, leading to improved routing efficiency.
- The cooling schedule plays a critical role in determining the balance between exploration and exploitation.

Overall, the results validate simulated annealing as a practical heuristic for tackling the NP-hard floor-planning problem in VLSI design.

5. Conclusion and Future Work

This project explored the application of **Simulated Annealing (SA)** to the VLSI floor-planning problem. By modeling modules as rectangles and defining a cost function that combines total wire length and overlap penalties, the algorithm was able to generate non-overlapping, low-cost placements within the chip boundary. The results confirmed that SA is effective in avoiding local minima and converging to high-quality layouts, validating its role as a robust metaheuristic for combinatorial optimization in VLSI design.

However, several opportunities for improvement exist. Future work may focus on:

- Developing more advanced cost functions that incorporate additional design factors such as thermal constraints, timing analysis, and power distribution.
- Exploring alternative neighborhood functions such as module swapping, rotation, or multi-module moves to improve solution diversity.
- Investigating hybrid optimization approaches that combine SA with other heuristics such as Genetic Algorithms or Tabu Search for faster convergence and better solution quality.
- Scaling the implementation to larger and more complex designs, closer to real-world industrial applications.

6. Applications

The techniques and methodologies used in this project extend beyond academic exploration and have significant relevance in practical domains:

- **ASIC and SoC Design:** Floor-planning is a crucial step in Application-Specific Integrated Circuits (ASICs) and System-on-Chip (SoC) design, directly impacting performance, power, and area.
- **FPGA Placement:** Similar optimization problems arise in Field Programmable Gate Arrays (FPGAs), where simulated annealing has historically been used for placement and routing.
- **General Optimization:** Beyond VLSI, simulated annealing can solve diverse optimization problems such as scheduling, resource allocation, and network design.
- **Electronic Design Automation (EDA) Tools:** Modern EDA tools incorporate heuristic-based optimization techniques like SA, making this project directly aligned with industry practices.

7. Limitations

While simulated annealing proved effective in producing feasible and optimized floor-plans, the current implementation has several limitations that restrict its scalability and industrial applicability:

- **Simplified Cost Function:** The cost function considers only wirelength and module overlaps. In practice, VLSI floor-planning must also account for timing, power consumption, clock distribution, and thermal constraints.
- **Scalability Issues:** The algorithm performs adequately for small to medium-scale problems. However, as the number of modules grows, the search space expands exponentially, leading to longer runtime and potentially less optimal solutions.
- **Basic Move Set:** The neighborhood function only involves relocating one module at a time. More advanced moves such as rotation, swapping, or hierarchical decomposition would enhance exploration of the solution space.
- **Cooling Schedule Sensitivity:** The quality of results is highly dependent on the choice of initial temperature, cooling coefficient, and number of iterations. Poor parameter tuning may lead to premature convergence or excessive runtime.
- **Lack of Industrial Benchmarks:** The experimental setup used artificial module sizes and nets. Validation against industry-standard benchmarks such as MCNC or GSRC suites would provide a stronger assessment of performance.

These limitations suggest that while simulated annealing is a valuable pedagogical and research tool, further refinement and integration with advanced optimization techniques are necessary for deployment in industrial electronic design automation (EDA) workflows.

References

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983. Available: https://www2.stat.duke.edu/~scs/Courses/Stat376/Papers/TemperAnneal/KirkpatrickAnnealScience1983.pdf

[2] J. Chen, W. Zhu, and M. M. Ali, “A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 41, no. 4, pp. 544–553, Jul. 2011. Available: https://www.researchgate.net/profile/Montaz-Ali/publication/224174253_A_Hybrid_Simulated_Annealing_Algorithm_for_Nonslicing_VLSI_Floorplanning/links/554218860cf21b214375a699/A-Hybrid-Simulated-Annealing-Algorithm-for-Nonslicing-VLSI-Floorplanning.pdf

[3] T. C. Chen and Y. W. Chang, “Modern Floorplanning Based on Fast Simulated Annealing,” in *Proc. Int. Symp. on Physical Design (ISPD)*, pp. 104–112, 2005. Available: https://cc.ee.ntu.edu.tw/~ywchang/Papers/ispd05-floorplanning.pdf

[4] K. Rajesh and R. Kumar, “Simulated Annealing Algorithm for Modern VLSI Floorplanning Problem,” *ICTACT Journal on Microelectronics*, vol. 2, no. 1, pp. 175–181, Apr. 2016. Available: https://ictactjournals.in/paper/IJME_V2_I1_paper_1_175_181.pdf

[5] J. P. Fang *et al.*, “A Parallel Simulated Annealing Approach for Floorplanning in VLSI,” in *Algorithms and Architectures for Parallel Processing (ICA3PP)*, LNCS, vol. 5574, pp. 291–302, 2009. Available: https://link.springer.com/chapter/10.1007/978-3-642-03095-6_29

[6] H. Mostafa *et al.*, “PARSAC: Fast, Human-quality Floorplanning for Modern SoCs with Complex Design Constraints,” arXiv preprint, 2024. Available: https://arxiv.org/abs/2405.05495

[7] J. Sun *et al.*, “Floorplanning of VLSI by Mixed-Variable Optimization,” arXiv preprint, 2024. Available: https://arxiv.org/abs/2401.15317

[8] S. Banerjee *et al.*, “Satisfiability Modulo Theory Based Methodology for Floorplanning in VLSI Circuits,” arXiv preprint, 2017. Available: https://arxiv.org/abs/1709.07241