Simulated Annealing for VLSI Design

Shrivatsa Gururaj Kulkarni

This manuscript was compiled on August 21,2025

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)) is the integration of thousands to millions of transistors onto a single silicon chip, allowing highly complex and powerful integrated circuits (ICs) to be created (J. Chen et al., 2011; Rajesh and Kumar, 2016) . VLSI has driven the evolution of mordern computing and communication systems and is the basis of microprocessors, memory devices, and application-specific integrated circuits (ASICs). As semiconductor technology continues to advance, the need for devices that are faster, smaller, and consume less energy increases, necessitating efficient VLSI design methodologies more than ever (T.-C. Chen and Chang, 2005; Rajesh and Kumar, 2016).

One of the core steps in the VLSI design process is **chip floor-planning**,which decides the physical organization of the modules on the chip die(T.-C. Chen and Chang, 2005). An optimal floor-plan guarantees minimal wiring complexity, less signal delay, equitable utilization of area, and better thermal and power distribution. On the other hand, if floor-planning is done poorly, it can result in large wire lengths, routing congestion, timing violations, and inefficient chip performance. As the number of modules increases, the number of potential placements grows exponentially, classifying floor-planning as an **NP-hard combinatorial optimization problem** (J. Chen et al., 2011; Kirkpatrick et al., 1983). The complexity has motivated the use of heuristic and metaheuristic methods to find solutions in practice.

Of all the different metaheuristics, **simulated annealing (SA)** has proven particularly effective in VLSI design (Kirkpatrick et al., 1983). SA takes its inspiration from the metallurgical process of annealing, where a piece of material is cooled slowly to reach a state of low-energy crystalline form. The algorithm imitates this by accepting not just better solutions but, with decreasing probability, worse solutions probabilistically. This attribute enables the search to avoid local minima and find an approximation of the global optimum (J. Chen et al., 2011; Fang et al., 2009). In floor-planning, the cost function generally integrates the *total wire length* of interconnections and *overlap penalties* among modules. Simulated Annealing progressively refines placements, with the cooling schedule making worse moves less likely to be accepted as time progresses, resulting in good-quality, non-overlapping layouts.

Hence, simulated annealing represents a strong and realistic framework for solving the challenges of chip floor-planning in VLSI design, balancing computational achievability and solution quality (Mostafa et al., 2024; Sun et al., 2024).

2. Problem Formulation

The **chip floor-planning problem** mathematically can be formulated as positioning an array of rectangular modules within a specified chip boundary to reduce interconnection cost and avoid overlaps (T.-C. Chen and Chang, 2005; Rajesh and Kumar, 2016).A module is defined by its width and height, and its location by the coordinates of its bottom-left corner on the chip. The layout should satisfy geometric and functional constraints while layout quality is optimized.

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 legal placement positions 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 one-to-one direct wire connection between two modules. A net is characterized by the cost of 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 (T.-C. Chen and Chang, 2005):

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

2.3. Overlap Constraint

Modules must not overlap since overlaps are physically infeasible designs. To penalize this constraint, an **overlap penalty** is applied to the cost function if two modules overlap. 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 makes chip floor-planning a **combinatorial optimization problem** in which the search space consists of all the possible module placements and the objective is to identify an optimal-cost layout.

3. Methodology: Simulated Annealing

3.1. Overview

Simulated Annealing (SA) is a probabilistic metaheuristic inspired by the metallurgy annealing process (Kirkpatrick et al., 1983). During physical annealing, a substance is first heated to high temperature and then slowly cooled, allowing atoms to settle into a low-energy crystalline state. In the same way, SA explores the solution space of an optimization problem by accepting probabilistically better solutions, as well as worse ones, with a vanishing probability. This prevents the algorithm from being trapped in local minima and allows it to make an estimate of a global optimum (J. Chen et al., 2011; Fang et al., 2009).

3.2. State Representation

In chip floor-planning, a **state** represents a complete placement of all modules within the chip boundary. Each module m_i is assigned a position (x_i, y_i) to denote 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 defines the quality of a placement and regulates the optimization process. It is defined as:

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

where WL is the total wire length (Equation 1) and P bis the overlap penalty (Equation 2). Lesser cost denotes a more efficient and feasible floor-plan.

3.4. Neighborhood Function

To introduce new solutions, SA uses a **neighborhood function**. The neighboring state is generated by randomly selecting one of the modules and moving it to another new valid location on the chip boundary. This introduces diversity in the search process and allows sequential 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 can be accepted with a probability that decreases with temperature.

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 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 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` , `math` for exponential probability functions, and `matplotlib` for visualization of cost convergence and final placements.
- **Modules:** Each module was a rectangle with defined width and height.
- **Nets:** Nets were modeled as pairs of modules with Manhattan distance used to measure interconnection cost.

4.2. Algorithm Parameters

The following parameters were used in the simulated annealing runs:

- 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 to balance exploration of the search space with convergence efficiency.

4.3. Cost Convergence

Figure 1 shows the cost history over iterations. The cost drops steeply in the initial iterations by accepting good moves, followed by slow fine-tuning as the temperature decreases. This behavior validates that the simulated annealing algorithm does not get trapped in poor local minima and converges to an optimal solution.

4.4. Final Placement

The resulting optimal placement from the algorithm is depicted in Figure 2. Each rectangle denotes a module, labeled with its module number. The placement is non-overlapping and has a small total cost and obeys chip boundaries. Nets between modules are minimized in length, which enhances wire length efficiency.

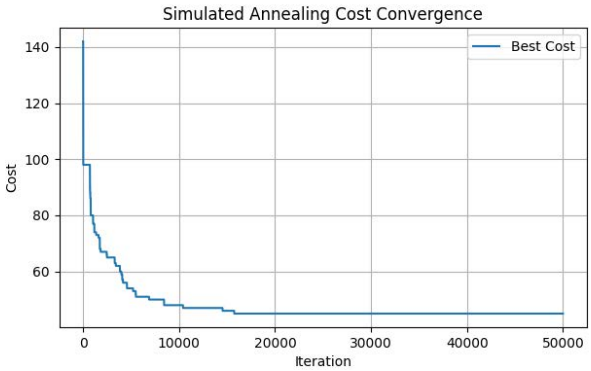

Figure 1. Cost convergence of simulated annealing across iterations.

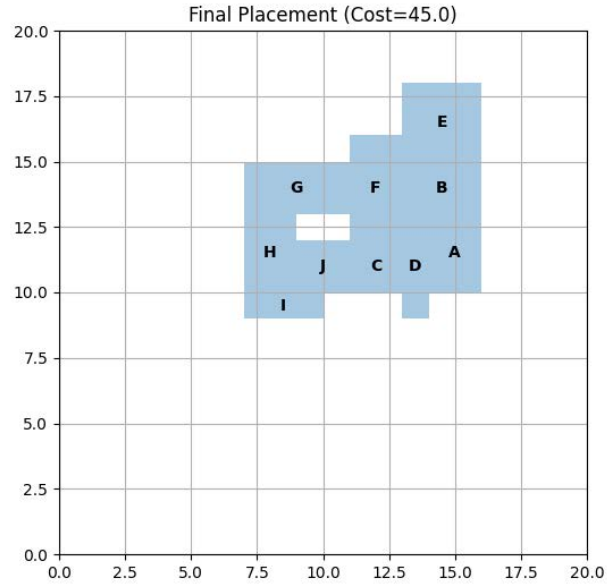

Figure 2. Final optimized placement of modules on the chip.

4.5. Discussion of Results

The results demonstrate that simulated annealing provides an effective approach to solving the chip floor-planning problem (J. Chen et al., 2011; Rajesh and Kumar, 2016). The cost function, that has both wire length 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. Through its representation of modules as rectangles and a cost function that considers total wire length and overlap penalties, the algorithm generated non-overlapping low-cost placements inside the chip boundary. The results confirmed that SA effectively avoids local minima and converges to good-quality layouts (J. Chen et al., 2011; Kirkpatrick et al., 1983), justifying its role as a robust metaheuristic for combinatorial optimization in VLSI design.

However, there are some areas of improvement in the future. Future work can target:

- Investing in higher-level cost functions incorporating more design variables such as thermal constraints (Mostafa et al., 2024; Sun et al., 2024), timing analysis, and power distribution.
- Exploring alternative neighborhood functions such as module swapping, rotation, or multi-module moves to improve solution diversity.
- Exploring hybrid optimization techniques that pair SA with other heuristics such as Genetic Algorithms or Tabu Search for higher quality solutions and convergence rate (Banerjee et al., 2017; Fang et al., 2009).
- Scaling up the application to more difficult and larger designs, to be closer to industrial-sized real-world problems.

6. Applications

The methodologies and techniques used in this project have broad applicability extending beyond the domain of academic research:

- **ASIC and SoC Design:** Floor-planning is a critical process in Application-Specific Integrated Circuits (ASICs) and System-on-Chip (SoC) design, directly impacting performance, power, and area (Mostafa et al., 2024).
- **FPGA Placement:** The same problems of optimization are faced in Field Programmable Gate Arrays (FPGAs), where placement and routing have in the past been treated by simulated annealing.
- **General Optimization:** In addition to VLSI, simulated annealing can be used to solve general optimization problems such as scheduling, resource allocation, and network design.
- **Electronic Design Automation (EDA) Tools:** Modern EDA tools use heuristic-based optimization techniques such as SA, thus making this project most suitably aligned with modern industry practices.

7. Limitations

Even though simulated annealing worked well to create feasible and optimized floor-plans, the current implementation has several limitations which restrict its scalability and industrial applicability:

- **Simplified Cost Function:** The cost function is reduced to wirelength and module overlaps. In reality, VLSI floor-planning also must be taken into account for timing, power, clock distribution, and thermal requirements.
- **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 comprises only moving one module at a time. More sophisticated moves like rotation, swapping, or hierarchical decomposition would improve exploration of the solution space.
- **Cooling Schedule Sensitivity:** The performance of results highly relies on the selection of initial temperature, cooling coefficient, and number of iterations. Bad parameter tuning can result in premature convergence or inordinate runtime.
- **Lack of Industrial Benchmarks:** Artificial module sizes and nets were used in configuring the algorithm. Industry benchmarks such as MCNC or GSRC suites would provide a more comprehensive measure of performance (Sun et al., 2024).

Thus, such shortcomings mean that simulated annealing can be a useful learning and research tool but requires improvement and augmentation with state-of-the-art optimization algorithms in order to be utilized in industrial electronic design automation (EDA) processes.

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