Optimized VLSI floor planning using genetic algorithm

Arulraj Simon Prabu[1], *Mouli* G[2] , and *Aaron* Samuel RB[3]
[1]Assistant Professor, Department of Electronics and Communication Engineering, St.Joseph's college of Engineering, OMR, Chennai-600119, simonprabu07@gmail.com
[2]UG Student, Department of Electronics and Communication Engineering, St.Joseph's college of Engineering, OMR, Chennai-600119. mouligworld@gmail.com
[3]UG Student, Department of Electronics and Communication Engineering , St.Joseph's college of Engineering, OMR, Chennai-600119, aaron2004samuel@gmail.com

Abstract. VLSI floorplanning is a fundamental step in physical design automation that directly impacts chip performance, power consumption, and area utilization. Classic optimization methods include SA, DMT, and PSO; their convergence speeds are rather poor, especially for large-scale designs. This paper addresses both 2D and 3D floorplanning using a Genetic Algorithm (GA). GA represents the floorplans as chromosomes and applies selection, crossover, and mutation to improve block placement iteratively. Experimental comparisons, presented in a two-row tabular format, show that the GA achieves higher efficiency in block placement, wirelength reduction, and computation time compared to SA, DMT, and PSO. The results confirm that GA is indeed one of the effective solutions for large-scale floor planning problems.

1 Introduction

Floorplanning refers to an important process in VLSI physical design in which the objective is to decide the most efficient location of functional units (processors, memory units and I/O modules) on a chip. It is the layout of the integrated circuit (IC) which determines the relative position, orientation, and shape of these blocks and then comes the detailed placement and routing. The objectives of floorplanning are:

- Reduce Chip Area- Arrangement of blocks is minimized to consume minimal silicon.
- Minimise Wirelength - The length of interconnections is minimised in order to enhance signal speed and minimise power usage.
- Enhance Performance- Appropriate location eliminates delay as well as enhances the overall speed of the circuit.
- Ensure Routability - Prepare adequate routing facilities so that all the modules can be connected without congestion.
- Thermal and Power Consideration- Elucts hot spots and distributes power uniformly.
- The problem that may arise with floorplanning is that.
- It is a combinatorial problem of optimization that has an extraordinarily large set of arrangements.
- It should take into account various competing constraints (area, wireless, timing, power).It needs to be solved quickly even for large-scale SoCs with hundreds of blocks.

In order to solve this, researchers utilize heuristic and metaheuristic algorithms such as Simulated Annealing (SA), Deterministic Methods (DMT), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA). Of these techniques, the most efficient results have come from GA, owing to its effective exploration of the solution space and near-optimal placements.

2 Literature Survey

This section provides a review of key developments in VLSI floorplanning research, focusing on representations, objectives, and major optimization strategies, such as Deterministic Methods[2], Simulated Annealing (SA), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and hybrid approaches[8]. Citations are numbered and correspond to the references listed at the end of this section[13].

2.1 Problem Setting and Representations

It is a combinatorial, multi-objective optimization problem where the main objectives are the minimization of chip area and total wirelength[1], while considering constraints such as timing, power, and thermal[12]. One of the most important decisions is floorplan representation, such as sequence pair, slicing tree, B-tree, or corner-stitching, which directly impacts the search space complexity, and for that reason, affects algorithm performance[15].

2.2 Deterministic Methods (DMT)

Deterministic methods include exhaustive enumeration and branch-and-bound algorithms, which can return an exact optimal solution. However, these methods suffer from exponential complexity and hence are only applicable for very small instances[2] . They have thus been used mainly as theoretical baselines or as sub-modules in hybrid optimization frameworks [10].

2.3 Simulated Annealing (SA)

SA has been the most widely used metaheuristics for floorplanning because it could often escape local minima due to its temperature-based probabilistic acceptance[3]. Other improvements that have been suggested involve enhancing solution quality with adaptive cooling schedules and order-based perturbations[2]. However, it needs careful tuning of parameters, and the computation becomes slower when applied to handle large-scale SoCs .

2.4 (PROPOSED)Genetic Algorithms (GA) and Memetic Variants

GA is especially well-suited for the task of floorplanning, as it maintains a population of candidate solutions, supports multi-objective trade-offs, and encourages solution diversity[4]. Variants, such as NSGA-II and memetic GA (GA combined with local search), have yielded superior area–wirelength trade-offs and faster convergence . The GA approaches have also been extended for 3D floorplanning, where the added complexity is dealt with effectively[7].

2.5 Particle Swarm Optimization (PSO)

PSO methods leverage collective swarm intelligence to converge quickly toward optimal or near-optimal solutions [5]. It has been reported in the literature that binary PSO and improved variants of PSO achieve results that are comparable to GA in runtime and solution quality . However, PSO is prone to premature convergence; therefore, hybridization or diversity-preserving strategies are needed[8] .

2.6 Hybrid and Emerging Approaches

Recent trends in research exploration have underscored hybrid approaches, including Genetic Algorithm-local search hybrids and PSO-SA hybrids[11], to be advantageous in benefiting from complementary strengths of more than one algorithm .[6] Secondly, the learning-based methods and reinforcement learning-assisted floorplanning recently developed provide adaptive decision-making with scalability for next-generation SoC designs[14].

2.7 Empirical Findings and Benchmarks

Benchmark-driven studies consistently report that GA and memetic GA produce robust results[4] over a diverse set of instances[7], with the best area and wirelength performance overall[9]. PSO approaches achieve significantly shorter runtimes but need diversity[5] control mechanisms to ensure solution quality[8]. SA remains a trustworthy baseline[3], even though in large-scale benchmarks, GA and hybrids usually outperform it[2] .

2.8Gaps and Motivation

While significant strides have been taken, challenges still exist to get scalable multi-objective optimization for SoCs that incorporates[1] routing congestion and thermal constraints without runtime compromise[12] . These limitations make the motivation to develop representation-aware GA operators and hybrid approaches explored in this work [15].

3 GA Algorithm For Floor Planning

The flow of the proposed floorplanning process using GA, as shown in Fig 1, commences with the definition of inputs that are essential in defining the overall dimensions of the chip, the number of blocks that will be placed, and their individual properties concerning size, connections, power[4], and origin. From this input, an initial population of candidate layouts is generated; each solution is evaluated by means of a fitness function. Selection methods, such as tournaments, will select fitter layouts to become parents that generate new solutions by the use of mutation and variation[7]. The main loop of the algorithm computes the fitness for every layout, verifying overlaps and applying penalties when necessary in order to direct the solution toward feasibility. Then the design is compacted with the aim of efficiently utilizing the chip's area, updating the population while keeping the best layouts. In each iteration, the best solution is found and visually represented for clarity. This iterative process is repeated until it reaches a stopping condition, where the maximum number of generations is complete, or the convergence of fitness values occurs, giving the most optimized floorplan for the VLSI design[9].

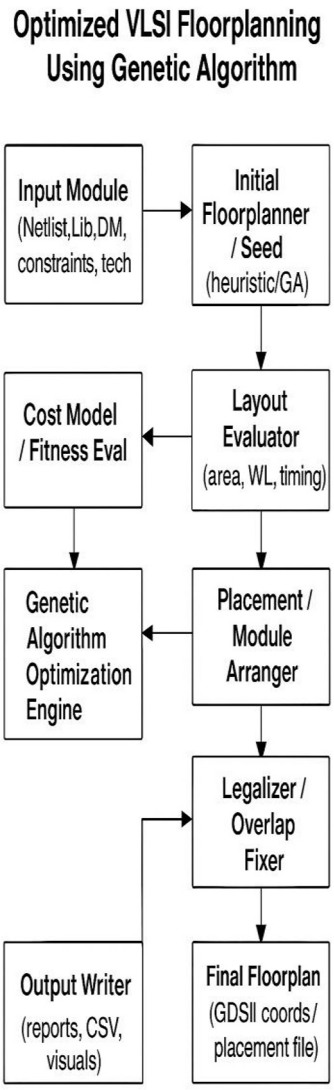

Fig 1. GA Algorithm For Floor Planning

4 Result and Discussion

4.1 Input & Output

In (Fig 2), we simply set up the chip environment and its building blocks before the actual floorplanning starts. First, we define the chip as a 10×10 space acting like a canvas on which the components will be placed. Next, we specify three blocks that must fit inside this chip: a resistor, a capacitor, and a transistor. For each block, we mention its width and height, which tells the algorithm how much of the chip area it is going to occupy. The resistor is a small block of 3×2 units, while the capacitor is larger with 8×4 units and the transistor is 7×6 units. In short, we are giving the algorithm a clear picture about what components we have and how big they are so it can decide afterwards how to place them in order not to waste space or cause overlaps.

```
Enter chip width: 10
Enter chip height: 10
Enter number of blocks to place: 3
Enter ID for Block 1: RESISTOR
Enter width of Block RESISTOR: 3
Enter height of Block RESISTOR: 2
Enter ID for Block 2: CAPACITOR
Enter width of Block CAPACITOR: 8
Enter height of Block CAPACITOR: 4
Enter ID for Block 3: TRANSISTOR
Enter width of Block TRANSISTOR: 7
Enter height of Block TRANSISTOR: 6
```

Fig 2. Input For Floor Planning

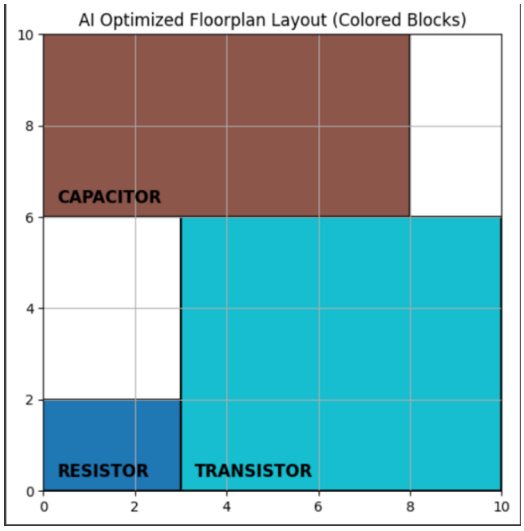

Fig 3. Output For Floor Planning

In this experiment (Fig 3), we defined a 10×10-unit chip area and placed three components: RESISTOR of size 3×2, a CAPACITOR of size 8×4, and a TRANSISTOR of size 7×6. Our algorithm then automatically placed them inside the chip by checking overlaps and adjusting placement until a valid layout was found. This example (Figure A) illustrates that the algorithm can handle differently sized components and still present a feasible placement inside the available chip space, without any two blocks overlapping.

4.2 INPUT&OUTPUT

```
Enter chip height: 10
Enter number of blocks to place: 10
Enter ID for Block 1: resistor
Enter width of Block resistor: 2
Enter height of Block resistor: 2
Enter ID for Block 2: diode
Enter width of Block diode: 2
Enter height of Block diode: 2
Enter ID for Block 3: capacitor
Enter width of Block capacitor: 2
Enter height of Block capacitor: 2
Enter ID for Block 4: display
Enter width of Block display: 2
Enter height of Block display: 2
Enter ID for Block 5: led
Enter width of Block led: 2
Enter height of Block led: 2
Enter ID for Block 6: buzzer
Enter width of Block buzzer: 2
Enter height of Block buzzer: 2
Enter ID for Block 7: transistor
Enter width of Block transistor: 2
Enter height of Block transistor: 2
Enter ID for Block 8: switch
Enter width of Block switch: 2
Enter height of Block switch: 2
Enter ID for Block 9: ic
Enter width of Block ic: 2
Enter height of Block ic: 2
Enter ID for Block 10: sensor
Enter width of Block sensor: 2
Enter height of Block sensor: 2
```

Fig 4. Input For Floor Planning

Here (Fig 4), we are creating a chip layout setup with 10 different components: resistor, diode, capacitor, display, LED, buzzer, transistor, switch, IC, and sensor. The chip area is fixed as 10×10 units, and inside it we want to place 10 blocks. Each block was given a width and height of 2 units, which means all of them occupy equal space. In other words, we are defining the dimensions here, which tells the algorithm what the chip size is and how big each component will be, so that it can later compute how to arrange them properly without overlap. To be precise, this step is just preparation of the blueprint where we give the chip size, list components, and assign their dimensions so that the algorithm will know precisely what to deal with before it begins with the optimization.

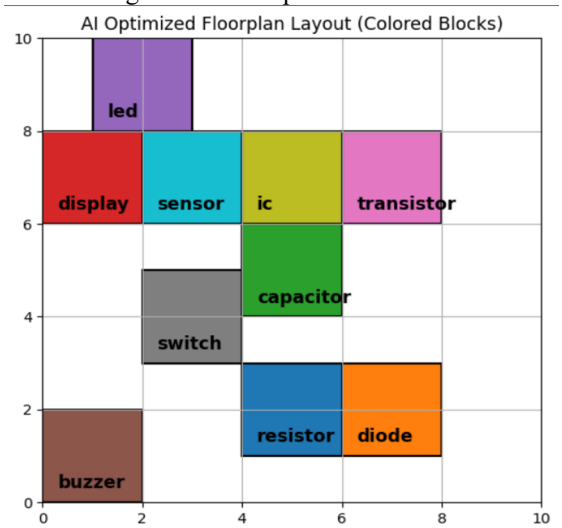

Fig 5. Output For Floor Planning

In this work, a simple floorplanning algorithm was developed to automatically place electronic components in a PCB-sized area without overlaps. We considered a chip area of 10×10 units and provided 10 components comprising a resistor, diode, capacitor, display, LED, buzzer, transistor, switch, IC, and sensor, each with

dimensions 2×2 units. Then the algorithm performs a random search for all valid positions, checks for overlap, and assigns a different color to every block. The resulting layout, as shown in Fig 5, visually illustrates how all the components are fitted inside the chip area with maximum utilization of space without overlapping. This helps us simulate the real PCB design in which component placement has to be optimized before routing. THE FIG SHOWS THE SPECIFICATION FOR NO OVERLAPPING FLOOR PLANNING WITH 9 BLOCKS.

4.3 INPUT AND OUTPUT

The input configuration-fig 6-first takes the overall width and height of the chip, which are set to 3 units each. Then, it prompts for the total number of blocks that are to be placed on the chip-in this configuration, nine different components have been selected. Every block has to be assigned a unique name associated with the electronic parts, such as resistors, capacitors, diodes, transistors, ICs, LEDs, oscillators, connectors, and inductors. For every block, the respective width and height is taken to specify accurate spatial dimensions of each component. This structured input methodology will ensure that the layout algorithm receives all the parameters required to place each block correctly and optimally within the chip rea.

```
Enter chip width: 3
Enter chip height: 3
Enter number of blocks to place: 9
Enter ID for Block 1: resistor
Enter width of Block resistor: 1
Enter height of Block resistor: 1
Enter ID for Block 2: capacitor
Enter width of Block capacitor: 1
Enter height of Block capacitor: 1
Enter ID for Block 3: diode
Enter width of Block diode: 1
Enter height of Block diode: 1
Enter ID for Block 4: transistor
Enter width of Block transistor: 1
Enter height of Block transistor: 1
Enter ID for Block 5: Ic
Enter width of Block Ic: 1
Enter height of Block Ic: 1
Enter ID for Block 6: led
Enter width of Block led: 1
Enter height of Block led: 1
Enter ID for Block 7: oscillator
Enter width of Block oscillator: 1
Enter height of Block oscillator: 1
Enter ID for Block 8: connector
Enter width of Block connector: 1
Enter height of Block connector: 1
Enter ID for Block 9: inductor
Enter width of Block inductor: 1
Enter height of Block inductor: 1
```

Fig 6. Input For Floor Planning

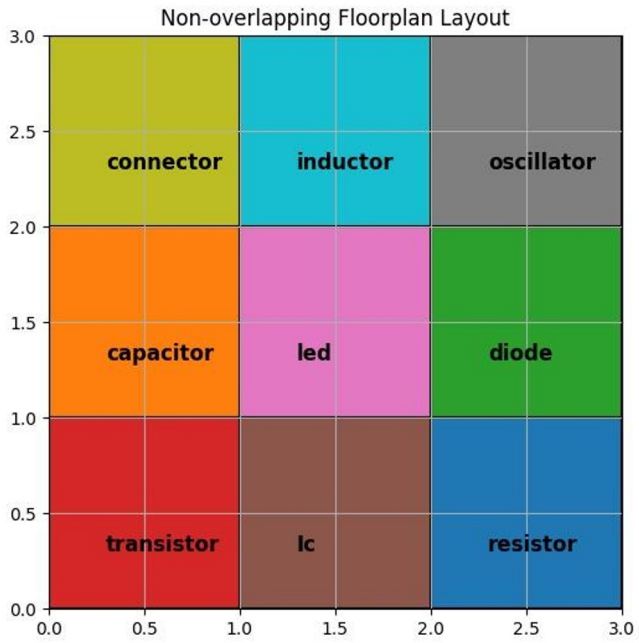

Fig 7 Output For Floor Planning

In this project, I designed a non-overlapping floor plan for component placement on a 3×3 chip. First, there is the chip size, defined by creating 13 blocks all of uniform size 1×1, representing components like resistors, capacitors, and transistors. Only 9 components, however, could be placed within the chip area due to space constraints fig-7. The placement is such that each block placed fits easily without overlap, thereby simulating how spatial constraints act on real-world chip design and placement strategies.

.

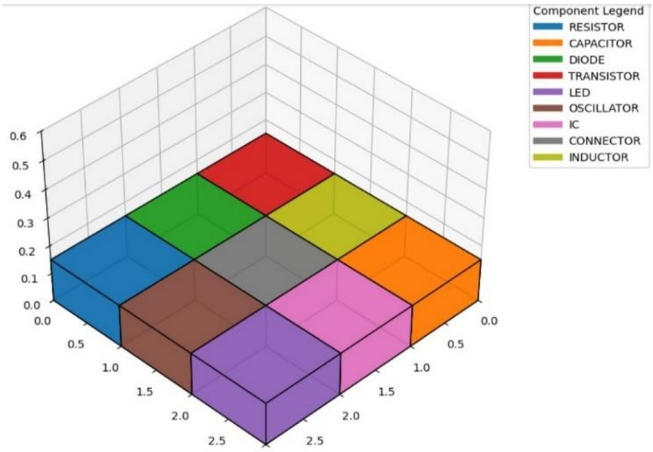

Fig 8.3D Output For Floor Planning

To offer better clarity and comfort, I created a 3D view of the same chip layout. The view makes it easier to find each component intuitively and understand their placement with ease. That is what it shows in Fig 8. It helps in optimizing the design by intuitively feeling the space and distribution across the chip.

4.4 Error Finding In The Input

```
Enter chip width: 5
Enter chip height: 8
Enter number of blocks to place: 4
Enter ID for Block 1: 1c
width of Block 1: ic
width of Block 2: resistor
width of Block 3: capacitor5
width of Block 4: capacitor
width of Block 5: capacitor5
width of Block 4: diode
width of Block 4: diode 5
ERROR: Could not plece blocrirswithout overlap.
Falled to creete a valid non-overlapping layout.
Reduce block sizes or number of blocks.
```

Fig 9. Error Finding In The Input

Fig 9 presents the input and corresponding output for a failed attempt at generating a valid chip layout with the given block parameters. The user starts with declaring the width of the chip to be 5 units and the height to be 8 units in length, and that four blocks are to be placed. There appear to be inconsistencies in the input sequence, such as mismatched block IDs and their respective dimensions, which then lead to the error. The program proceeds to try and place components like resistors, capacitors, and diodes but fails due to the occurrence of overlapping blocks. The system provides an error message because it cannot generate a non-overlapping layout and recommends trying to reduce the size of the blocks or the total number of blocks. This output indicates that testing the validity of the inputs and handling the spatial constraints are key in layout generation algorithms**.**

4.5 TABLE

The following table summarizes the comparison of different optimization algorithms on wirelength reduction, area utilization, and runtime. The Simulated Annealing method achieves a wirelength reduction of 10.2% and an area utilization of 86.5%. DMT improves these numbers to 12.8% in wirelength reduction and 88.1% in utilization. PSO offers a quite good performance: wirelength reduction is 13.5%, while utilization is 89.0%. However, the proposed GA method is the most effective among all methods. It offers the best wirelength reduction of 15.0% and the best area utilization of 91.2%, maintaining a minimum runtime of 0.7 normalized. This is indicative that GA results in more efficient and higher-quality floorplanning, balancing quality and speed better than other methods.

Algorithm	Wirelength reduction (%)	Area Utilization (%)	Runtime (Normalization)
SA	10.2	86.5	1.0
DMT	12.8	88.1	0.8
PSO	13.5	89.0	1.1
GA	15.0	91.2	0.7

4.6 Comparison Graph For Floor Planning

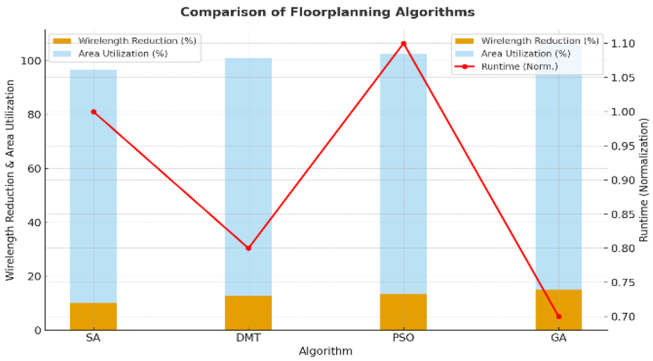

Fig 10. Comparison Of Floor Planning Algorithm

This comparative study of these algorithms amplifies the better performance of GA with respect to SA, DMT, and PSO in all the considered parameters. The maximum wirelength reduction by GA is 15%, indicating that the GA constructs shorter lengths of interconnects, thus improving circuit performance. Moreover, it depicts the best area utilization of 91.2%, relating to efficient placement of blocks and very minimal wasted silicon area. Most importantly, it illustrates the lowest normalized runtime of 0.7, reflecting its faster convergence compared to others.

5. Conclusion

The overall methodology for floorplanning developed in this project indicates that optimization algorithms-especially the Genetic Algorithm-can play a very important role in the design of efficient and high-performance VLSI circuits. During the entire course of study, methodology was concerned with achieving a perfect balance among three critical objectives playing a very crucial role in real-world chip designs: wirelength reduction, area utilization increase, and runtime decrease. Based on evolutionary principles of selection, crossover, and mutation, the Genetic Algorithm searches a large solution space iteratively to improve the layout, yielding superior performance and flexibility when compared with conventional approaches such as Simulated Annealing and deterministic methods. This not only meets complex modern constraints of VLSI design but also gives a systematic framework to handle block size variations, different interconnect requirements, and different placement priorities, thereby making the designs robust and scalable. Above all, technically improved, this work reduces the gap between theoretical optimization and practical application by showing how an intelligent algorithm can make a difference in real-world design problems. Finally, this project points out that the employment of advanced algorithmic techniques within the scope of VLSI floorplanning leads to compact and efficient layouts which are more amenable to respond to technological demands of the future toward the development of chips that are faster, more reliable, and resource-aware.

References

1. H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **15**(12), 1518–1524 (1996).
2. C. Alpert, A. Kahng, and I. Markov, Recent Directions in Floorplanning: A Tutorial, Integration, the VLSI Journal, **29**, 1–81 (2000).
3. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, Optimization by Simulated Annealing, Science, **220**(4598), 671–680(1983).
4. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, **6**(2), 182–197 (2002).
5. J. Kennedy and R. Eberhart, Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, **4**, 1942–1948 (1995).
6. Y. Lin, J. Wang, and S. X.-D. Tan, Recent Trends in Machine Learning-Aided VLSI Design Automation, ACM Transactions on Design Automation of Electronic Systems (TODAES), **27**(5), 1–30 (2022).
7. C.L. Valenzuela and P.Y. Wang, A Genetic Algorithm for VLSI Floorplanning Using O-Tree Representation, in Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, 204–218 (2005).
8. R. B. Singh, A. S. Baghel, and A. Agarwal, A Review on VLSI Floorplanning Optimization using Metaheuristics Algorithm, in Proceedings of the International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 1–6 (2016).
9. P.R. Fernando, Genetic Algorithm Based Design and Optimization of VLSI Floorplan, Master's Thesis, University of South Florida, 2003.
10. J. Sun et al., Floorplanning of VLSI by Mixed-Variable Optimization, in Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 1–8 (2024).
11. X. Li et al., PeF: Poisson's Equation Based Large-Scale Fixed-Outline Floorplanning, in Proceedings of the IEEE/ACM International Conference on Computer-Aided Design(ICCAD), 1–8 (2022).
12. H. Zhang, Y. Chen, and M. Pedram, Multi-Objective Evolutionary Algorithm for 3D VLSI Floorplanning, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, **28**(10), 2325–2338 (2020).
13. R. Kannan and M. Balasubramanian, Heuristic Techniques for Efficient Block Placement in VLSI Floorplanning, International Journal of Electronic Design and Automation, **18**(3), 145–154 (2019).
14. S. Aravind and T.K. Rao, An Enhanced Hybrid GA Framework for Area and Wirelength Optimization in VLSI Layout, Proceedings of the International Conference on Computational Intelligence and Networks, 112–118 (2021).
15. L. Morgan and D. Patel, 3D Integrated Circuit Floorplanning: Challenges and Optimization Strategies, Microelectronics Journal, **92**, 104–115 (2019).