ISSN: 2424-8460(O)
2251-2608(P)

Original Research Article

A review on advances in design methodologies for integrated circuits

Qingye Wei

The University of Southampton, Southampton, Hampshire, SO17 1BJ, United Kingdom

Abstract: The field of integrated circuit (IC) design has witnessed remarkable advancements driven by the need for higher performance, reduced chip size, and enhanced energy efficiency. This review explores state-of-the-art methodologies in IC design, with a focus on layout optimization and the application of metaheuristic algorithms such as simulated annealing and genetic algorithms. It also examines novel representation techniques like sequence pair, B*-tree, and corneal block sequences that address layout complexity. Furthermore, the study proposes integrating machine learning, reinforcement learning, and predictive models to enhance the adaptability and efficiency of design processes. By offering a comprehensive overview and presenting forward-looking perspectives, this review aims to contribute to the development of next-generation ICs, addressing scalability challenges and ensuring improved performance for advanced electronic systems.

Keywords: Integrated circuits; Layout planning; Meta-heuristic Algorithms

1. Introduction

The field of integrated circuit (IC) design has evolved significantly, driven by the need for higher performance, smaller chip sizes, and more efficient power usage. With the advancement from two-dimensional (2D) integrated circuits to three-dimensional (3D) designs, new challenges and opportunities have emerged, especially in optimizing the layout and addressing manufacturing constraints such as TSV (Through-Silicon Via) technology. Traditional IC design methodologies, while powerful, often face limitations when it comes to the complexity and integration density of modern circuits. As such, there is a growing interest in novel approaches that can push the boundaries of what is possible in IC design.

This paper reviews recent advances in various IC design methodologies, with a particular emphasis on optimization methods, including reinforcement learning and metaheuristic algorithms. By exploring both conventional and emerging approaches, we provide a comprehensive understanding of the methodologies that are pushing the boundaries of IC design. Beyond the existing research, we also propose novel perspectives for overcoming some of the critical challenges faced in this domain.

Our perspective emphasizes the need for more adaptive and hybrid approaches that integrate reinforcement learning with traditional metaheuristic methods to enhance the efficiency of layout optimization. Specifically, we suggest that reinforcement learning can be leveraged not only for layout exploration but also for adaptive parameter tuning within metaheuristic algorithms, thereby addressing one of the key limitations of manual hyperparameter adjustment. Moreover, we propose a synergistic use of machine learning models that can predict potential dead-ends in layout configurations early, thereby reducing computation time significantly. This proactive use of predictive models within optimization processes, coupled with feedback-based reinforcement, could lead to more effective and robust IC designs.

In this review, we aim to not only present the state-of-the-art methodologies but also offer a new conceptual

framework for the future of IC design optimization. We highlight the potential of interdisciplinary techniques, including deep learning, graph-based neural networks, and optimization theory, to reshape the design landscape. These perspectives, we believe, will play a cruc ial role in addressing scalability issues, improving design quality, and ultimately enabling the development of more powerful integrated circuits for the next generation of electronic devices.

2. Integrated circuits design

2.1. VLSI design flow

The VLSI design process is highly complex and can be broadly divided into structural design, functional design, logical design, circuit design, and physical design. Before commencing the design, the system specifications are first defined to set out the overall objectives and high-level requirements of the system. These requirements cover aspects such as functionality, performance, physical dimensions, and manufacturing technology. Following these specifications, a detailed design specification is prepared to guide the entire design process. The VLSI design process, as illustrated in **Figure 1**, includes the following stages:

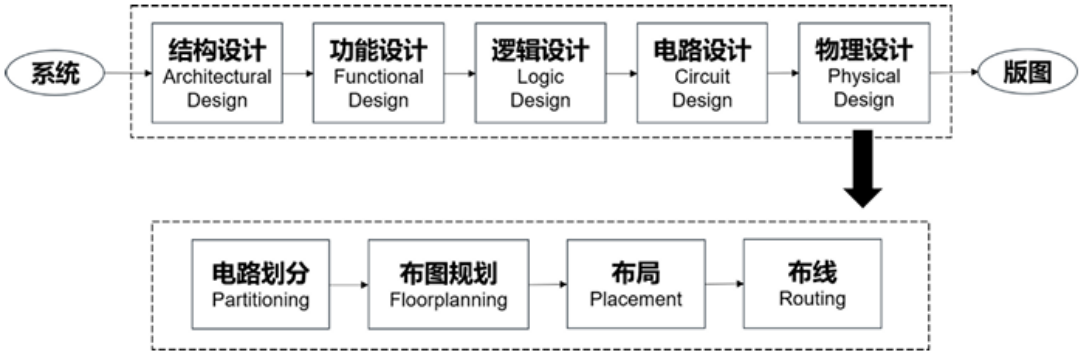

Figure 1. VLSI design flow chart.

The architectural design of the chip is first determined to meet the system specifications, ensuring that it satisfies various system requirements. This includes the integration of analog and mixed-signal modules, memory management, the types and numbers of computation IP cores, chip-to-chip communication, support for standard protocols, the use of both soft and hard IP modules, pin assignments, packaging, chip package interfaces, power requirements, and the selection of process technology. After the architecture is established, the functional and logic design phase begins. In this phase, the functions of each module and their interconnections are defined. The functional design specifies the inputs, outputs, and timing requirements for each module, while the logic design phase involves defining the chip's functionality and timing logic at the register-transfer level (RTL) using hardware description languages (HDL). Following this, the circuit design phase addresses the design of key low-level elements, which are mainly handled at the transistor level. While logic synthesis tools automatically convert Boolean expressions into gate-level netlists for most digital logic on the chip, the circuit design focuses on elements such as static RAM, modules, input/output blocks, and analog circuits. The correctness of these designs is verified using circuit simulation tools like SPICE. Finally, in the physical design phase, all components are instantiated with geometric representations, and layout methods are used to place the components in optimal positions. Routing algorithms are then applied to interconnect the components, resulting in the final chip layout, which is used for subsequent manufacturing, testing, and packaging.

2.2. VLSI physical design

Physical design is the most critical and time-consuming step in the entire VLSI design process. The input to this stage is the circuit components and their connection netlist, which are transformed from their circuit representation into geometric representations. The netlist, which shows the interconnection relationships between components, is converted into geometric routing patterns. This phase determines the geometric positions of the components on the chip, optimizes relevant metrics, and produces the final layout. Physical design directly impacts several key performance indicators of the chip, including:

First, performance can be affected by excessively long or poorly designed interconnections, leading to signal delays and increased power consumption. Second, area is influenced by the placement of interconnected modules at distant locations, which can increase the chip's area and, in turn, affect both cost and performance. Third, reliability is impacted by the presence of numerous vias, which can significantly reduce the circuit's overall reliability. Fourth, power consumption is determined by factors such as the gate length of transistors. Smaller gate lengths enable higher operating speeds but may result in higher leakage currents and greater manufacturing variability. Larger gate lengths, on the other hand, tend to lead to higher dynamic power consumption. Finally, yield can be compromised if the wiring is too dense, which may cause short circuits during manufacturing and ultimately reduce the yield of the final product.

The physical design process is highly complex and critical to the overall chip design, and it can be broken down into several key steps. The first step is circuit partitioning, which involves dividing the circuit into smaller sub-circuits or modules to reduce the complexity of the problem and define the interconnections between these sub-circuits or modules. Each sub-circuit or module can then be designed or analyzed independently. Next, floorplanning is used to determine the shape and location of each sub-circuit or module, as well as the positions of IP blocks or macro-modules. The third step is placement, where the positions of standard cells are determined. Finally, routing is performed, which consists of two main stages: global routing and detailed routing. Global routing allocates the resources for the connecting lines, while detailed routing places these lines onto specific metal layers and routing channels.

2.3. TSV technology

Three-dimensional integrated circuits (3D ICs) offer several advantages, including smaller chip area, shorter wire lengths, higher memory bandwidth, and easier implementation of heterogeneous integration. These characteristics make 3D ICs a promising solution for continuing Moore's Law. One of the most important technologies in 3D integration is vertical interconnection, specifically through TSV (Through-Silicon Via) technology. 3D ICs achieve vertical interconnection and communication through TSVs.

TSV technology has already been adopted by companies such as IBM, Toshiba, and STMicroelectronics for production. TSVs can be manufactured using three different methods: Via-first, Via-middle, and Via-last. In the Via-first method, vias are created before CMOS (Complementary Metal Oxide Semiconductor) processing. To avoid potential issues with temperature cycling in CMOS processing, thermal resistive materials are used to fill the TSVs. In the Via-middle method, vias are created between the CMOS process and the Back-End of Line (BEOL) process. In this case, TSV fillers can be made from common materials such as copper. The Via-last method involves creating vias after the BEOL process or bonding step. This allows the circuit layers to be fabricated first in the factory, followed by alignment and bonding in the packaging workshop. Due to the

ISSN: 2424-8460(O)
2251-2608(P)

manufacturing process, TSV diameters in the Via-last method range from 5μm to 20μm, which is larger than the diameters in the Via-first and Via-middle methods, where TSV diameters range from 1μm to 5μm. As a result, the alignment accuracy of the Via-last method is lower compared to the other two methods. Additionally, TSVs in all three methods block the device layers, but only the Via-last method blocks the metal layers.

The Via-first and Via-middle structures are similar. **Figure 2** shows the structural diagrams of Via-first (left) and Via-last (right).

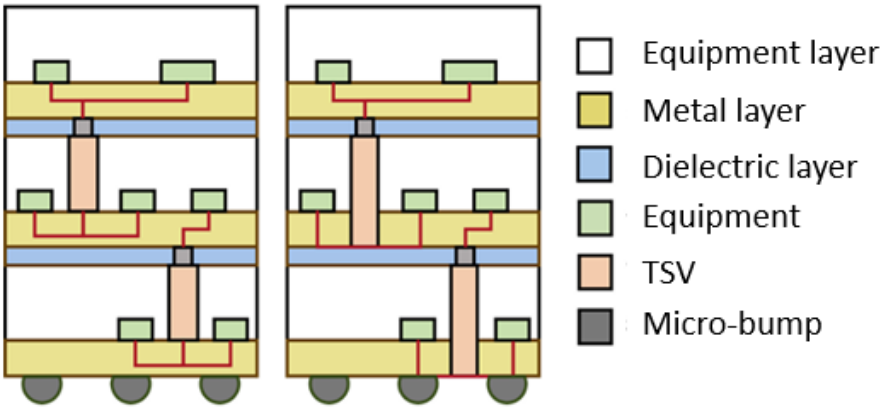

Figure 2. Structure diagram of the Via-first and Via-last TSV.

3. Layout planning representation

Layout planning representation directly determines the size of the space and the complexity of encoding and decoding, and choosing the appropriate layout planning representation is crucial to the layout planning problem.

The geometric relationship between circuit modules is usually determined by dividing the layout area into several rectangles. In order to limit the size of the solution space, the researchers proposed three different layout structures, namely Slicing structure, Mosaic structure and General structure. Divisible structure can be considered as a special case of Mosaic structure, which is a special case of general structure.

Divisible structures can be obtained by recursively dividing the layout in either vertical or horizontal directions. In the Mosaic structure, the layout is divided into rectangular Spaces, each of which is occupied by one and only one module. General structure Compared to Mosaic structure, the layout-area can be divided into more space than the number of modules, some of which will not be occupied by any module. Both the Mosaic structure and the general structure can represent an indivisible structural layout. In addition to these three structural categories, the researchers also introduced a Compact type of packaging, such as L-compact (B-compact), which means that no module in the layout can be moved to the left (down) while the other modules are fixed. In the layout formed by modules a, b, and c in **Figure 2.3**, module c can be moved to the left, so the layout is not compact. Thus, a compact package is a special structure whose solution space intersects the solution space of a Mosaic structure and a divisible structure.

Figure 3 shows the interrelationship diagram of solution space of divisible structure, Mosaic structure, general structure and compact structure:

ISSN: 2424-8460(O)
2251-2608(P)

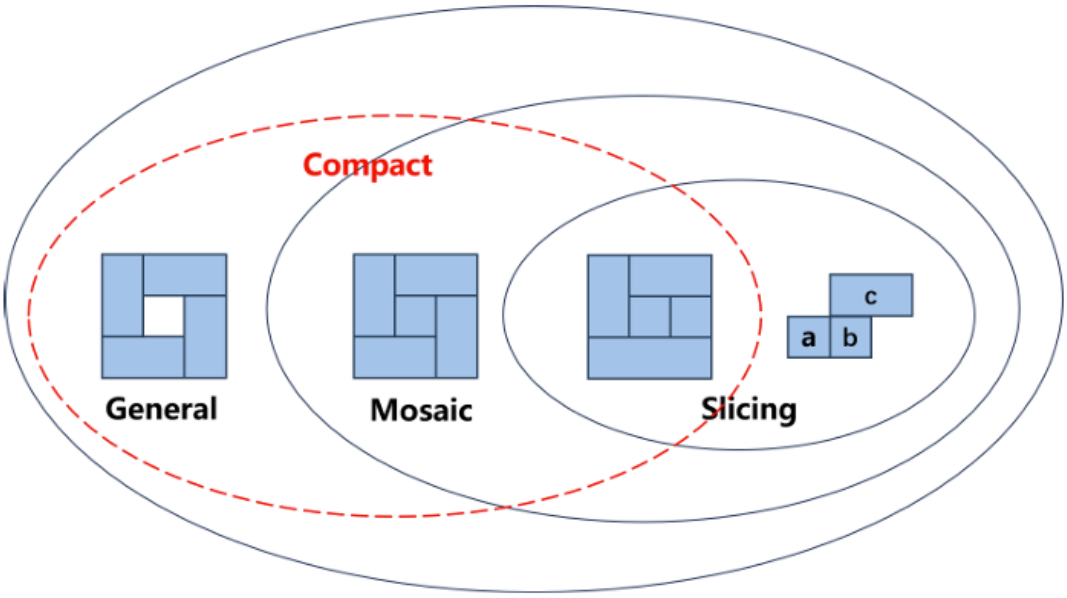

Figure 3. Layout structure solution spatial relationship diagram.

3.1. Standard polish notation

The divisible structure is obtained by dividing the layout recursively vertically or horizontally, so it is represented by a directed rooted binary tree called a "slice tree". Each internal node in the tree is marked by the "*" or "+" operator, corresponding to the vertical partition and horizontal partition respectively, and each leaf node corresponds to the module, which is represented by a number from 1 to n. However, given a divisible layout, there may be multiple slice trees corresponding to it.

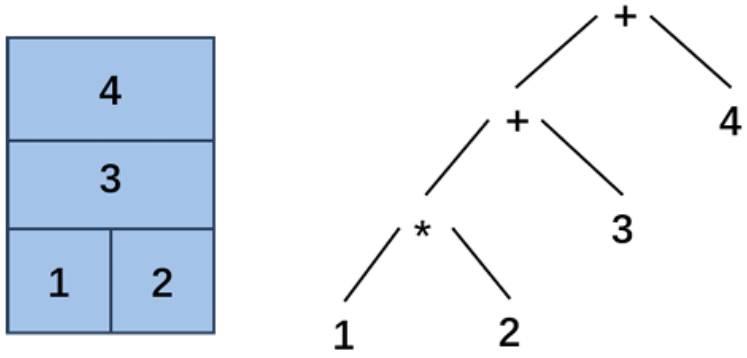

Figure 4. Layout and its standard Polish representation.

In order to obtain a non-redundant representation of a divisible structure layout, Wong and Liu proposed a Skewed Slicing Tree, in which no node in a skewed slicing tree is the same as its right child node, that is, there is no continuous "*" and "+" operators in the expression. They used the post-order traversal of the slant slice tree as a representation of the layout plan and called it standard Polish representation, and it also proves that the divisible layout of n modules has a unique corresponding Standard Polish Notation with a length of 2n−1. For example, in **Figure 2.4**, the layout of four modules corresponds to the standard Polish notation "12*3+4+".

The standard Polish representation corresponds to the actual layout one by one, there is no redundant

representation, and its solution space is small. However, the standard Polish notation can only represent the layout of the partition structure, which has certain limitations.

3.2. Corneal block sequence representation

Corneal Block sequence representation (Corner Block List; CBL belongs to the layout representation of Mosaic structure, and the Mosaic structure conforms to the following three characteristics:

1. There is no blank area in the layout, and each divided rectangular area has and only one module. In a layout without white space, the internal line segments intersect to form T-connections, each of which consists of a non-penetrating line segment and a penetrating line segment, as shown in **Figure 5**.

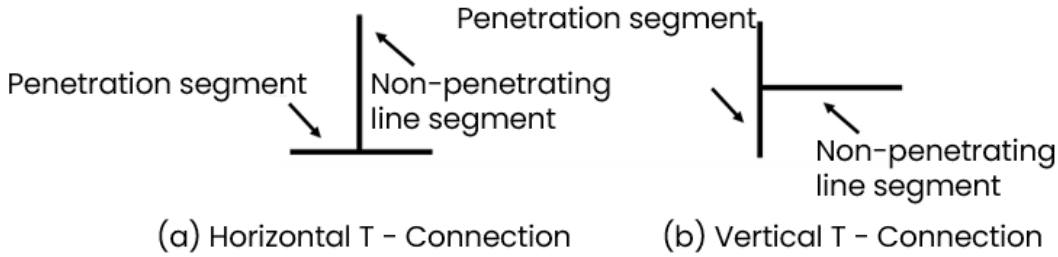

Figure 5. T - Connection.

2. Topological equivalence when line segments slide. The two topologies before and after the sliding of the non-penetrating line segment in the defined T-connection are the same, and the position of the sliding line segment is determined by the actual length and width of the module, as shown in **Figure 6**.

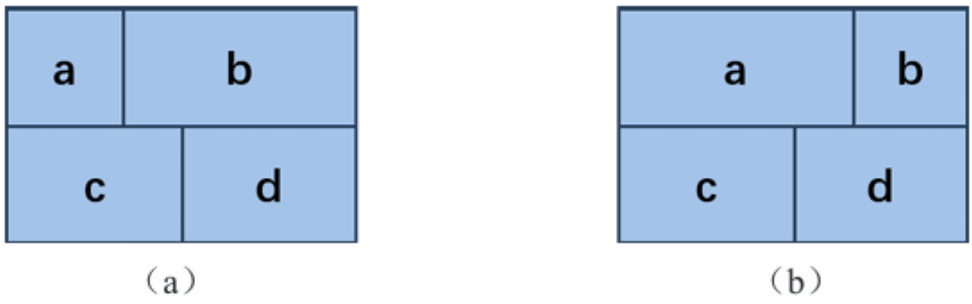

Figure 6. Topological equivalence when line segments slide.

3. The topology is not degenerate. Two different T-connections are not allowed to coincide at the same point, and if this degradation occurs, slide the non-penetrating segment of one of the T-connections by a small distance to separate the two T-connections. The corneal block sequence representation has the following relevant definitions:

Definition 1: A module is said to be a corner module in which it sits in the layout structure if the sub-module is located in the upper-right most region of the layout structure. Lemma 1: Given a Mosaic structure with one or more modules, it must have a unique one called a module.

Definition 2: A T-connection is called the corner T-connection of the current layout-structure if the T-connection consists of the dividing line where the left and lower sides of the corner module in the layout-structure are located.

ISSN: 2424-8460(O)
2251-2608(P)

Definition 3: Direction of T-connection: The direction of a T-connection is called the vertical direction if the T-connection can be obtained by rotating it 90° counterclockwise. The direction of a T-connection is called horizontal if the T-connection can be rotated 180° counterclockwise by the T-type.

Definition 4: A T-junction is called an implicated T-junction of a corneal mass if its penetrating segment is the same dividing line as the non-penetrating segment of the angular T-junction.

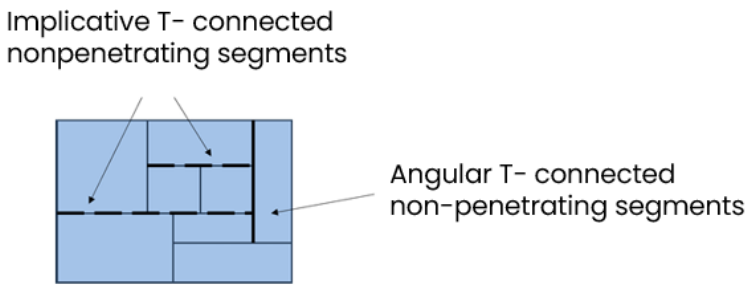

Figure 7. Angular T-connections and implicative T-connections.

Definition 5: Direction of the corner module: the direction of the corneal block is the same as the direction of the corresponding Angle T- connection, and in the corneal block representation method, "1" represents the horizontal direction and "0" represents the vertical direction.

Deletion of a corner block (**Figure 8.**): If the corner block is horizontal, move its left boundary to the right boundary of the chip to delete the corner block, and drag its left boundary's attached T-junction to the right boundary. If the corner block is vertical, move its lower boundary to the upper boundary of the chip to delete the corner block, and drag its lower boundary's attached T-junction to the upper boundary. The information of the attachments is recorded by a binary string Ti. The binary string Ti consists of consecutive 1s and a 0 indicating the end. The number of consecutive 1s represents the number of attached T-junctions.

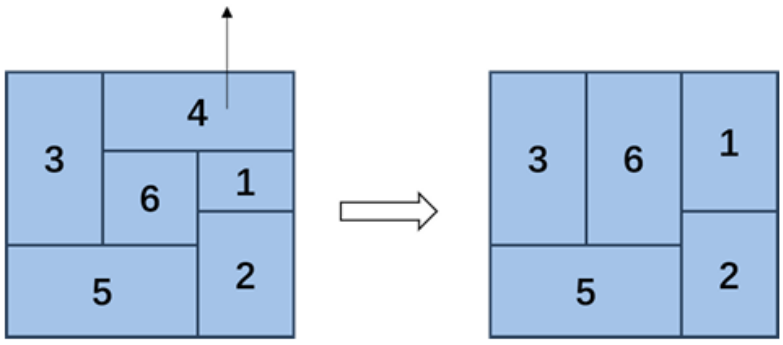

Figure 8. Angle module removal operation (Removal 4).

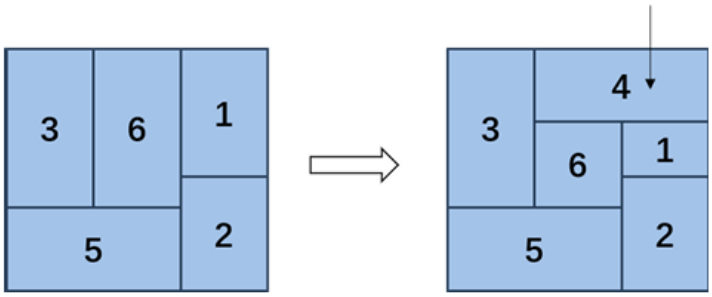

Figure 9. Corneal mass insertion procedure (Insertion 4).

Insertion operation of corneal block (**Figure 9.**): The insertion operation of corneal block is the inverse process of deletion. If the corneal block to be inserted is in the vertical direction, push down the upper border of the chip, and its length can cover a set of given t-connected horizontal line segments to obtain an area where the corneal block can be inserted. If the corneal block to be inserted is in the horizontal direction, push the vertical line segment on the right border of the chip to the left to obtain an area. Because the insertion operation is the strict inverse process of the deletion operation, the layout structure after the insertion operation is still a mosaic layout structure.

The angular block sequence representation method consists of a triplet (S, L, T), where S, L, and T describe the process of continuously removing angular blocks from a layout. When removing an angular block, the block name, block direction, and the corresponding T^i are recorded, forming the sequence $\{T^N, T^{N-1}, ..., T^2\}$. Once all the blocks are removed, the sequence is reversed to obtain the block name sequence S, the block direction sequence L, and the concatenated binary string $\{T^2, T^3, ..., T^N\}$, resulting in (S, L, T). For example, the angular block layout in **Figure 10**, with seven blocks a, b, c, d, e, f, g, can be represented using the angular block sequence representation method as S=(abcdefg), L=(001010), T=(0010011010).

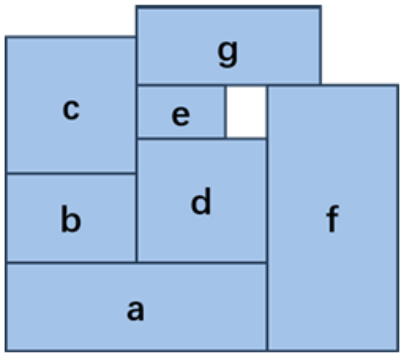

Figure 10. Corneal block sequence and sequence layout.

Corneal block sequence representation has the ability to represent the topological relationship between modules, and its solution space is small, and the complexity of codec is relatively low. However, this representation is only applicable to the Mosaic structure of the layout, there are certain limitations, cannot ensure that each corneal block sequence has a corresponding Mosaic structure of the layout.

3.3. Sequence Pair

The sequence pair (SP) representation can represent the general structure of a layout. It consists of two sequences (S+, S−), each of which contains n elements, where each element represents a module. The SP can

describe the geometric relationships between non-overlapping modules, and it imposes the following relationships on each pair of modules:

$$(<\cdots,a,\cdots,b,\cdots>;<\cdots,a,\cdots,b,\cdots>) \rightarrow \ a \text{ is on the left of } b$$

$$(<\cdots,a,\cdots,b,\cdots>;<\cdots,b,\cdots,a,\cdots>) \rightarrow \ a \text{ is above } b$$

Each pair of modules constrains each other horizontally or vertically, these constraints make the modules move as far left and down as possible in the actual placement, while satisfying the topological relationships imposed by the sequence pairs. Given a set of sequence pairs, first construct the horizontal constraint graph and the vertical constraint graph, and then get the actual placement by calculating the longest path. For example, the placement of seven modules a, b, c, d, e, f, g in **Figure 10** can be represented by the sequence pair (cbgedaf, abcdefg).

Sequence pair representation is a method of layout planning that can accurately represent the topological relationship between modules. As a general structure of layout planning method, compared with other notation, it has a wide applicability, can represent any type of layout. In addition, the perturbation operation of the representation is very simple, and only the position of the module in the sequence pair can be changed to achieve the perturbation. However, its solution space is large, and the same layout may have multiple corresponding sequence pairs.

3.4. B*-tree notation

B*-tree is based on an ordered binary tree and is used to represent the layout of a compact structure. Each node in the tree corresponds to a module, the root node represents the module at the bottom left of the layout, starting from the root node, the left subnode is defined first, and then the right subnode is defined, the nearest unvisited module on the right side of each module is the left subnode in the B*-tree, and the nearest unvisited module above the module is the right subnode in the B*-tree. As shown in **Figure 11**, b^3 is located at the bottom left of the layout, which is the root node of B*-tree, and b^5 and b^2 are adjacent to b^3 and are located on the right and top of b^3 respectively, so b^5 and b^2 are the left and right subnodes of b^3 in B*-tree, respectively. Given a B*-tree, the specific position of the module can be determined by depth-first traversal, when module b^A is placed in the position of (x^A, y^A), its left subnode module b^B in the B*-tree is in the x-axis coordinate $x^B = x^A + w^A$, where w^A is the width of module b^A, and y^B takes the minimum value to avoid overlapping with the placed module. After recursively returning from module b^B, the x-axis coordinates of module b^C in the right subnode of module b^C in B*-tree $x^C = x^A$, and y^C takes the minimum value of the overlap between the other surface and the placed module.

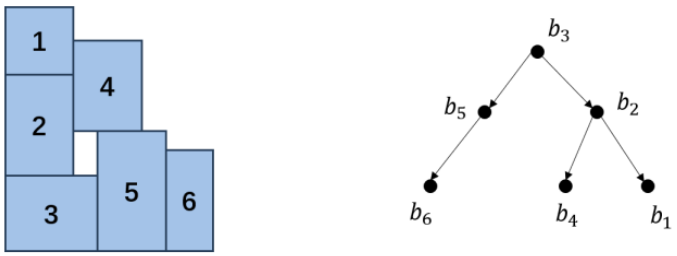

Figure 11. The actual layout and its B*-tree representation.

B*-tree is a solution with small space, and for any valid layout, there is a unique B*-tree representation. However, B*-tree can only represent compact structures, and there are certain limitations when dealing with other types of layouts, which may overlook some excellent layouts. In addition, compared with relative sequence pairs, the perturbation operation of B*-tree is relatively complex, which may be unfavorable to the implementation of the algorithm.

4. Meta-heuristic algorithms

Because the floor planning problems of both 2D integrated circuits and 3D integrated circuits are NP-hard problems, researchers usually use meta-heuristic algorithms, such as simulated annealing algorithm, to solve the floor planning problem. Next, we will introduce some commonly used meta-heuristic algorithms in floor planning problem.

4.1. Simulated annealing algorithm

Simulated Annealing (SA) algorithm is one of the earliest and most popular algorithms used in various stages of VLSI physical design. The floor planning algorithm based on simulated annealing is easy to implement. Many researchers have achieved good results in floor planning problems by combining sequence pairs, cornea block sequences, B * -tree and other floor planning representation methods.

The core idea of simulated annealing algorithm is to gradually reduce the energy of the system by simulating the solid heat annealing process, in order to expect to find a better solution in the search space. The algorithm simulates the process of solid matter from high temperature to low temperature, allowing large oscillations and accepting the probability of inferior solutions at high temperatures, and reducing the amplitude of oscillations and the probability of accepting inferior solutions at low temperatures.

The specific steps of the simulated annealing algorithm are as follows:

Step one: Initialization. At the beginning of the algorithm, set the initial temperature as the starting point of annealing, and randomly set an initial solution as the current solution.

Step two: generate a neighborhood solution by perturbation, and then in each step, generate a neighborhood solution by applying a random perturbation to the current solution, and calculate the corresponding objective function value.

Step three: Sample neighborhood solutions according to the Metropolis rule. If the objective function value of the neighborhood solution is better, that is, closer to the optimal solution, then the neighborhood solution is taken as the current solution; if the objective function value of the current solution is better, then the probability of accepting the neighborhood solution is calculated according to the Metropolis rule, and whether to accept the neighborhood solution as the current solution is determined according to the probability. The probability calculation formula is as follows:

$$P(s,s',t_k)=\begin{cases}1, & if\ f(s')\le f(s)\\ \exp\left(\dfrac{-\left(f(s')-f(s)\right)}{t_k}\right), & if\ f(s')>f(s)\end{cases}$$

where f(s) is the objective function, t^k is the temperature of the kth iteration, and s and s′ are the current solution

and neighborhood solution, respectively.

Step four: Temperature update, gradually reduce the temperature according to the preset cooling schedule. Common cooling methods include linear cooling and exponential cooling.

Step five: Iterative repetition, repeat steps two to four until the temperature drops to the set termination temperature. At each iteration, an attempt is made to find a better solution in the search space, and as the temperature decreases, the probability of accepting a worse solution decreases gradually.

Simulated annealing algorithm is widely used in optimization problems in many fields, it has better global search ability and adaptability, and has certain advantages in solving complex problems and non-convex problems. The key lies in the probability of accepting inferior solutions and the setting of temperature, reasonable configuration can balance the ability of global search and local search, avoid falling into local optimal solution. In addition, the convergence speed of the algorithm is also related to the characteristics of the problem itself, so it is necessary to carefully adjust the parameters in practical applications to obtain better results.

4.2. Genetic algorithms

The Genetic Algorithm (GA), developed from Darwin's theory of evolution, is an optimization algorithm that simulates the process of natural evolution. It finds optimal solutions to problems by imitating genetic operations such as selection, crossover, and mutation. The execution process of the algorithm typically includes the following steps:

1. Initialization of the Population: Randomly generate a set of initial individuals, known as the population. Each individual represents a solution to the problem and is associated with a chromosome composed of genes.

2. Fitness Evaluation: Using a specific fitness function, evaluate each individual in the population and calculate its fitness. The fitness value reflects the quality of the solution encoded by the individual, often defined as the value of the objective function being optimized.

3. Selection: Select individuals from the population based on their fitness values. Guided by the principle of natural selection, individuals with higher fitness are more likely to be chosen as parents.

4. Crossover: From the parent individuals, pick pairs and perform genetic crossover operations to produce new offspring. This process involves exchanging genetic segments between the parents to create new individuals.

5. Mutation: Introduce mutations into the individuals to create new genetic variations. Mutation simulates natural genetic changes and increases the diversity of the population by altering some genes of the individuals.

6. Population Update: Combine the parent and offspring individuals to form a new generation. Common strategies such as elitism or roulette-wheel selection may be used to determine which individuals will be passed on to the next generation.

7. Iteration: Repeat Steps 2 through 6 until the stopping criterion is met, for example, when a predefined number of iterations is reached or when a satisfactory fitness threshold is achieved.

Through continuous selection, crossover and mutation operations, genetic algorithm can search the solution space of the problem and gradually approach the optimal solution. Because of its parallelism and global search ability, genetic algorithm has a good effect in solving complex optimization problems. But the performance of genetic algorithm is also affected by the characteristics of the problem and the selection of algorithm parameters, so reasonable adjustment and optimization should be carried out in practice.

5. Conclusion

In conclusion, the evolution of integrated circuit (IC) design methodologies has led to innovative approaches to tackle the growing complexity and demands of modern electronics. This review highlights key advancements, including the use of novel representation techniques and metaheuristic algorithms such as simulated annealing and genetic algorithms, to optimize the design process. The integration of machine learning models and hybrid optimization strategies offers promising directions for addressing scalability and performance challenges. As IC design continues to advance, interdisciplinary approaches leveraging reinforcement learning, deep learning, and optimization theory will play a pivotal role in shaping the future of this field, enabling more efficient, reliable, and powerful circuits to meet the needs of emerging technologies.

References

[1] Nayak P. A study of technology roadmap for application-specific integrated circuit[D]. Rice University, 2021.

[2] Shalf J. The future of computing beyond Moore's Law[J]. Philosophical Transactions of the Royal Society A, 2020, 378(2166): 20190061.

[3] Leiserson C E, Thompson N C, Emer J S, et al. There's plenty of room at the Top: What will drive computer performance after Moore's law?[J]. Science, 2020, 368(6495): eaam9744.

[4] Burg D, Ausubel J H. Moore's Law revisited through Intel chip density[J]. PloS one, 2021, 16(8): e0256245.

[5] Ding B, Zhang Z H, Gong L, et al. A novel thermal management scheme for 3D-IC chips with multi-cores and high power density[J]. Applied thermal engineering, 2020, 168: 114832.

[6] Huang C Y, Dewey G, Mannebach E, et al. 3-D self-aligned stacked NMOS-on-PMOS nanoribbon transistors for continued Moore's law scaling[C]//2020 IEEE International Electron Devices Meeting (IEDM). IEEE, 2020: 20.6. 1-20.6. 4.

[7] Kang K, Benini L, De Micheli G. Cost-effective design of mesh-of-tree interconnect for multicore clusters with 3-D stacked L2 scratchpad memory[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2014, 23(9): 1828-1841.

[8] Ben Abdallah A, Ben Abdallah A. 3D integration technology for multicore systems on-chip[J]. Advanced Multicore Systems-On-Chip: Architecture, On-Chip Network, Design, 2017: 175-199.

[9] Hu X, Stow D, Xie Y. Die stacking is happening[J]. IEEE micro, 2018, 38(1): 22-28.

[10] Ghaderi Z, Alqahtani A, Bagherzadeh N. AROMa: aging-aware deadlock-free adaptive routing algorithm and online monitoring in 3D NoCs[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 29(4): 772-788.

[11] J. Cong, C. Liu, and G. Luo, "Quantitative studies of impact of 3D IC design on repeater usage," in Proc. Int. VLSI/ULSI Multilevel Interconnection Conf., 2008, pp. 344–348.

[12] Kim D H, Athikulwongse K, Healy M B, et al. Design and analysis of 3D-MAPS (3D massively parallel processor with stacked memory)[J]. IEEE Transactions on Computers, 2013, 64(1): 112 125.