WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Optimization of Physical Domain Design For VLSI

***Pratiksha Tiwari, Dr. Shubham Dubey***

**Department of Electronics And Communication Engineering, Shri Krishna University Chhatarpur , (MP)**

**ABSTRACT**

Designing complex VLSI systems with millions of transistors requires dividing the overall circuit into smaller, manageable subcircuits through a process called partitioning. This step is essential in reducing design complexity and time. Partitioning aims to minimize interconnections (Mincut), reduce latency, optimize power consumption by managing sleep time, and analyze algorithmic efficiency.

After partitioning, details such as the shape, orientation, area, and number of pins for each block are defined. These blocks are then arranged on the chip surface in a process called floorplanning, where the goal is to minimize total area and wire-length, which directly impacts power and delay.

Since both partitioning and floorplanning are NP-hard problems, heuristic optimization methods are employed. This research utilizes Particle Swarm Optimization (PSO), a hybrid PSO–Ant Colony Optimization (PS-ACO), and an enhanced Parallel PSO (PPSO) algorithm. These are applied to achieve optimal partitioning and efficient layout, especially for advanced 3D ICs, where area, wire-length, and Through Silicon Vias (TSVs) are optimized.

Simulations using ISPD'98, MCNC, and GSRC benchmarks were conducted in MATLAB on a 2.40 GHz Intel system. The results show that the proposed methods improve efficiency in layout design and are suitable for future large-scale VLSI implementations.

**Keywords:** Partitioning, Floorplanning, Placement, PSO, ACO, Cutsize, Sleeptime, NP-Hard, Sequence pair (SP), LCS, ISPD, MCNC, GSRC

## INTRODUCTION-

The remarkable advancements in integrated circuit (IC) technology have largely been driven by the automation of various stages in the design and fabrication of VLSI (Very Large Scale Integration) circuits. These circuits are built by layering different materials on a silicon base, known as a wafer, to form numerous electronic components.

At the core of IC development is the transformation of circuit descriptions into a geometric representation, commonly referred to as the layout. This layout consists of multiple layers of two-dimensional geometric shapes that represent different parts of the circuit. Before manufacturing, the layout undergoes thorough verification to ensure compliance with all design rules and constraints. Once verified, the layout is converted into a set of design files that capture all its geometric and structural details.

An optical pattern generator processes these design files to produce pattern generator outputs, which are subsequently used to fabricate masks. These masks play a critical role in the photolithographic steps of semiconductor manufacturing, defining how material is deposited or etched on the wafer. For accurate production of ICs, precise spatial and geometrical data of all components is essential.

The process of transforming an electronic circuit's specification into a physical layout on silicon is known as physical design. As the industry continues to push the limits of miniaturization—with feature sizes as small as 14 nanometers or less—this stage has become increasingly complex and sensitive to errors. For example, Graphcore, a leading AI chip designer based in the UK, has managed to integrate up to 60 billion transistors and more than 1,500 processing cores onto a single chip.

Due to the intricacy of the design rules and the precision required in modern semiconductor manufacturing, Computer-Aided Design (CAD) tools are indispensable throughout the physical design process. The objective of physical design is to determine an optimal placement of components within a two-dimensional (or occasionally three-dimensional) space and to establish efficient interconnections among them. This must be done in a way that satisfies performance requirements, optimizes chip area, and adheres to manufacturing constraints.

Since wafer space is limited and expensive, physical design algorithms are tasked with maximizing area utilization to reduce production costs and enhance yield. Moreover, the physical arrangement of components directly affects the chip's performance. Therefore, layout generation must strike a balance between functionality, manufacturability, and performance metrics. Scalability and speed are also crucial—efficient algorithms not only reduce the time required to complete designs but also allow designers to iterate and improve layouts more freely.

## REVIEW LITERATURE

VLSI design has brought the power of the mainframe computer to the laptop, from its beginnings in the early 1950s to the fabrication of circuits with millions of components today. The development of advanced design tools and software has enabled this enormous growth in the field of VLSI design. VLSI design tools must not only be computationally efficient, but also perform close to optimally to deal with the complexity of millions of components. In the physical domain design, Partitioning and Floorplanning are the two major steps that determine the overall system performance. Several algorithms that address the partitioning and Floorplanning issues are discussed in this chapter.

**Literature review on Partitioning**

Any complicated system must be decomposed into a group of smaller subsystems to be designed efficiently [1]. To speed up the design process, each subsystem may thus be created separately and concurrently. Partitioning is the term for the breakdown process [11]. Within three broad criteria, partitioning efficiency may be improved. First, the system must be thoroughly disassembled such that the system's original functionality is preserved. Second, during this process, an interface definition is created, to connect all of the subsystems. The interface links between any two subsystems should be minimal. Finally, the decomposition method should be simple and efficient, so that the decomposition time is a minimal percentage of the overall design time [1, 12]. There are several algorithms have been reported to achieve different partitioning goals. Some prominent algorithms presented are described below:Kernighan & Lin [6] have presented one of the earliest techniques to deal with the partitioning problem. They have presented a heuristic method known as KL-Algorithm, for partitioning arbitrary graphs. The KL algorithm is both effective in finding optimal partitions, as well as fast enough to deal with practically large problems. The KL-algorithm is a local optimization algorithm and has limited ability to get out of local minima. Fiduccia and Mattheyses [11] presented a modified version of the KL algorithm. In the FM Algorithm, in a single motion, one vertex is moved across the cut. They have used their cutsize notion to hypergraphs.

Glasser & Hoyte [14] has examined the problem of optimally sizing the transistors in a digital MOS VLSI. They have developed a micro model and new theorem on optimally sizing of the transistor in a critical path to reduce the delay and power dissipation. They have discussed the design automation procedure to perform the required optimization. Sanchis [15], presented an adoption of a two-block iterative improvement partitioning technique to multiple block network partitioning procedures. As per their work, the total number of ideal partitions depends primarily on the total number of circuit components and degree of distribution of the network and it fluctuates minimally with network size. A novel hypergraphs partitioning method based on a multilevel paradigm was proposed by Karypi et al. [16]. A succession of progressively abrasive hypergraphs was built in the multilevel paradigm. Gradually projecting and refining the bisection to the next finer hypergraph level, the smallest hypergraph's bisection was computed and used to generate the original hypergraph's bisection.

## METHODOLOGY-

**1. Partitioning of VLSI Circuits for mincut, delay & sleep time optimization.**

Circuit partitioning plays a crucial role in the physical design phase, where a complex circuit is divided into smaller, more manageable modules. This segmentation must adhere to various design constraints. The main goals of partitioning typically include minimizing the number of connections (mincut) between different partitions, reducing communication delays, extending the periods during which modules remain inactive to save power, and ensuring that the partitioning algorithm operates efficiently.

This chapter presented an approach that aims to balance these objectives using the Particle Swarm Optimization (PSO) technique. To evaluate the effectiveness of the proposed method, standard benchmark circuits from the ISPD’98 suite were utilized, offering a reliable basis for performance comparison and analysis..

### *1.1 Problem Definition*

A partitioning problem may be divided into two types: bi partitioning and multi partitioning.

### *1.2. Solution Methodology*

**Solution Methodology**

A PSO method for multimodal continuous function optimization is proposed in this Chapter. PSO is used for global optimization by updating particle locations to achieve quick convergence .The steps for using the PSO method to solve the partitioning problem are as follows;

**Steps for calculating time complexity of Mincut:**

Time complexity (in terms of how many times each statement is being executed) can be calculated as

- Initializing ‘cutsize’ will execute only once.
- First ‘for loop’ will execute length (partiiton1) times i.e. N/2 times.

- Second ‘for loop’ will execute length (partiiton2) times i.e. N/2 times.
- ‘If’ statement will execute only when there is an interconnection between partition1 and partition2 i.e. 3N times.
- Cutsize increment will take only one operation.
- Time complexity of mincut module is $1+3N^3/4$. Steps for calculating time complexity of Delay:
- . **Steps for calculating Time complexity of Area:**
- Area file will be given as an input only once.
- Its path name and file name will be stored in ‘file’ only once.
- fopen will open the stored area file.
- Each and every character and digit is read from that file.
- Size of area_array is calculated once.
- 0s are assigned to area_matrix.
- ixx is assigned a value once.
- For loop’ will execute N (N-2) times.
- Time complexity of read_area module is N (N-2) +6.

**Steps for calculating time complexity of Initial position of a particle:**

- ssigning array ‘m’ to an array of zeros will execute once.
- Initialization of Partition1 and partition2 to zero will be of one operation.
- For loop’ will execute 3N times.
- Assigning x, y to ‘0’ will take only one operation, x, y are being used for indexing of partition1 and partition2.
- ‘For loop’ to differentiate the partitions, will execute N (N/2) (N/2) times.
- Time complexity of initial_position module is $3+9N+2+N (N^2/4) + (N-7)$.
- Length of partition1 and partition2 will be executed one time.
- (N-7) nodes will be exchanged to have balanced partitions

**Steps for calculating Time complexity of node swapping:**

- ‘Node_to_swap’ value is stored in part1 and part2. Corresponding to this value, node form position1 and posiiton2 is stored in temp1 and temp2 respectively for temporary storage.
- Then, corresponding nodes in both positions are exchanged.
- After that, new positions are stored in newpartition1 and newpartition2.
- Total time complexity of nodes_swapping is 8.

**Steps for calculating Time complexity of sleep_matrix:**

- Calculation of node execute once.
- Assignment of clock will execute once.
- St Matrix initialization will be of one operation.
- ‘For’ loop to calculate the st matrix will take $N \times M \times M$ time.
- Time complexity of sleep_matrix module is$1 + 1 + 1 + N \times M \times M = 3 + NM^2$.

**Steps for calculating Time complexity of sleep_test**

- Calculation of nodes execute once.
- Calculation of length of partition1 execute once.
- Calculation of length of partition2 execute once.

- 'for' loop to calculate sleep time for partition1 will take N × M times.
- Similarly, 'for' loop to calculate sleep time for partition2 will take N × M times.
- Total time complexity of sleep_test module is 1 + 1 + 1 + M × N + M × N = 3 + 2MN

**Summary**

This chapter introduces a novel method for dividing a larger system into smaller subsystems using a bipartitioning approach. The Particle Swarm Optimization (PSO) technique is employed effectively to optimize multiple objectives simultaneously. When both inter-partition delay and sleep time are considered in a multi-objective fitness function—with equal weighting assigned to sleep time and mincut—the delay and mincut values tend to decrease, while sleep duration increases, eventually stabilizing as the number of iterations increases. As shown in Table 3.1, the proposed method yields notable improvements in key performance metrics: mincut is improved by an average of 32.47%, delay is reduced by 54.76%, sleep time is extended by 60.14%, and power consumption is reduced by 6.58% on average. The computational complexity of the method is approximately N^3 \cdot NP \cdot M.

***2.Floorplanning for area optimization using P-PSO technique***

1. **The floorplanning techniques**
2. **Sequence-pair**

***3. Experimental results***

To evaluate the proposed strategy, the MCNC and GSRC benchmarks were utilized. The MCNC set includes five soft and five hard modules, while the GSRC set consists of six of each type. Soft modules are flexible in shape within a given area and aspect ratio range, whereas hard modules have fixed shapes and only allow rotation. Sequence pair representation was applied for handling hard modules in our experiments. The performance of the proposed P-PSO algorithm on MCNC benchmarks was compared against methods in [28, 35, 51, 52, 53, 86], focusing on area optimization. For GSRC benchmarks, comparisons were made with techniques in [29, 54, 55, 87]. Notably, [28] integrates parquet with sequence pair for fixed-outline floorplanning. The commercial tool BloBB [29] supports both slicing and non-slicing designs. In [35], B*-tree with Simulated Annealing (SA) was used for module placement. PSO-based slicing was explored in [51], while [52] adopted Hybrid SA for non-slicing B*-tree layouts. Sequence pair-based fast alignment and clustering are addressed in [54], and [55] combines Ant System with SA for 3D floorplanning. DOTFloor in [53] offers a time-efficient SA-based approach, [86] uses Harmony Search inspired by musical improvisation, and [87] investigates the Lion Optimization Algorithm for area efficiency.

| **Algorithm** | | **SP (Parquet- 1) [28]** | **SA with B*-tree[35]** | **PSO[51]** | **HSA[52]** | **DOTFloor [53]** | **PSO- GA [86]** | **Ours (P-PSO)** |
|---|---|---|---|---|---|---|---|---|
| apte | Area ($mm^2$) | 47.07 | 47.31 | 47.31 | 46.9 | 47.40 | 46.92 | 46.74 |
| | DS(%) | 1.1 | 1.6 | 1.6 | 0.7 | 1.86 | 0.7 | 0.4 |
| xerox | Area ($mm^2$) | 19.83 | 20.12 | 20.2 | 20.01 | 20.16 | 20.08 | 19.52 |
| | DS (%) | 2.42 | 3.85 | 4.21 | 3.3 | 4.02 | 3.63 | 0.87 |
| hp | Area ($mm^2$) | 9.14 | 8.46 | 9.5 | 9.01 | 9.71 | 9.2 | 8.91 |
| | DS (%) | 11.6 | 4.47 | 14.95 | 10.32 | 16.79 | 12.17 | 9.32 |
| ami33 | Area ($mm^2$) | 1.19 | 1.22 | 1.28 | 1.2 | 1.64 | 1.25 | 1.17 |
| | DS (%) | 3.36 | 5.69 | 10.16 | 4.17 | 29.88 | 8 | 1.7 |
| ami49 | Area ($mm^2$) | 37.27 | 37.45 | 38.8 | 36.48 | 47.02 | 38.31 | 36.33 |
| | DS (%) | 4.91 | 5.34 | 8.66 | 2.85 | 24.63 | 7.49 | 2.45 |

**Table 4.3:** Area optimization results for MCNC Benchmark Circuits (in $mm^2$)

**DS (%): Percentage of dead space in the total area occupied by the floorpla**

| **Algorithm** | | **Hierarchical [29]** | **SA with Cluster Constraints [54]** | **Hybrid ACO-SA[55]** | **LOA [87]** | **Ours (P-PSO)** |
|---|---|---|---|---|---|---|
| n10 | Area ($mm^2$) | ---- | ---- | ---- | 2.3 | 2.24 |
| | DS (%) | ---- | ---- | ---- | 3.91 | 1.34 |
| n30 | Area ($mm^2$) | ---- | ---- | ---- | ---- | 2..18 |
| | DS (%) | ---- | ---- | ---- | ---- | 4.59 |
| n50 | Area ($mm^2$) | 2.06 | ---- | ---- | 2.15 | 2.03 |
| | DS (%) | 11.65 | ---- | ---- | 15.35 | 10.34 |
| n100 | Area ($mm^2$) | 1.92 | 1.88 | 1.8 | 2.08 | 1.84 |
| | DS (%) | 8.38 | 5.41 | 0.56 | 11.17 | 2.23 |
| n200 | Area ($mm^2$) | 1.91 | 1.85 | 2.0 | 1.97 | 1.79 |
| | DS (%) | 6.77 | 4.79 | 12.5 | 13.94 | 2.72 |
| n300 | Area ($mm^2$) | 2.97 | 2.93 | 3.4 | 3.07 | 2.81 |
| | DS (%) | 8.08 | 6.83 | 19.71 | 11.07 | 2.85 |

**Table 4.4:** Area optimization results for GSRC Benchmark Circuits (in $mm^2$)

DS (%): Percentage of dead space in the total area occupied by the floorplan

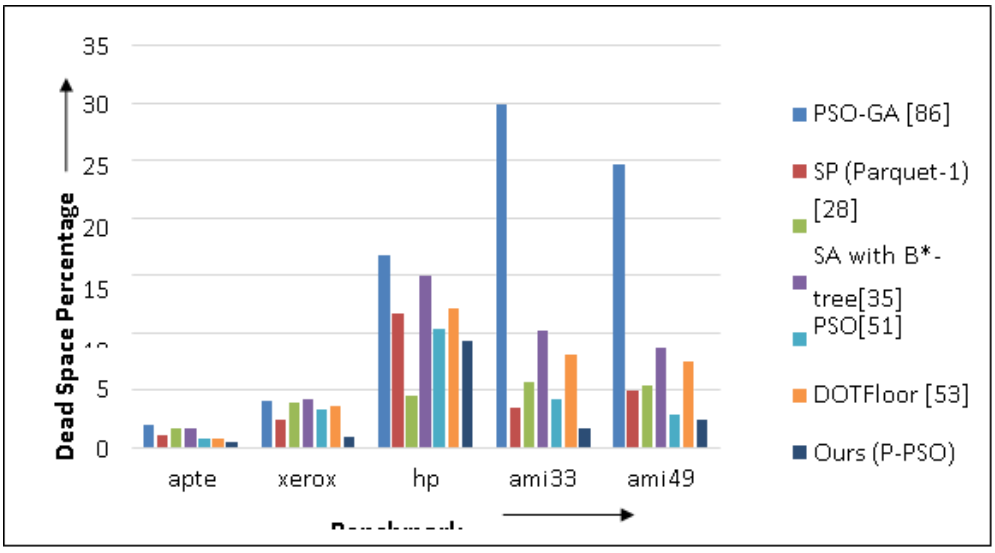


**Figure 4.3:** Dead Space Comparison of MCNC benchmark circuits with other algorithms

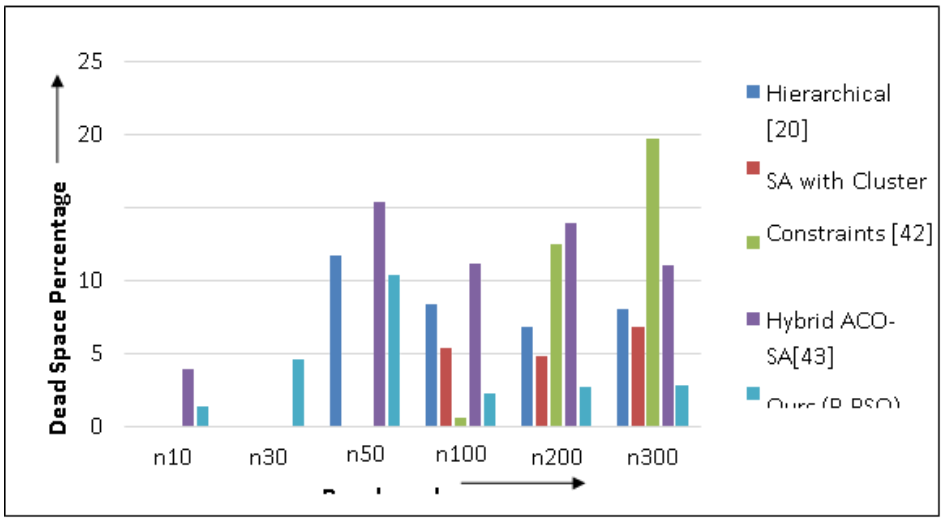


**Figure 4.4:** Dead Space Comparison of GSRC benchmark circuits with other algorithms

### *4.6. Summary*

In the presented work, we tested the proposed algorithm on the MCNC & GSRC benchmark circuits for the modern floorplanning with area constraints, based on our new Parallel-PSO and sequence pair representation. The result shows that our algorithm leads to a quicker convergence to the desired objective function. Our experimental result shows significant speedup over the classical PSO. In addition, the obtained results occupy lesser area as compared to the previous algorithms suggested by various authors. The results obtained by the proposed algorithm shows an average improvement of 5.15% on MCNC and 7.47% on GSRC benchmark circuits.

## 5. CONCLUSION

The dissertation focuses on physical domain design for VLSI circuits, addressing partitioning, floorplanning, and placement for both 2D and 3D ICs. It proposes algorithms for optimizing key design parameters.

In Chapter 3, a PSO-based bi-partitioning approach is introduced to minimize cutsize and delay while maximizing sleep time. The PSO method is fast and effective for large circuits, achieving a balanced tradeoff among objectives.

Chapter 4 presents Parallel-PSO (P-PSO), which overcomes PSO's drawbacks—such as loss of diversity and premature convergence—by enabling multi-swarm exploration and dynamic dimensionality. P-PSO effectively optimizes area in Sequence-Pair-based floorplanning

**REFERENCES**

[1] N. Sherwani, "Algorithms for VLSI physical Design and Automation," 3rd Edition, Springer (India) Private Limited, New Delhi 2005.

[2] G. E. Moore, "Cramming More Components Onto Integrated Circuits," in Proceedings of the IEEE, vol. 86, no. 1, pp. 82-85, Jan. 1998.

[3] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Hypergraph partitioning with fixed vertices [VLSI CAD]," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, issue 2, pp. 267-272, 2000.

[4] P. Ghafari, E. Mirhadi, M. Anis, A. Areibi and M. Elmasry, "A low-power partitioning methodology by maximizing sleep time and minimizing cut nets," In proc. Fifth International Workshop on System-on-Chip for Real-Time Applications (IWSOC'05), Banff, AB, Canada, pp. 368-371, July 2005.

[5] Stephen Coe, Shawki Areibi and Medhat Moussa, "A hardware Memetic accelerator for VLSI circuit partitioning," Comput. Electr. Eng. Vol. 33, No. 4, pp. 233–248, 2007.

[6] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," The Bell System Technical Journal, vol. 49, no. 2, pp. 291-307, 1970,

[7] Sandeep Singh Gill, Dr. Rajeevan Chandel and Dr. Ashwani Chandel, "Genetic Algorithm Based Approach To Circuit Partitioning," International Journal of Computer and Electrical Engineering vol. 2, no. 2, pp. 196-202, 2010.

[8] J. Lin and Z. Hung, "SKB-Tree: A Fixed-Outline Driven Representation for Modern Floorplanning Problems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 3, pp. 473-484, 2012.

[9] M. Tsai, T. Wang and T. Hwang, "Through-Silicon Via Planning in 3-D Floorplanning," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 8, pp. 1448-1457, 2011.

[10] Sadiq M. Sait, Feras Chikh Oughali, Mohammed Al-Asli, "Design partitioning and layer assignment for 3D integrated circuits using tabu search and simulated annealing," Journal of Applied Research and Technology, Volume 14, Issue 1, pp. 67-76, 2016.

[11] C. J. Alpert, A. E. Caldwell, A. B. Kahng and I. L. Markov, "Hypergraph partitioning with fixed vertices [VLSI CAD]," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 2, pp. 267-272, 2000.

[12] A. H. Farrahi and M. Sarrafzadeh, "Geo_Part: A System Partitioning Algorithm to Maximize Sleep Time," Digest of the Technical Papers of ICCAD, San Jose, CA, 1995.

[13] A. H. Farrahi and M. Sarrafzadeh, "System partitioning to maximize sleep time," In Proc. of IEEE International Conference on Computer Aided Design (ICCAD), San Jose, CA, USA, pp. 452-455, Dec. 1995.

[14] C. Ababei, S. Navaratnasothie, K. Bazargan and G. Karypis, "Multi-objective circuit partitioning for cutsize and path-based delay minimization," In Proc. IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, USA , pp. 181- 185, Nov. 2002.