**OPTIMIZATION OF PHYSICAL DOMAIN DESIGN FOR VLSI**

A Thesis

*SUBMITTED TO*

**BIRLA INSTITUTE OF TECHNOLOGY**

****

**FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY**

**in**

**ENGINEERING**

**By**

**ATUL PRAKASH, M. TECH**

**PHD/EC/10053/2013**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING**

**BIRLA INSTITUTE OF TECHNOLOGY**

**MESRA: RANCHI, INDIA**

**2021**

**DECLARATION**

I certify that:

1. The workcontainedinthethesisisoriginalandhasbeen done by myself under the general supervision of my Guide.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have followed the guidelines provided by the Institute in writing the thesis.
4. I have conformed to the norms and guidelines given in the EthicalCode of Conduct of the Institute.
5. Whenever I have used materials (data,theoretical analysis and text) from other sources, I have given due credit to them by citing them inthe text of the thesis and giving their detailsin thereferences.
6. Whenever I have quoted written materials from other sources, I have put them underquotation marks and given due credit to the sources by citing them and giving required details in the references.

(Atul Prakash)

**APPROVAL OF THE GUIDE**

Recommended that the thesis entitled ‘Optimization of Physical Domain Design for VLSI’ prepared by Mr. Atul Prakash, under my supervision and guidance be accepted as part of fulfilling the requirements for award of the degree of Doctor of philosophy.

To the best of my knowledge, the contents of this thesis did not form a basis for the award of any previous degree to anybody else.

Date:

(Dr. Rajesh KumarLal)

Research supervisor

Associate Professor

Dep. of Electronics and Communication Engg.

BIT, Mesra, Ranchi-835215

***Dedicated***

***to***

***My Parents, Wife & Son***

**Acknowledgement**

I received a lot of help and support while carrying out the work presented in this dissertation from many people. I did want to use this time to express my profound gratitude to everyone who has helped me along the way, particularly Birla Institute of Technology, Mesra, under whose auspices I was able to complete my project.

I would take the opportunity to express a deep sense of gratitude and thank profusely my supervisor Dr. Rajesh Kumar Lal, Dept. of ECE, BIT Mesra, Ranchi for his guidance in organizing and writing this thesis and giving me the opportunity to pursue my goals in such an interesting area. Without his wise counsel,advice and academic support, it would have been impossible to complete this thesis.

I am indebted to all other faculty members of ECE department, BIT Mesra, Ranchi, for their intellectual support throughout my stay at BIRLAINSTITUTE OF TECHNOLOGY, Ranchi.

I acknowledge the pivotal role played by my Ph.D. DC members Prof. S. K. Ghorai, Dr. Aminul Islam, Dr. Abhijit Mustafiand, Dr. S. K. Mishra for their valuable suggestions during my annual progress reports.

My parentsShri Harendra Kumar Dwivedi &Smt. Renu Dwivedi and my wife Kumari Priyankawere always there for me as a moral support. Their confidence in me gave me a fillip to complete this thesis work.

I also acknowledge the roles played by my old-time friend Dr. Abhishek Kumar Jha and fellow researcher &friend Mr. Saurabh Sarkar for their moral support and much needed advice during my Ph.D. course.

Above all, I am thankful to the almighty God for giving me the strength to carry out the present work.

Atul Prakash

**Abstract**

Any complicated electronic system requires a lot of planning; it has to be divided into smaller subsystems. To reduce the amount of time it takes to produce a design, each subsystem is designed separately and simultaneously. A partitioning process is used to break a large system down into smaller components.

In the circuit’s physical design process, partitioning is a critical stage. Subdividing multi-million transistor circuitsinto reasonable portions,it is sometimes required to construct sophisticated digital logic circuits. Therefore, partitioning is both a design challenge to decompose a larger system into smaller sub-systems that composes interrelated elements and an algorithmic approach for solving tough and complicated combinatorial optimization goals, such as in the case of logic or layout synthesis.

Partitioning is created with certain targets in mind, which has to be accomplished individually or accumulatively. Four goals are pursued in this dissertation: reducing the Mincut between the different partitions, decreasing the inter-partition latency, to reduce power consumption by optimizing the sleep period of the partitions and determining the different time complexities of the proposed algorithms.

Once the partitioning phase of the circuit is complete, the area occupied by each sub-circuit is determined, the different shapes and orientation of the blocks may also be defined and the number of pins (terminals) associated withevery block is also known. Additionally, the netlist, provides the interconnections among the various blocks and pins, is provided. Assigning a certain shape and orientation to a block and then placing these blocks or nodes on the layout surface and connecting their terminals in accordance with the provided netlist&completes the layout.

The blocks are placed in such a way that the chip's overall area is minimized by minimizing dead space (the space where no block is placed); additionally, the floorplanning is done in such a way that the total wire-lengthobtained by interconnecting blocks on the chip is minimized, which reduces power consumption as well as delay.

Both the partitioning and Floor-planning are NP-hard problems and can be solved by any heuristic approach. Optimization techniques used in this dissertation are Particle Swarm Optimization (PSO)and hybridized Particle Swarm-Ant Colony Optimization (PS-ACO) algorithm. In addition, an improved version of PSO, the Parallel-PSO (P-PSO) is proposed. The algorithms are developed to implement PSO, technique for circuit partitioning finding the global best solution for determining lowestmincut optimizing the sleep time and reducing the delay objectives. The second part of the dissertation presents the Floor-planning technique using PS-ACO and Parallel-PSO to optimize the total area occupied along with minimizing the wire-length. The 3D-IC design presented in the last part, simultaneously optimizes the parameters such as the area occupancy, wirelength and Through Silicon Vias (TSVs) using the parallel-PSO. The results of the partitioning, floorplanning, placement, and 3D-IC design algorithms are compared with those of the other algorithms.

The partitioning problem is tested and simulated using Netlist files in the ISPD'98 standard reference format, while floor-planning simulation and testing is performed using MCNC and GSRC circuits. The algorithm simulation is done in MATLAB software, on a 2.40 GHz Intel Computer on Windows10 platform. These methods can be applied on multi-million transistor circuits in the near future, incorporating CPU running time as an additional parameter.

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

**TABLE OF CONTENTS**

**Page No.**

**List of figures vi**

**List of Tables xi**

**Abbreviations xiii**

**Chapter 1: Introduction 1**

1. Introduction to VLSI Physical design 1
2. Overview of partitioning 3
3. Circuit Representation 8
   1. Graph Representation 9
   2. Hypergraph representation 9
4. Overview of Floor-planning 11
5. Overview of Placement 13
6. Motivation 14
7. Objectives 16
8. Organization of the Thesis 16

**Chapter 2: Literature Review 19**

1. Literature Review on Partitioning 19
2. Literature review of Floor-planning & Placement 23
3. Summary 29

**Chapter 3: Partitioning of VLSI Circuits for mincut, delay & sleep time 30**

**optimization using PSO technique**

1. Introduction 30
2. Objective Function 31
3. Problem definition 31
   1. Mincut Optimization 32
   2. Delay Minimization 33
   3. Sleep Maximization 35
   4. Combined Mincut, Delay & Sleep Time Optimization 39
4. Solution Methodology 40
5. Results & Discussion 57
6. Summary 63

**Chapter 4: Floorplanning for Area Optimization using P-PSO 64**

**technique**

1. Introduction 64
2. Problem Statement 64
3. The Floorplanning Technique 65
   1. Sequence-Pair 66
4. Present Work 69
   1. The parallel Particle Swarm Optimization 71
   2. Parameters of the proposed Algorithm 75
5. Experimental Results 75
6. The Validity of the Proposed Algorithm over Classical PSO 77
7. Summary 85

**Chapter 5: Multi Criteria optimized Placement for Area & Wirelength 86**

**optimization**

1. Introduction 86
2. Problem Formulation 86
   1. The placement Techniques 87
      1. Effect of Placement on Routability 87
3. Solution Methodology 89
   1. Particle Swarm Optimization 89
      1. PSO Algorithm & its Parameters 90
      2. Swarms & Particles 92
      3. Comparisons between PSO & other Algorithms 93
   2. Ant Colony Optimization 94
      1. Real Ants’ Behavior 94
      2. Artificial Ants 97
   3. Hybridized PSACO 97
4. Parameters of the Proposed Algorithm 103
5. Experimental Results 103
6. Validity of the Proposed Algorithm over PSO 103
7. Summary 116

**Chapter 6: Simultaneous optimization of the area, wirelength and 117 TSVs in a 3D IC design**

1. Introduction 117
2. Problem formulation 118
   1. TSVs Optimization 119
   2. Wirelength Optimization 120
   3. Combined Area, Wirelength & TSVs Optimization 121
3. The 3D IC Design Technique 122
4. Experimental Results 128
5. Summary 134

**Chapter 7: Conclusion & Future Scope 135**

1. Conclusion 135
2. Future Scope 137

**References 138**

**Appendix I: Benchmark Circuits 149**

**List of Publications 156**

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure** | **Caption** | **Page No.** |
| **Figure 1.1** | VLSI Design Cycle | 3 |
| **Figure 1.2** | Overview of circuit partitioning | 5 |
| **Figure 1.3** | Partitioning of a Circuit | 7 |
| **Figure 1.4** | Different levels of Partitioning | 8 |
| **Figure 1.5** | Graph representation of the circuit | 9 |
| **Figure 1.6** | Hypergraph Representation | 9 |
| **Figure 3.1** | Bipartitioning of a Circuit | 31 |
| **Figure 3.2** | An example of a six-terminal net based on MRST (a) MRST is used to create the entire net structure (b) The dimensions of the net enclosing region are shown by the segments | 34 |
| **Figure 3.3** | Using circuit partitioning to take advantage of sleep mode operation (a) Memory Segmentation & (b) Power Down Partitioning | 35 |
| **Figure 3.4** | Partitioning to maximize sleep time | 38 |
| **Figure 3.5** | Algorithm for finding the initial partitions | 47 |
| **Figure 3.6** | Algorithm for finding the interconnections between two partitions | 48 |
| **Figure 3.7** | Algorithm to find delay for given partitions | 50 |
| **Figure 3.8** | Flow chart to calculate sleep matrix | 51 |
| **Figure 3.9** | Flow chart to calculate the sleep time | 53 |
| **Figure 3.10** | Algorithm for swapping the nodes between partitions  No | 53 |
| **Figure 3.11** | Algorithm to create new particles | 54 |
| **Figure 3.12** | Algorithm for maximizing sleep time and reducing the cutsize& delay in Bipartitioning using PSO | 56 |
| **Figure 3.13** | Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N10\_Series | 59 |
| **Figure 3.14** | Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N20\_Series | 60 |
| **Figure 3.15** | Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N25\_Series | 60 |
| **Figure 3.16** | Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N30\_Series | 61 |
| **Figure 3.17** | Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N40\_Series | 61 |
| **Figure 3.18** | Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N50\_Series | 62 |
| **Figure 3.19** | Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N60\_Series | 62 |
| **Figure 4.1** | The sequence-pair for the specified placement is (132645, 245136) | 67 |
| **Figure 4.2** | **a** Floorplan for the Initial sequence pair (. **b** Floorplan for the sequence pair (after exchanging module 1 & 3 only in Sequence). **c** Floorplan for the sequence pair (After exchanging module 2 and 6 in both Sequences. **d** Floorplan for the sequence pair , when module 3 is rotated | 68 |
| **Figure 4.3** | Dead Space Comparison of MCNC benchmark circuits with other algorithms | 80 |
| **Figure 4.4** | Dead Space Comparison of GSRC benchmark circuits with other algorithms | 80 |
| **Figure 4.5** | Floorplan for apte (0.40% dead Space) | 82 |
| **Figure 4.6** | Floorplan for xerox (0.87% dead Space) | 83 |
| **Figure 4.7** | Floorplan for hp (8.91% dead Space) | 83 |
| **Figure 4.8** | Floorplan for ami33 (1.7% dead Space) | 83 |
| **Figure 4.9** | Floorplan for ami49 (2.45% dead Space) | 83 |
| **Figure 4.10** | Floorplan for n10 (1.34% dead Space) | 84 |
| **Figure 4.11** | Floorplan for n30 (4.59% dead Space) | 84 |
| **Figure 4.12** | Floorplan for n50 (10.34% dead Space) | 84 |
| **Figure 4.13** | Floorplan for n100 (2.23% dead Space) | 84 |
| **Figure 4.14** | Floorplan for n200 (2.72% dead Space) | 85 |
| **Figure 4.15** | Floorplan for n300 (2.85% dead Space) | 85 |
| **Figure 5.1** | a collection of blocks and interconnected networks | 88 |
| **Figure 5.2** | two distinct solutions to the same problem | 88 |
| **Figure 5.3** | Particle Swarm optimization | 90 |
| **Figure 5.4** | Real ants on their way from the nest to the food source | 94 |
| **Figure 5.5** | When an obstruction arises on the path, the ants have an equal chance of turning left or right | 95 |
| **Figure 5.6** | Pheromone deposited more quickly on the shorter path | 95 |
| **Figure 5.7** | All the ants are selecting the shortest path. | 95 |
| **Figure 5.8** | Algorithm for minimizing area and wirelength using PSO-ACO | 102 |
| **Figure 5.9** | Comparison of Area of MCNC benchmark circuits with other algorithms | 107 |
| **Figure 5.10** | Comparison of Wirelength of MCNC benchmark circuits with other algorithms | 107 |
| **Figure 5.11** | Comparison of Area of GSRC benchmark circuits with other algorithms | 108 |
| **Figure 5.12** | Comparison of Wirelength of GSRC benchmark circuits with other algorithms | 108 |
| **Figure 5.13** | Placement for apte (Y=0.0) | 110 |
| **Figure 5.14** | Placement for xerox (Y=0.0) | 110 |
| **Figure 5.15** | Placement for apte(Y=0.5) | 110 |
| **Figure 5.16** | Placement for apte(Y=1.0) | 110 |
| **Figure 5.17** | Placement for hp (Y=0.5) | 111 |
| **Figure 5.18** | Placement for xerox (Y=0.5) | 111 |
| **Figure 5.19** | Placement for xerox (Y=1.0) | 111 |
| **Figure 5.20** | Placement for hp (Y=0.0) | 111 |
| **Figure 5.21** | Placement for hp (Y=1.0) | 112 |
| **Figure 5.22** | Placement for ami33 (Y=0.0) | 112 |
| **Figure 5.23** | Placement for ami33 (Y=0.5) | 112 |
| **Figure 5.24** | Placement for ami33 (Y=1.0) | 112 |
| **Figure 5.25** | Placement for ami49 (Y=0.0) | 112 |
| **Figure 5.26** | Placement for ami49 (Y=0.5) | 113 |
| **Figure 5.27** | Placement for ami49 (Y=1.0) | 113 |
| **Figure 5.28** | Placement for n10 (Y=0.0) | 113 |
| **Figure5.29** | Placement for n10 (Y=0.5) | 113 |
| **Figure 5.30** | Placement for n10 (Y=1.0) | 113 |
| **Figure 5.31** | Placement for n30 (Y=0.0) | 113 |
| **Figure 5.32** | Placement for n30 (Y=0.5) | 114 |
| **Figure 5.33** | Placement for n30 (Y=1.0) | 114 |
| **Figure 5.34** | Placement for n50 (Y=0.0) | 114 |
| **Figure 5.35** | Placement for n50 (Y=0.5) | 114 |
| **Figure 5.36** | Placement for n50 (Y=1.0) | 114 |
| **Figure 5.37** | Placement for n100 (Y=0.0) | 114 |
| **Figure 5.38** | Placement for n100 (Y=0.5) | 115 |
| **Figure 5.39** | Placement for n100 (Y=1.0) | 115 |
| **Figure 5.40** | Placement for n200 (Y=0.0) | 115 |
| **Figure 5.41** | Placement for n200 (Y=0.5) | 115 |
| **Figure 5.42** | Placement for n200 (Y=1.0) | 115 |
| **Figure 5.43** | Placement for n1300 (Y=0.0) | 115 |
| **Figure 5.44** | Placement for n300 (Y=0.5) | 116 |
| **Figure 5.45** | Placement for n300 (Y=1.0) | 116 |
| **Figure 6.1** | (a) Via-First TSVs & (b) Via-Last TSVs | 118 |
| **Figure 6.2** | wirelength estimation models (a) Bounding box of all terminals of a net, (b) Bounding box of all terminals of a net and TSVs associated with them and (c) a net divided into subnets and summing up individual subnet wirelength | 120 |
| **Figure 6.3** | Algorithm for minimizing TSV, Area and Wirelength in two layered 3D-IC using P-PSO | 128 |
| **Figure 6.4** | Comparison of No. of TSVs with other algorithms | 131 |
| **Figure 6.5** | Comparison of wirelength with other algorithms | 131 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table** | **Caption** | **Page No.** |
| **Table 3.1** | Results of Mincut, delay and sleep time results for various circuits using the PSO technique | 58 |
| **Table 3.2** | Comparison of Mincut results for various circuits using PSO technique with GA | 58 |
| **Table 3.3** | Runtime (in seconds) for the proposed PSO | 59 |
| **Table 4.1** | characteristics of MCNC Benchmark circuits | 76 |
| **Table 4.2** | characteristics of GSRC Benchmark circuits | 77 |
| **Table 4.3** | Area optimization results for MCNC Benchmark Circuits (in mm2) | 78 |
| **Table 4.4** | Area optimization results for GSRC Benchmark Circuits (in mm2) | 79 |
| **Table 4.5** | Area (in mm2) comparison of PSO and proposed algorithm for MCNC benchmark circuits | 81 |
| **Table 4.6** | Area (in mm2) comparison of ACO-SA and proposed algorithm for GSRC benchmark | 81 |
| **Table 4.7** | Runtime (in seconds) comparison on MCNC Benchmark for the proposed P-PSO | 82 |
| **Table 4.8** | Runtime (in seconds) comparison on GSRC Benchmark for the proposed P-PSO | 82 |
| **Table 5.1** | Area & Wirelength optimization results for MCNC Benchmark Circuits | 104 |
| **Table 5.2** | Area & Wirelength optimization results for GSRC Benchmark Circuits | 105 |
| **Table 5.3** | Runtime (in seconds) comparison on MCNC Benchmark for the proposed PSACO | 106 |
| **Table 5.4** | Runtime (in seconds) comparison on GSRC Benchmark for the proposed PSACO | 106 |
| **Table 5.5** | Area & Wirelength comparison of PSO and proposed algorithm for MCNC benchmark circuits (Y=0.5) | 109 |
| **Table 5.6** | Area & Wirelength comparison of SKB-SA and proposed algorithm for GSRC benchmark circuits (Y=0.5) | 109 |
| **Table 6.1** | 4-layered 3D IC Parameters optimization comparison of results on MCNC & GSRC Benchmark Circuits | 130 |
| **Table 6.2** | 2-layered 3D IC Parameters optimization results for MCNC Benchmark Circuits | 132 |
| **Table 6.3** | 2-layered 3D IC Parameters optimization results for GSRC Benchmark Circuits | 132 |
| **Table 6.4** | Runtime (in seconds) comparison for 4-Layered 3D IC | 133 |
| **Table 6.5** | Runtime for 4-Layered 3D IC (in seconds) | 133 |

**Abbreviations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IC** | Integrated Circuit |  | **SP** | Sequence pair |
| **IoT** | Internet of Things |  | **TPWL** | Trajectory Piece Wise Linear |
| **SSI** | Small Scale Integration |  | **IMF** | Interconnect-Driven Multilevel Floorplanning |
| **VLSI** | Very Large scale Integration |  | **P/G** | Power/Ground |
| **UK** | United Kingdom |  | **PE** | Processing Elements |
| **AI** | Artificial Intelligence |  | **FPGA** | Field Programmable Gate Arrays |
| **CAD** | Computer Aided Design |  | **MUL** | Multiplier Block |
| **CPU** | Central Processing Unit |  | **CLB** | Configurable Logic Block |
| **PCB** | Printed Circuit Board |  | **RAM** | Random Access Memory |
| **SOC** | System-On-a-Chip |  | **PPTP** | Pre Post Terminal Propagation |
| **TSV** | Through Silicon Via |  | **HPWL** | Half Perimeter Wirelength |
| **PSO** | Particle Swarm Optimization |  | **IO** | Input- Output |
| **ISPD** | International Symposium on Physical Design |  | **SDS** | Slack-Driven Shaping |
| **MCNC** | Microelectronics Center Of North Carolina |  | **TCG** | Transitive Closure Graph |
| **GSRC** | Giga Scale Research Centre |  | **LCS** | Longest Common Sequence |
| **PSACO** | Particle Swarm Ant Colony Optimization |  | **O-Tree** | Orthogonal Tree |
| **SKB** | Skewed B\*- Tree |  | **HSA** | Hybrid Simulated Annealing |
| **SA** | Simulated annealing |  | **DOTFloor** | Diffusion Oriented Time-Improved Floorplanner |
| **KL** | Kernighan-Lin |  | **FATT** | Fast Assumption Technique for Temperature |
| **FM** | Fiduccia-Mattheyes |  | **ACO** | Ant Colony Optimization |
| **MOS** | Metal Oxide Semiconductor |  | **MRST** | Minimum Rectilinear Steiner Tree |
| **NP** | Non Polynomial |  | **CE** | Circuit Elements |
| **3D-IC** | 3 Dimensional Integrated Circuit |  | **NIS** | Non-Overlapping Interval Set |
| **GA** | Genetic Algorithm |  | **ES** | Evolutionary Strategies |
| **TS** | Tabu Search |  | **EA** | Evolutionary Algorithm |
| **DPSO** | Discrete Particle Swarm Optimization |  | **GP** | Genetic Programming |
| **MA** | Memetic Algorithm |  | **MD** | Multi Dimensional |
| **BloBB** | Block-packing with Branch-and-Bound |  | **LOA** | Lion Optimization Algorithm |
| **BDF** | Bus Driven Floorplanning |  | **BCSA** | B\* Tree Crossover Based Simulated Annealing |
| **B\*-Tree** | Binary Star Tree |  | **DFS** | Depth First Search |
| **TRAFFIC** | Trapezoidal floorplanning for integrated circuits |  | **DS** | Dead Space |

**CHAPTER 1**

**Chapter 1**

**Introduction**

The recent revolution in information technology has changed our daily life; not only our approach at work but also our life at home and surrounding are changed. The internet has been extensively used as a means to circulate knowledgeable information, create communication, managing E-commerce. This has been possible only due to the revolution in computing technology as well as in communication technology; both of these are possible due to the revolution in the field of Integrated Circuit (IC) technology. The various electronic devices like microprocessors, computers, memories, interface chips, etc., use ICs as their core element. ICs are the backbone of the recently developed Internet of Things (IoT) technology. This revolution in the field of ICs in recent times has been one of the phenomenal achievements of humanity. The IC technology was emerged in the 1960s from the SSI (Small Scale Integration) to VLSI (Very Large Scale Integration), integration of a few transistors to the integration of multi-millions of transistors in a single chip. Transistor count found in a chip was less than 10,000 until the early 1980s, but since then VLSI-industry has taken a phenomenal development growth. In present days’ industries are fabrication millions of transistors on a single chip.

1. **Introduction to VLSI Physical Design**

The automation of many stages required in the designing and manufacturing of VLSI circuits has enabled (and continues to enable) this extraordinary progress in IC technology Integrated circuits compose numerous electronic components created by stacking many distinct materials in a certain order on a silicon foundation known as a wafer. The IC designer converts the circuit information into a geometrical representation popularly known as the Layout. Multiple layers of flat geometric forms make up a layout. After that, the layout is double-checked to verify that it fits all of the design specifications. This operation generates a set of design files that describe the layout. An optical pattern generator transforms these design files into pattern generator files. The files obtained from the optical pattern generator are then used to generate patterns known as masks. Masks are utilized during the manufacturing process to shape the silicon wafer employing a series of photolithographic processes. The specific information regarding the geometrical design and the spacing between distinct components are required for the production of an electric component or ICs. The physical design process refers to the process of translating an electrical circuit's specification into a geometric layout. Because increasingly small components need a high tolerance for individual components, the physical design process becomes exceedingly time-consuming and error-prone. A component's smallest feature size can be as tiny as 14 nm or even less. Graphcore, a UK-based AI chip manufacturer, has been able to fit up to 60 billion transistors and over 1500 processing units into a single silicon wafer.

Because the fabrication process necessitates precise details of each component, all the stages involved in the physical design process necessitate the use of Computer-Aided Design (CAD) tools. Thus, the goal of the physical design is to find an optimal arrangement of devices on a two-dimensional space (or, in some cases, in three-dimensional spaces), as well as an efficient routing scheme between these optimally placed devices, so that the desired functionality and performance meets the design criteria. As the space on a wafer is limited, algorithms must make the most of it to reduce costs and increase yield. Furthermore, the configuration of devices influences a chip's performance. Physical design algorithms should also make sure that the layout produced meets all of the manufacturing process's requirements. Finally, algorithms must be fast and scalable. Efficient algorithms not only result in a quick turnaround time but also give freedom to designers to make iterative layout adjustments [1].

The VLSI physical design method works with extremely basic geometric elements like polygons and lines. Hence, these design algorithms are highly intuitive and have a lot in common with combinatorial optimization and graph algorithms. Physical design automation is the study of graph theoretic and combinatorial approaches for manipulating geometric objects in two and three dimensions. A pure geometric viewpoint, on the other hand, overlooks the electrical (both digital as well as analog) part of the physical design challenge [1]. Polygons and lines in a VLSI circuit layout have certain inter-related electrical characteristics that display extremely complicated behavior and are dependent on a variety of factors. As a result, while creating algorithms for VLSI physical design process, it is critical to keep the electrical characteristics of geometric shapes in mind. The influence of electrical factors on physical design has a significant impact on the design and development of new algorithms[1].

1. **Overview of Partitioning**

The transistor count in CPUs has increased at an aggressive rate during the last few decades. According to Moore's Law, on a chip, the number of transistors in a dense integrated circuit nearly doubles about every two years. Hence, circuit design challenges have gotten increasingly complicated [2]. From design through packaging, a formal specification of a VLSI chip starts the VLSI design cycle and progresses through several phases to the production of packaged chips.

**Presentation3.tif**

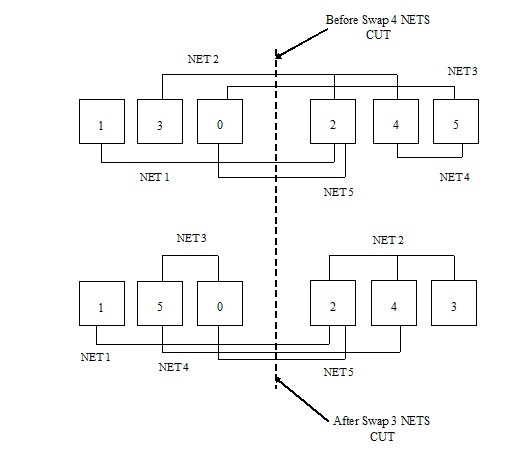
**Figure 1.1:** VLSI Design Cycle[1]

Physical design tools are in desperate need of development, both to manage larger design sizes, as well as to address the issue of interconnection latency. As the connecting wires account for the majority of system delay, wirelength in high-performance systems should be as minimal as possible. Better circuit layout handles this type of optimization at large. Nevertheless, new studies indicate that there is a considerable degree of sub-optimality. Two considerations dominate the environment for partitioning in physical design: top-down design and the emphasis on spatial embedding. Partitioning facilitates dividing and overcoming the meta-heuristics for optimizing floor layout, scheduling and placement. Finally, the algorithm technology developed influences how well onecan address a specific partitioning formulation and optimize a certain aim. [3]

Circuit partitioning is the process of dividing a circuit into smaller sections. For various reasons, it is an important component of the layout. Partitioning divides a circuit into sections that are implemented on various physical components like printed circuit boards or chips. The goal is to divide the circuit into sections so that the component sizes are within specified limits as well as a reduction in the complexity of the components' connections [4]. As the size of today's computer chips grows greater (i.e., chips with more than 10 million transistors in sub-micron regions), the necessity of getting near-optimal architectures that effectively insert and route signals grows. Partitioning is a critical technique for minimizing connections between regions of the chip so that modules may be more efficiently put together and routed to minimize wirelength, congestion and overall design speed. Several goals achieved by partitioning are:

1. **Minimization of the number of cuts:**The number of linkages must be reduced at all stages of partitioning. Reducing interconnections not only decreases latency but also the number of interfaces between the partitions, making design and production easier. The mincut issue is another name for minimizing interconnections.

The primary goal of circuit partitioning is to divide a circuit into two or more parts while aiming to reduce the number of cutnets while keeping a balance on the number of nodes in each partition. Figure 1.2 depicts the reduction of cutnets by switching module among the different partitions.

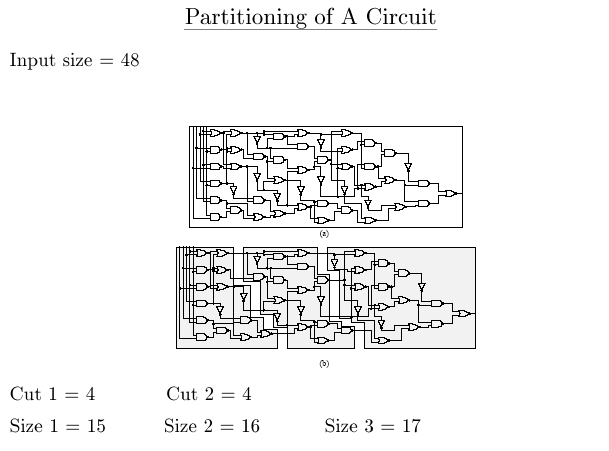
****

**Figure 1.2:** Overview of circuit partitioning[5]

1. **The variation in the number of elements** (inputs, delays, logical gates, outputs and fan-out points) allocated to each division is minimized. The most essential goal in this class is the delay caused by partitioning. A crucial path may cross between partitions several times, because of a circuit's partitioning.. Because the inter partition delay is much greater than the intra partition delay, it is a critical component in circuit partitioning and may be utilized as a partitioning target at all partitioning levels.

Aside from the partitioning objectives, the partitioning is limited to meet the demands of the user. Important considerations for partition constraints are:

* + - 1. **Terminal counts:**Partitioning methods at any level must divide the circuit in such a way that the total number of nets necessary to link one sub-circuit to another does not exceed the sub-circuits terminal count. The restriction in case of system level partitioning is the maximum number of terminals accessible on the printed circuit boards that may link the PCB to the system bus. In the case of the board level partitioning, this limit is the pin count of the package used for chips and the number of terminals available to a sub circuit in the case of chip level partitioning.
      2. **Partitioning area**: It is the limitation in system level partitioning to decrease manufacturing costs while maintaining a small footprint. It is the balancing constraint in board level partitioning to ensure that partition areas are nearly equal.
      3. **Partition numbers:** At both the system and board levels of partitioning, the number of partitions appears to be a limitation. It is significant because, while havingmultiple partitions simplifies the processing of discrete partitions, it may increase the manufacturing cost as well as the number of interconnections among the different partitions. Circuit partitioning is the fragmentation of a complicated system into simple & smaller subsystems to allow each subsystem to be designed separately, hence speeding up the designing process and minimizing interconnections between the subsystems. A balancing constraint is frequently applied to guarantee that each subsystem has roughly the same amount of components and this decomposition is carried out hierarchically until each subsystem is manageable in size [1].



### CUT 2 = 4

### SIZE 2= 16Components

### Cut1=4

### SIZE 1= 15 Components

### SIZE 2= 16 Components

**Figure 1.3:** Partitioning of a Circuit [1]

Partitioning is a Hierarchical process, carried out at three independent levels:

1. **System Level Partitioning**: When a complicated system is divided into several simpler subsystems, each of which may be designed and manufactured separately on anindividual PCB is known as System Level Partitioning. It is utilized in multi-chip systems when inter-chip connectivity delays have a significant impact on system performance. In a system,the number of PCBsand the number of boards are inversely related to each other. As a result, reducing the number of boards is an essential concern in the circuit partitioning of complicated circuits without degrading system performance owing to delays induced by connections between components on separate boards.
2. **Board Level Partitioning:** In this, the circuit assigned to the PCB, is divided into multiple sub circuits, each of which is manufactured as a VLSI chip. In contrast to system level partitioning, which takes into account the size and terminal restrictions of each partition, this partitioning seeks to reduce chip area. As the off-chip latency is significantly greater than the on-chip delay, reducing the number of chips is a key element in system performance.
3. **Chip Level Partitioning:**In this, the chip's circuit is divided into smaller sub-circuits. Chip level partitioning is required because it allows for a more efficient chip design. After partitioning, each sub-circuit, known as a block, is built separately. There is no area limit for any division since physical dimensions do not bind partitioning. However, for design process optimization, partitions can be restricted by user-defined area limitations [1].

Chip

Board

System

Partitioning

**Figure 1.4:** Different levels of Partitioning [1]

1. **Circuit Representation**

At all levels of partitioning, a group of nodes and the netlist are the inputs to the partitioning algorithm.We need to give the input to the partitioning problem in a suitable form to assist it in the process. For circuit partitioning there are three ways of representing a circuit:

1. **Graph Representation**

The circuit-partitioning problem can be formulated as graph-theoretic notation with nodes representing the components and edges representing the connecting wires. The input to the problem consists of a graph, where a vertex in the vertex-set and is an edge in the edge-set. Let represents the area of a vertex andrepresents the weight or cost of an edge. In addition, it is given that the number of partitions are and the capacity of each subset or partition.

The output consists of disjoint subsets such that union of all subsets constitutes set.

C

A

B

D

**Figure 1.5:** Graph representation of the circuit

If is the cost of the edge cut and the cost of the mincut is minimized where the set of edges cut by the partition, , is called the mincut.

1. **Hypergraph Representation**

A hypergraph is a graph represented by with vertex set and hyperedge set .For each vertex in, , is the number of edges incident to it and called as the degree of that vertex. Size represents the size of vertex.

A

B

C

D

#### Figure 1.6: Hypergraph Representation

The number of vertices that incident on a hyperedge e, equals its degree. N is the number of edges between a vertex and a vertex and denoted by. A binary clustering tree is one in which the leaves represent the individual vertices. The tree's internal nodes are groups of vertices. If all the vertices are grouped together then there is a single, the root of which represents all the vertices. A number of different trees may represent the sections of the graph at intermediate phases of the clustering process. The connection between vertex and vertex is indicated bywhen the vertices are clustered jointly in a clustering-based method. A hyperedge is a signal net that connects a collection of cells and a pin is a connection between a vertex and an edge, thus the number of pins on a vertex v equals the number of pins on an edge e equals. The placement problem is defined as assigning hypergraph vertices to physical places to decrease the overall (bounding box) length of all hyperedges to the minimum feasible length. The goal of partitioning is also to minimize the overall number (or weight) of cutting edges while keeping each side's area restrictions in mind.

Circuit partitioning is essentially a graph-theoretical problem: known as weighted graph notation, where components are represented by nodes and edges are the cables linking them. The weights of the nodes represent the sizes of the respective components, while the weight of the edgedenotes the number of wires required for connecting different components. Bi-partitioning methods split the graph into two parts so that the net number of nets linking nodes in a distinct segment is minimal and the weight of nodes in each part does not exceed the prescribed capacity. The above-mentioned goal is known as mincut value. Mincut algorithms are algorithms that view wire crossing division borders as interdependence and try to reduce these interdependencies. Mincut partitioning is achieved by inserting circuit blocks on both sides of an imaginary slice through a cluster, reducing the amount of wires crossing the cut. The process is recursively repeated for each of two smaller clusters until only primitives remain. Partitioning a set of circuit elements into two or more parts such that connectivity between the weighted edges across the blocks can be minimized is computationally hard problem. Kernighan and Lin demonstrated that in the worst case, dividing a set of circuit parts into k blocks would require exponential time by exploring all conceivable permutations in which numberof circuit elements might be divided into nearly equal blocks of size of the order [6].

1. **Overview of Floorplanning**

Following the circuit-partitioning step, the amount of space used by each block (sub-circuit) may be determined along with the potential forms of the blocks and the number of pins required by each of them. The netlist is also available that specifies the connections between the blocks. To complete the layout, we must specify each block's shape, place it on the layout surface andconnect its pins in accordance with the netlist. The blocks are placed in two steps, the Floorplanning (involves the planning of shaping the blocks and interconnects) and the Placement phase (blocks are assigned to a given position). During the placement step, blocks are assigned a specific position on the layout surface such that any two blocks do not overlap each other and there must be enough available space on the layout surface to finish the interconnections between different blocks [1].

The blocks are arranged in such a way that the overall area of the pattern is minimized. Additionally, the pin positions for every block are calculated. A collection of blocks, their potential shape and size, number of terminals in every block and the netlist are the inputs to the Floorplanning The blocks with known dimensions are fixed blocks, whereas the block whose dimensions are not known are flexible blocks. As a result, for each block, we must choose an acceptable form (if the shape is unknown). The Placement issue refers to the task of allocating locations of the fixed blocks on the layout area. When a portion or all of the blocks are flexible, the floorplanning difficulty arises. As a result, the placement is a subclass of the floorplanning problem. The chip planning issue may be referred to as *interconnect planning problem* when it is required in addition to floorplanning. Hence, floorplanning is nothing but a constrained form of the chip-planning problem. The nomenclatures are rather perplexing because floorplanning difficulties are also placement problems, yet these terms are widely in use and acceptable. It is preferable to determine the pin positions at the same instant as the block positions. However, to keepthe placement problem simpler, only after all of the blocks' positions are known,the pins are assigned.*Pin assignment* is the process of designating pin placements. All these stages are necessary for the overall physical design cycle, as the high-quality routing cannot enhance an ill-planned layout. To put it another way, the overall layout quality in terms of size and performance, all the three stages of chip planning, floorplanning & placement are critical. All these planning take into account a variety of parameters. These aspects are discussed in further detail as[1]:

1. **The Shape of the blocks:** The blocks are assumed rectangular to make the problem easier to grasp. For the same reason, the forms generated by the floorplanning algorithms are generally rectangular. Floorplanning algorithms to determine the form of a block use aspect ratios. A block's aspect ratio is the ratio of its height to its breadth. Typically, the aspect ratios have upper and lower limitations, restricting the block's possible size. Other forms, such as L-shapes, need a lot of calculation.
2. **Routing considerations:** Routing must be regarded as an important component of the challenge in chip planning. In placement and floorplanning algorithms, estimating the space necessary for routing may suffice. The blocks are arranged in such a way that there must be enough routing space between them for routing algorithms to accomplish the job of routing nets among the blocks. The placement operation must be redone if complete routing is not achievable.
3. **High-performance circuit floorplanning and placement:** All crucial nets in high-performance circuits must be routed within their time budgets. To put it another way, essential routes must be maintained as short as feasible. The term "performance-driven floorplanning" refers to the process of floorplanning (placement) for high-performance circuits (placement).
4. **Packaging considerations:** When the circuit is turned on, all of these blocks produce heat. The heat dissipated throughout the group of blocks'surface (or surface of the group of the block) put together by the placement algorithms.Hence, during the chip designing process, we must keep the blocks that produce a lot of heat as far as possible, keeping in mind that delay does not increase significantly due to the increased wirelength. This may clash with the goal of high-performance circuits and a compromise must be reached.
5. **Pre-placed blocks:** It is possible that some of the blocks' placements may be set in some circumstances, or a dedicated area for their placement may be provided. For example, the clock buffer, for example, may need to be in the chip's core in high-performance processors. This is done to decrease the delay between the clock signal's arrival times to various blocks. A designer may define a block's placement zone in some situations.
6. **Overview of Placement**

In the physical design cycle, placement is a crucial phase. Poor placement takes more space, resulting in a decrease in performance. It usually results in a tough, if not impossible, routing challenge. A collection of blocks, terminals for each block and the netlist are the inputs to the placement phase. When the layout of a circuit is complete, the block's dimensions are also known. Because an exceptional routing cannot rectify a poor layout, the placement step is critical in the entire physical design process. To put it another way, the placement phase has an impact on the layout's overall quality in terms of space and performance. Blocks are placed at three distinct levels [1]:

1. **System level placement:** The placement challenge at this level is to group all of the PCBs in such a way that the total area occupied is as small as possible.
2. **Board level placement:** All available chips on a board, as well as discrete devices, must be positioned inside a defined region of the PCB at the board level. All of the blocks are rectangular and fixed. Furthermore, certain blocks may be pre-positioned. Components may be mounted on both sides thanks to PCB(which type of PCB) technology. The number of routing layers on a PCB is virtually unrestricted. As a result, regardless of the quality of component placement, the nets may always be routed. The algorithms for board-level placement have two goals: a minimization in the number of routing layers and achieving the system's performance objectives.
3. **Chip level placement:** The issue at the chip level might be chip planning, positioning, or floorplanning, as well as pin assignment. The blocks can be either flexible or permanent and some of them can even be pre-placed. The board level placement problem differs from the chip level placement problem due to the restricted number of layers available for routing in a chip. Moreover, the circuit is only constructed on one side of the substrate.This suggests that certain 'poor' locations may be impossible to route. The fact that a particular location is un-routableis unknown until the routing is not attempted. This causes significant delays in the design's completion.
4. **Motivation**

In the physical design process, *Partitioning* is an important phase; in this phase, a circuit or system is divided into a collection of smaller parts (components). Partitioning leads us to some advantages such as:

1. Due to the ever-growing gigantic increase of complexity in the field of VLSI system design and fabrication, partitioning helps the designers to design synthesis tools that allow to automatically generating huge systems.Synthesis and simulation tools often do not able to cope up with the complexity of the whole system under development and designers need to concentrate on crucial parts of a system to accelerate the design cycle. Hence, the design technology usually needs the partitioning of the system.
2. A large system may become too complex to fabricate due the area and, I/O constraints. Hence partitioning may be needed at various design steps.
3. Automatic partitioning tools are becoming an important topic in the field of academia and industry with the development of intense submicron technologies. An efficient partitioning can significantly minimize the complexity of the design procedure and can handle engineering changes.
4. With the new technologies, the cost of the transistors has dropped significantly but the cost of the I/O pads remains almost constant. Thus, the dimension of the interface between different chips determines the notable portion of the manufacturing expenses. A good partitioning can significantly reduce this cost.
5. In submicron design, the gate delay is dominated by the interconnection delays. Therefore, the partitioning hugely influences the system performance.
6. For the hardware & software co-design, partitioning decompose the design into software and hardware.

*Floorplanning* is another important phase in physical design cycle where shape and location of the node on the wafer is determined. Floorplanning is the foundational step for quality implementation of an SOC. A good floorplanning is such that:

1. Floorplanning is the foundational step for quality implementation of an SOC.
2. In the physical design flow, floorplanning is the starting phase. It is the process of recognizing blocks/ circuit elements that should be placed together and assigning them fabrication area for them in such a way as to meet the conflicting goals of required performance and available space.
3. To make the production cost effective and to utilize the optimization potential of the entire system; different parts of the system should be implemented in a reasonable way to get low-cost fabrication, easy adoption to the changing requirements and optimal system performance, floorplanning can help to make profit.
4. In most of the design methodologies, Area and Speed are considered as the parameters that should be traded off against each other as because there are limited routing resources. By using more routing resources, the design will operate slower. Minimizing the chip area not only allows the design to use lesser resources, but also allows the portion of the design to be closed together. This guide to shorter interconnect distances, faster end-to-end signal paths, less routing resources to be used and faster & more consistent place and route times.
5. Chip planning or Floorplanning phase is very critical in the overall physical design cycle. An ill placed floorplan layout results in higher chip area and total wirelength, which in turn affects the performance of the chip in terms of total power consumption and delay.
6. A good floorplan can reduce the cost of fabrication as well as minimizes the routing congestion.
7. **Objectives of the Thesis**

The physical domain design of VLSI circuits have many problem related to partitioning, Floorplanning, Placement and 3D IC designing. A poorly designed circuits may introduce problems such as delay, power consumption and fabrication cost. The main objectives of the thesis are:

1. To propose physical design automation in the Partitioning so that may address problems such as inter partition delay, reducing the interconnections between the partitions and maximizing the sleep time to reduce the power consumption.
2. To propose a design technique that reduces the overall area occupied by a circuit to the smallest achievable size while maintaining circuit performance.
3. To simultaneously minimizing, the area occupied by a circuit and total wirelength required to connect different circuit elements.
4. The aim of the thesis is to find an optimal solution of physical design automation in the Partitioning and Floorplanning, so that the required chip area is minimized along with the minimization of the total power consumption and the inter Partition & intra Partition delay.
5. To present a strategy for designing 3D ICs that may concurrently minimize TSVs, area occupancy and total wirelength needs.
6. **Organization of the Thesis**

Chapter 2 presents the literature review on the partitioning of VLSI circuits. The literature covers the various partitioning algorithms proposed by the researchers. The results they obtained are discussed. In the same chapter, the research related tofloorplanning & Placement and the related merit/demerits has been also discussed. In addition, different types of 3D-IC design algorithms along with optimized parameters have also been surveyed. This chapter further outlines various literature related to the problems and issues of physical design automation in VLSI circuits.

Chapter 3 introduces the multi-objective optimization technique for placement-related issues. Our proposedParticle swarm optimization (PSO) has been used to optimize the parameters like cutnet, delay and sleep time of individual partitions. Detailed algorithms and flow charts related to optimization problemshave been proposed. In the result section, the ISPD’98 benchmark circuits have been used to show the effectiveness of the proposed algorithm. There is an average 32.47% improvement in mincut, an average 54.76% reduction in delay, an average 60.14% improvement in sleep duration anda reduction of power consumption by an average of 6.58%. An average of 9.71% improvement on mincut has been achieved as compared to the results in [7]

Chapter 4 presents the modified P-PSO technique to optimize the area occupied by a chip. It highlights the demerit of using classical PSO to solve the problem where we need more than one particle to be treated as best. To overcome this drawback a modified version Parallel Particle Swarm Optimization (P-PSO) has been proposed. The validity of the proposed algorithm is tested on MCNC and GSRC benchmark circuits. The obtained results have been compared with a few previously proposed algorithms by various researchers and found to be performing better both in the case of time taken to complete the algorithm and area occupied by a chip on the wafer. The results obtained by the proposed algorithm show an average improvement of 5.15% on MCNC and 7.47% on GSRC benchmark circuits**.**

Chapter 5 examines the third step i.e. placement in VLSI physical domain design. In this phase, the area occupied by a chip is minimized along with the total wirelength required to interconnect different blocks according to the netlist provided. In this chapter, we have investigated the hybridized PSACO algorithm on MCNC & GSRC benchmark circuitsto optimize our combined objective function.The results obtained show an average improvement of 2.52% & 18.27% on the area and wirelength respectively as compared to PSO [52] and -2.94% & 2.06% as compared to SKB-SA [8].

Chapter 6 introduces a novel 3D IC design algorithm to design 2-layer and 4-layer 3D IC. The parallel-PSO technique has been utilized to simultaneously optimize the three parameters namely Through Silicon Vias (TSVs), the total area occupied and, the total wirelength required to interconnect different blocks as per the netlist provided.The obtained results show an average improvement of 52.35% & 29.64% on wirelength& TSV count respectively as compared to SA [9] and an average improvement of 8.36% & 14% on area & TSV count respectively as compared to Tabu Search [10].

Chapter 7 includes the conclusion and future scope.

**CHAPTER 2**

**Chapter 2**

**Literature Review**

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.

1. **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. They demonstrated that for rapid approximation for mincut partitioning, just a few passes are required in practice. They focused on balancing the partition by taking care of the number of blocks in each partition and the block size.

The gain of the FM method simulates the influence of a vertex gain to change the cutsize, which is one of its primary drawbacks. The gain of surrounding vertices is not taken into consideration. As a result, by enhancing the gain of nearby vertices and removing moves that lower the gain of neighboring vertices, the likelihood of discovering a better solution increases. Also for both the KL and FM algorithm, the final partition depends on the initial random partition. This problem is addressed by Krishnamurthy in 1984 [13] by generalizing the ideas of Fiduccia &Mattheyes and suggests a class of increasingly sophisticated heuristics. They have proposed their algorithm in such a way that it accounts for high gain &lower dependencies on the initial partition.

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 componentsand 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. Alpert et. al. [17] investigated the effects of fixed terminals on hypergraph partitioning algorithms in their study. They used a cutting-edge multilevel hypergraph partitioning. They proposed the existence of fixed terminals that can make partitioning simpler. It took less work to get consistent solution characteristics that were close to the best possible. They investigated the pass statistics of the flat FM- partitioning technique.Consequently, they concluded that having more fixed terminals means implies that improvements within a pass are more probable to occur at the start. They also suggested that the unique architecture of partitioning in the fixed-terminals regime had significant implications for partitioning design and use in top-down placement. All these algorithms only concentrated to minimize the number of cuts in a bipartition.

Farrahi & Sarrafzadeh [18-19] have partitioned a system to optimize the amount of sleep time. The sleep period inspires the idea of deactivating memory refresh circuits, turning off the power, or halting the clock signals while circuit elements are inactive, hence reducing power consumption. As assigning different control signals for each circuit element is impractical due to the potential for high power consumption, they recommend partitioning a circuit based on the activity pattern of its elements, such that they can be switched into sleep mode for extended periods rather than turning off a single circuit element. They proved that partitioning is an NP-hard issue. Geometric partitioning (geo part) utilizing tree data structure is described in their work.

Ababei et al [20]simultaneously optimized the cutsize and delay in a multi-objective hMetis partitioning. They altered the partitioning procedure by creating a new goal function for the most essential pathways based on the path-based delay component. They employed the standard slack-based delay component in the goal function to avoid semi-critical paths from turning critical.

Sait et al[21] presented a change to the Fidducia-Mattheyeses algorithm that considers the cutnet's power consumption minimization. Power-FM is the name of the new heuristic. They looked at the benefits of Power-FM as an initial solution generator for multi-criteria optimization over other iterative methods like Genetic Algorithm (GA) and Tabu Search (TS). Their heuristic indicated a suitable place to start when utilizing iterative algorithms for multi-criteria optimization.Ekpanyapong& Lim [22]offered a unified framework for multilayer partitioning with retiming, aiming for both delay and power reduction at the same time. They proposed retiming delay and power are more important for sequential circuits than static delay and overall power. They also introduced the GEOPD method for simultaneous optimizing delay and power, which allows for a smooth tradeoff between power, cutsize, wirelength &delay. They presented an effective method for converting the timing and power analysis results from the original netlist to a coarsened sub-netlist that can be utilized for power and multi-level delay optimization.

Ghafari et al [23]provided a modified version of the Genetic Algorithm as well as a geometric iterative approach based on segmented trees.They separately formulated the delay and power optimization and then combined these two objectives in a normalized objective function. They tested their algorithm on three hypothetical cases and demonstrated the improvement.

Peng, Chen & Guo [24] proposed Discrete Particle Swarm Optimization (DPSO) for optimization of VLSI interconnection (netlist) bipartition. They designed evaluation functions and operators for crossover and mutation. The proposed algorithm was tested on the ISCAS89 benchmark and compared the improved results with the traditional Genetic Algorithm.

Gill et al [7, 25]presented a swarm intelligence-based and GA-based circuit partitioning approach. They split the circuit into divisions and reduced the number of links between them.

Shanavas et al [26]used Memetic Algorithm (MA) to solve the VLSI partitioning problem. They demonstrated that the MA solves the VLSI partitioning problem by combining a hierarchical design method, a Genetic Algorithm and constructive techniques like Simulated Annealing.The MA uses the local search technique to produce the optimal solution avoiding premature convergence.

Cong & Leung [27]investigated the optimum wire sizing problem using the Elmore delay model in their work. They demonstrated that the best wires sizing solutions meet a variety of characteristics, including separability, monotonicity and dominance. They created a polynomial-time optimum wire-sizing method for any connection tree topologies using the Elmore delay model based on these features. They compared their model to an RC delay-based wire-sizing solution.

1. **Literature Review on Floorplanning & Placement**

Floorplanning is an important design stage in VLSI physical design since it specifies the size, shape and positions of modules on a chip, as well as the overall chip area, interconnects and latency. VLSI floorplanning is an NP-hard problem in terms of computation. The floorplan representation is an important part of the floorplanning stage. The complexity and search space of the floorplan design is influenced by the floorplan representations. To tackle the VLSI floorplan problem, numerous academics have proposed various heuristics and metaheuristic algorithms. A Few of them are listed below:

Adya*&*Markov [28] have used Simulated Annealing technique to address the floorplanning problem. In their paper, they have studied Fixed-outline Floorplanning with sequence pair representation. They demonstrated that fixed-outline floorplan issues are substantially more difficult than traditional floorplan problems. They proposed that new objective functions be used to drive simulated annealing, as well as new sorts of movements that would better direct local search.The slack calculation is used to generate their recommended movements. The authors addressed wirelength improvements and soft block aspect ratio optimization in the combined objective function.

Chan and Markov [29] introduced BloBB, a novel floorplanner based on multi-level branch-and-bound. They sought to quantify the difference between optimum slicing and non-slicing floorplans by comparing optimal packing. They demonstrated their findings by packing soft blocks of every MCNC benchmark except apte and every GSRC benchmark.

Chen and Chang [30] investigated two new types of contemporary floorplanning problems namely fixed-outline floorplanning and bus-driven floorplanning (BDF). Their algorithm, Fast-SA, employs arapid three-stage simulated annealing (SA) technique used to depict a B\*-tree floorplan. They developed an adaptive Fast-SA to handle fixed-outline floorplanning that can actively modify the weights in the cost function to optimize the wirelength. They demonstrated their findings using soft blocks for fixed-outline floorplanning. They also investigated the feasibility criteria of the B\*-tree with bus restrictions for the BDF based on Fast-SA.

Sassone and Lim [31] introduced trapezoidal floorplanning for integrated circuits (TRAFFIC), a unique approach for wire and area minimization. Theyused the connectivity grouping, basic geometry and a restricted brute force approach to obtain an average 18% lower wire estimate than achieved by simulated annealing (SA).

Lin and Hung [32] presented an SKB-tree representation for two floorplanning issues, fixed-outline and voltage-island driven. They proposed that the SKB-tree is appropriate for dealing with fixed-outline floorplanning because it dynamically allocates space for blocks, allowing them to be put into a specified outline. They have also utilized this SKB-tree characteristic to deal with voltage-island driven floorplanning. They put blocks of the same voltage together in one location to conserve power routing resources, simplify power planning and decrease IR loss. The authors utilize the Simulated Annealing approach to find the best layout.

Murata et al [33] proposed the sequence-pair floorplan, a pair of module name sequence known as sequence pair (SP) that represents a floorplan uniquely. The authors utilized simulated annealing approaches to find the best wirelength-driven solution.

Long et al [34] advocated the optimization of Microarchitecture configurations &floorplan at the same time. They focused first on floorplanning under specified microarchitecture configurations and subsequently on minimizing throughput degradation caused by pipelined global interconnects-based microarchitecture. The authors proved that an accurate trajectory piecewise linear (TPWL) model takes more offline setup time than a crude access ratio-based strategy to attain greater throughput.Their testing results indicate that both proposed models result in much higher throughput than standard floorplanning approaches.They then used the TPWL approach to create a single throughput model for pipelined global interconnects and microarchitecture configurations, which they subsequently used to investigate themicroarchitecture combinations and related floorplan variations.

Chen et al. [35] proposed an interconnect-driven multilevel floorplanning framework to manage large-scale interconnect-driven multilevel floorplanners (IMF).Rather than using the “Λ-shaped” structure of bottom-up coarsening and top-down un-coarsening, they used the "V-shaped" framework of top-down un-coarsening (partitioning) followed by bottom-up coarsening (merging). They initially partitioned the floorplan region using min-cut bi-partitioning to reduce global interconnects and therefore overall wirelength and then used the Simulated Annealing optimization technique to bottom-up merge these partitions in fixed-outline floorplanning iteratively.They presented their findings on the MCNC and GSRC benchmarks. All the above algorithms [29-35] investigated the floorplan technique to address either the floorplanning area or the wirelength minimization.

Power/Ground (P/G) network co-design with floorplan, according to Li et al [36], can increase the quality of power design at the floorplanning stage.The authors describe a B\*-tree based method for floorplanning & P/G network co-design in their study, to minimize violations, employing an efficient P/G network analysis technique and a guided incremental floorplanning algorithm.Pavel and Sen [37] proposed that power consumption in Systems-on-Chip can be lowered by lowering the Processing Elements (PEs) voltage Levels. They propose a cost function that takes into account not just the overall space requirement, but also the total power consumption and the maximum number of voltage islands in the system.They offered a greedy algorithm depending on the floorplanning of PEs with multiple voltage islands.

For heterogeneous FPGAs Liu, Chen & Yoshimura [38] proposed a three-phase floorplanning strategy, which included MULs (multiplier blocks), CLBs (Configurable Logic Blocks) andRAMs.They optimized the wirelength using a non-slicing floorplanning method, then used a min-cost-max-flow algorithm to tune the assignment of CLBs such as all the functional modules get satisfied by their CLB requirements and finally, a network flow model was used to allocate MULs and RAMs to modules.

According to Chong &Pasir [39], die count estimation is critical for best product cost estimation based on floorplanning feedback.They looked at the Gross Die Estimator equation as well as the drawbacks of using a gross die estimator.

Hoo et al [40] proposed Pre-Post Terminal Propagation (PPTP) that uses a multilevel structure to manage soft module floorplanning. PPTP was used at the root node to improve the amount of partitioning that could be done.The authors used PPTP to compensate for the absence of terminal propagation in later partitions, which could result in a lack of information regarding external pins at each level of the tree.Based on GSRC benchmarks, they employed PPTP to compute optimal HPWL for soft module floorplanning.

Lim et al [41] proposed an optimal flip-chip floorplanning method by using a log-sum exponent wirelength model with an analytical placement technique. To save the silicon area and to reduce wirelength the authors have placed IO pads inside the core area as area IOs.Nain and Jeske [42] proposed a placement-aware 3-D floorplanning algorithm for wirelength minimization that takes into account the 3-D placement of logic gates and allows for the introduction and evaluation of a vertically aligned part of the same gate assigned to different device layers. The authors used the sequence pair technique and demonstrated their findings on the MCNC and GSRC benchmarks.

In a 3-D IC, Li et al [43] used Through-silicon vias (TSVs) to connect the signals that are used in two or more layers. They claim that because TSVs take up a lot of device space and are much bigger than logic gates, it is vital to redistribute the white space. The authors present a three-dimensional floorplanner that can simultaneously design functional modules and position TSVs while optimizing overall wirelength under a fixed-outline constraint.

According to Hong et al[44], 3D floorplanning should account for die stack ordering since die properties can vary due to growth process variances.They present a novel 3D floorplanning method that takes into account dice stacking.

SDS is a slack-driven shaping method that is used for fixed-outline floorplanning. Usingsoft blocks based on non-slicing floorplanning proposed by Yan and Chu [45],SDSminimizes the layout height by re-shaping the soft blocks, ensuring that the aspect ratio of layoutdoes not exceed the provided upper bound, given a predetermined upper bound on the layout aspect ratio.The amount of change in a block was calculated by spreading the entire available slack to each block in a methodical manner.The outcomes of the experiments are displayed on the HB and MCNC benchmarks.

Rani & Rajaram [46] in their experiment handled the alignment constraints using B\*tree representation with Differential Evolutionary algorithm. To reduce the solution space, feasibility conditions of a non-slicing floorplan with alignment constraints have been examined by the authors. The experimental results were shown on the MCNC benchmark.

Rabozzi et al. [47] point out that floorplanning is a crucial step when working with partially reconfigurable FPGA designs, as it has a notable impact on the system's behavior and reconfiguration overhead.The authors proposed the use of genetic algorithm (GA) combined with a local search approach to automate floorplanning on a stated direct problem representation. They tested their strategy on a synthetic benchmark suite as well as real-world case studies. The researchers then compared their findings against state-of-the-art algorithms and alternative engines based on the same direct problem representation.

He and his colleagues [48] argue that fixed-outline floorplanning not only consumes too much time but also difficult to accomplish when additional constraints are taken into account. The authors presented an approach for dealing with fixed block limitations that depends on fixed-outline packing with the post process. The authors worked around the fixed-outline constraints by recursively modifying the crucial path in the Transitive Closure Graph (TCG).They also added some virtual nodes to TCG for restricted blocks with fixed positions, allowing the incremental process to satisfy both fixed-block and fixed-outlines restrictions. On the MCNC benchmark, the experimental findings were displayed.

Tang et al. [49] stress the importance of a quick approach for converting sequence pairs to floorplan.They demonstrated a new technique based on a pair of weighted sequences and the Longest Common Sequence (LCS).To address sequence pair problem, they presented a simple and efficient O(n2) approach. They also demonstrated that the LCS may be implemented in the order of O using a more complex data structure (n log n). The authors were able to achieve a 60-fold increase in speed compared to the previous graph-based approach.

Pang et al [50] have tried to reduce the complexity of O-tree based floorplan from the O(n3) order to O(n2). They have tested their results on the MCNC benchmark.

To overcome the floorplanning problem, Chen et al [51] proposed incorporating the principle of crossover and mutation operator into PSO.Integer coding built upon module number was used with the proposed technique.The authors present experimental results for MCNC & GSRC benchmark using a slicing tree data structure.

To solve the floorplanning area optimization problem, Chen et al [52] introduced hybrid Simulated Annealing (HSA). To balance global and local exploitation, the suggested HSA uses a new greedy technique to generate an initial B\*-tree, a new operation on the B\*-tree to explore the search space and a novel bias search strategy.The authors presented experimental results using MCNC standards.

Choudhury and Pradhan [53] introduced DOTFloor (Diffusion Oriented Time-improved Floorplanner), a time-efficient and reliable floorplan method based on a SA (Simulated Annealing) engine.The on-chip temperature, the chip area andwirelength, all were optimized.They also created the FATT (Fast Assumption Technique for Temperature), a heat-diffusion-based stochastic thermal model that allows for quick speculation of the hotness throughout the optimization process.They displayed their results on the MCNC benchmark and compared them to the HotFloorplan, an existing floorplanning application.

Pinge et al. [54] proposed a unified solution for dealing with the alignment and cluster limitations in floorplanning using sequence pair representation. They concluded that their strategy aids in the pruning of infeasible sequence pair solutions and decreases the solution space, allowing the algorithm to run faster.The results of the area optimization were displayed on the MCNC and GSRC benchmarks.

To build a floorplan for 3D ICs with fixed-outline limitations, Xu et al [55] developed a two-phase method combining Simulated Annealing (SA) & Ant Colony Optimization (ACO).They made two decisions during the packing process: first, they chose a block to pack and then they found a suitable location for the chosen block in the partially completed floorplan.They used soft modules to demonstrate their findings on a handful of the GSRC benchmarks.

1. **Summary**

In this chapter, we provide an overview of early history and previous research works related to partitioning and Floorplanning. The various research works based on different published work related to optimizing mincut, minimizing inter-partition and maximizing sleep times to save overall power consumption while bi-partitioning a large circuit have been cited. The minimization of wirelength consumption and reducing the floorplan area has also been cited. In 3D-IC design, the cited works are based mainly on optimizing the number of TSVs requirements with a few of them addressing the area and wirelength constraints as well. Although in 3D-IC design the researchers tried to simultaneously optimize the maximum of two parameters among TSV, area and wirelength; none of them investigated to optimize all these three design constraints simultaneously. There are certain situations where the different design constraints are needed to be optimized simultaneously. In chapters, 3 to 6 simultaneous optimization of various design parameters of Partitioning and Floorplanning have been investigated.

**CHAPTER 3**

**Chapter 3**

**Partitioning of VLSI Circuits for mincut, delay & sleep time optimization using PSO technique**

Circuit partitioning is a physical design methodology that divides a given circuit into segments while adhering to certain constraints and achieving specific goals, such as minimizing the mincut (number of physical connections between partitions), minimizing the delay between partitions, maximizing the sleep time of the partition to reduce power consumption, and determining the algorithm's time complexity. In this chapter, we have discussed the simultaneous optimization of these parameters using PSO technique. The ISPD’98 benchmark circuits have been used to illustrate the optimization goals.

1. **Introduction**

The first step of a VLSI physical design is to partition a given large circuit in manageable numbers of smaller partitions, such that each of the smaller sub-partitions may be designed separately and then various partitions are connected. The number of interconnections among the partitions is known as mincut or cutnet. The partitioning of a system can be done in two different ways; in the first way, the larger circuit may be divided into predefined number of partitions, while in the second way the circuit is bi-partitioned. The first way is computationally a tedious task and a design algorithm designed for a particular circuit may not be applicable for the other similar circuits with different circuit parameters. On the other hand, bipartitioning technique recursively applied to achieve the predefined number of partitions. Portioning of a circuit generated inter partition delay due to the introduction of cutest, which generated a delay component known as inter partition delay. This delay is much larger than the intra partition delay and may significantly affect the performance of a chip. Hence, the partitioning is done in such a way that the cutest number is kept to be as low as possible. Partitioning introduces an extra advantage where the different partitions may be controlled differently where it may not be necessary to keep all the partitions in a switched-on mode. A given partition may be kept into sleep mode to optimize the power requirement. Hence, partitioning is performed to achieve the minimum number of mincut and minimum inter partition delay while maximizing the sleep time.

1. **Objective Function**

The partitioning of the circuit is non-polynomial hard and deterministic algorithms cannot tackle it well.The particle swarm algorithm is a stochastic algorithm that is both probabilistic and iterative. It may be used to split circuit netlists effectively.

1. **Problem Definition**

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

1. The **bipartitioning** challenge entails bisecting a given circuit to minimize the cut (i.e., the interconnections linking the two partitions) or the ratio-cut (i.e., the ratio of cutest between two partitions to the product of cardinality of the two partitions).

C (X, X’)

X’

X

**Figure 3.1:**Bipartitioning of a Circuit

1. The **multi-partitioning** problem entails breaking down a bigger system into more than two smaller subsystems, for example, a k-way multi-partitioning circuit is divided into subsystems to minimize interconnections between them. Each sub-system is frequently subjected to an area constraint, ensuring that all sub-partitions have about equal numbers of components. There are two types of multiway partitioning methods now in use: recursive and direct. The direct approach partitions the circuit into k-partitions directly, whereas bipartitioning is used iteratively until the desired number of partitions is attained in the recursive technique. The recursive method is easy and quick to compute. However, it does, have three main flaws. First, we cannot get the required multi-way k partitioning by iteratively utilizing bipartitioning if the is not a power of 2. Second, it becomes increasingly difficult to lower the cut size since early-stage cut nets cannot be eliminated when bi partitioning is applied to finer graphs. A highly efficient cut set in one-step, for example, may force the next stage to function on dense blocks. When we apply more bipartitioning to these packed blocks, it has a detrimental influence on the result. Third, recursive partitioning seeks to reduce the cost (k-1) metric rather than the cost 1 metric, which is frequently the goal.
2. **Mincut Optimization**

The task entails splitting the circuit netlist into two subsets, as well as cutting some of the connections (edges).

The cost of a partition is the number of edges that belong to two different partitions and this cost may be described as follows:

(3.1)

Where denotes the cost of an edge connecting to the components in two sides.

Where, represents the vertices in respective sides.

In graph-theoretic form, the partitioning problem may be described more intuitively. A hypergraph can be used to describe a partitioning problem. Where, and are the collection of vertices and a set of hyperedges respectively.

A block or node is represented by vertex, while a hyperedge represents the net.

The task is to divide into partitions.

Where,

(3.2)

(3.3)

Since the problem implies the bipartitioning of a circuit, the equality requirement must be met as follows:

(3.4)

Where, are the nodes in two partitions [1]

1. **Delay Minimization**

First, we examine the essential input/output ports (pads) routes. The crucial path is the one that has the greatest delay between the I/O pads.

(3.5)

Where,

[1]

We utilize the well-known Elmore Delay model to compute this delay. There is two parts to our delay model.The gate delay is the first component.We are usinga typical intrinsic delay for a typical input transition and a typical output net capacitance for all gates.The wire delay is the second component, which we estimate using the Elmore delay model. This can be is calculated as follows: -

(3.6)

(3.7)

(3.8)

Where,

: Wire lumped resistance

: Wire lumped capacitance

The overall lumped capacitance of each net's source node, set to zero[21].

As it is required to estimate the length of each edge to compute and.The statistical net-length estimation approach, MRST (Minimum Rectilinear Steiner Tree) model, is used. Assume that the net in figure 3.1(a) is segmented as indicated in figure 3.1(b). In the net enclosing zone, the lengths of the dashed line segments sum to the lengths of .As a result, the MRST net's real wirelength is the half perimeter plusthe solid line segments.

(3.9)

a

b

a

b

(a)

(b)

**Figure 3.2:**An example of a six-terminal net based on MRST (a) MRST is used to create the entire net structure (b) The dimensions of the net enclosing region are shown by the segments.

The length of the additional segments is determined by the number of terminals and the size of the enclosing region and is denoted by. It is also affected by the position of the terminals.The length of a net connecting m cell in a rectangular area with width 'a' and height 'b' may be determined as shown below:

(3.10)

Where the first term, is a representation of, the number of nets is m and the dimensions of the net enclosing region are.The parameters for fitting are calculated as.When a net is severed during recursive partitioning, it receives a wire delay, this is used to re-evaluate all delays on the paths that comprise that network. The greater the degree of recursive division, the more a net is divided and the greater is the wire delay. Every net cut during the initial bi-partitioning phase (Figure 3.1) is contained in a rectangular region equal to the chip area in this case, with an aspect ratio of one for simplicity.When a net is cut, its delay is set only for the first time. In this dissertation, we have used a copper manufacturing technology in our tests (unit length resistance unit length capacitance) [56].

1. **Sleep Maximization**

**Presentation2.tif**

(a)

**Presentation1.tif**

(b)

**Figure 3.3:**Using circuit partitioning to take advantage of sleep mode operation

(a) Memory Segmentation & (b) owerDown Partitioning [19]

Let be the total number of circuit elements (CEs). may be switched into sleep mode during time period, if is idle during that interval. Intervals are non-overlapping if. A set of intervals is non-overlapping if they do not overlap pair wise.

The idle set of is made up of a series of non-overlapping intervals, also known as a NIS (Non-overlapping Interval Set), in which m is inactive. Assume that a set represents the idle set of M. where is the idle set of [16].

The term "empty interval" is used to describe a space that is not filled with anything.We say covers if either, or for intervals and), (that is, all intervals encompass the empty interval).The number is used to calculate the length of an interval, the interval is said to contain point.If one of the intervals in a NIS includes point, the NIS is said to contain point p.The largest interval covered by both, written as, is defined as the intersection of two intervals (or empty for non-overlapping intervals).Similarly, it is defined as the intersection of more than two intervals.

Because the intersection operation on more than two intervals is plainly commutative, there is no need for parenthesis. The NIS produced by the non-empty pair wise intersection of two NISs and, designated as, is defined as:

(3.11)

The intersection of more than two NISs is defined in the same way. The intersection of several NISs is a commutative operation therefore, parentheses are unnecessary.We saycovers ifgivenNISs The NIS Ni's endpoint set EN is defined as the set of interval endpoints in N as:

(3.12)

The total of the lengths of the intervals in a NIS is defined as.The internal intersection of S is defined as the intersection of all the NISs in S, given a set of NISs.

(3.13)

The endpoint set ES of S is defined as the union of the endpoint sets of the NISs in S as:

(3.14)

Given a set a bi-partitionof S ifand.The bipartitioning is b-balanced ifwhere represents the cardinal number of set and it is referred to as the size of partition. A bipartitioning of defines a similar bipartitioning of and vice versa, where. In other words, is the set of CEs whose idle sets are subdivided into.At a given point, the density of a partition is defined as the number of NIS in that include an interval containing. The gain of S's b-balanced bipartitioning is defined as follows:

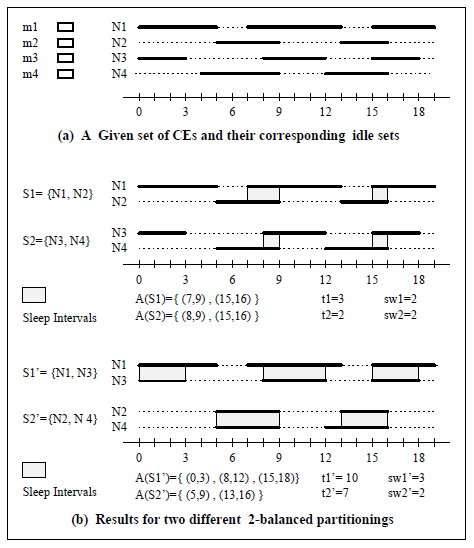
(3.15)

Where:

(3.16)

(3.17)

The characteristics represent the sleep periods of partitionsrespectively, while the values represent the switching of partitions, respectively. The idle set of the corresponding partition is the internal intersection of a partitioni.e., the maximum times during which all CEs in are idle. Furthermore, any time in a partition's idle set will be referred to as the partition's greatest sleep interval.A sleep interval of a partition is a valid sub-interval of that partition's maximal sleep interval [16].



**Figure 3.4:** Partitioning to maximize sleep time [16]

It should be emphasized that larger discrete overlapping of idle time requires more switching and more complex control circuitry.As a result, the gain should be a decreasing function of and an increasing function of.The gain function that must be maximized in a bi-partitioning problem is defined as:

(3.18)

In equation (3.16), the total of accounts for power consumption savings caused by the partitions' sleep mode operation, whileaccounts for the added control circuitry's power consumption overhead. The parameter controls the relative relevance of power savings () and overhead termsthe technology and kind of circuitry in module m have an impact on it.

Letthe power usage with or without sleep mode is denoted by respectively, as determined by our partitioning approach, then using

(3.19)

(3.20)

Where, the power consumption of each memory partition in operation and sleep modes, respectively, are and the computation time is T. The percentage of energy savings R will be given by[16]:

(3.21)

For a particular memory chip, is generally greater than 25, implying that the proportion of power consumed is at least:

(3.22)

1. **Combined mincut, delay and sleep time optimization**

While the purpose of traditional partitioning algorithms is to reduce cutsize, which hMetis partitioning excels at,[18]the multi-objective partitioning issue is substantially more challenging.One of the most challenging elements is that there is no one optimal solution in the solution space for each goal.Furthermore, an optimum result for one goal may imply the implementation of a suboptimal result for another. As a result, the idea of a good solution has become confusing.The following formulation of a satisfactory solution is used in [57]:

1. Allow for fine-grained control over the objectives' tradeoffs.
2. Produce predictable partitioning.
3. Provide a method for dealing with objectives that relate to different types of quantities (range, variance and sensitivity to partitioning changes).

Currently, the algorithm for partitioning must be capable of minimizing objectives, the number of wires cut (thus decreasing placement congestion), how frequently most vital routes are cut (thus minimizing delay in the circuit) and maximizing objective S: the number of sleep time, the amount of time the partitions remain idle. However, because the three objectives are distinct, maximizing one does not always indicate that the others are likewise optimized and vice versa. For multi-objective optimization, we use a combination-based formulation: The combined objective will be a scalar combination metric, as seen below:

(3.22)

Where,

are the preference vectors,

is the current sleep

is the current mincut,

is the current delay,

Maximizing equation (3.21), on the other hand, attempts to generate a partitioning that is not significantly different from any of the optimum in terms of any beginning objective.The distance between the best solution locations of each objective may be traversed using the preference vector. Therefore, predictable partitioning and adjustment over the control of the tradeoff between the three goals may be achieved[20]

1. **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:

**Algorithm for solving partitioning problem using PSO**

1. Begin by converting the netlist into matrix form at the beginning.
2. Bipartition the circuit into 0 and 1 partitions as

(3.23)

Where,are the nodes in two partitions

Also, from (3.4),

{0 partition} {1 partition}

1. Initialize the particles’ positions using some random value and divide this random value with the length of either partition generated earlier. Use this calculated random value to swap the node between the two previous partitions to create new partitions.

Denotes PSO particles with ranging from. (Any user defined value)

1. Create any number of particles using this concept. Calculate their mincut using

(3.23)

In addition, delay using the equation (3.6). Evaluate the sleep time of the two partitions. Correspondingly evaluate fitness function, for all the particles using (3.22), taking preference vectors (for 50% weight to sleep and mincut objectives).

1. Take the maximum of these values as and all other values as individualfor all the particles.
2. Initialize the velocity for all the particles.
3. Assume that all of the particles' initial positions are their
4. Using PSO equations, particle velocityand particle positionare evaluated. This updated particle position is used further to create new particle using the same concept as discussed in step (c).
5. Apply PSO equations L number of times.
6. Calculate number of interconnections, delay and combined fitness function for all the particles using (3.23), (3.6) and (3.12) respectively.
7. Assume that the new combined fitness function =

Compare with for each particle.

If, then accept

If, then will remain the same.

1. Find the particle with the smallest fitness function and mark its position as.
2. Using equations(3.22) & (3.23), for each particle, compute new locations using their current (the particle with the *gbest*fitness value's position).
3. END the program if the termination condition is true (i.e., the number of iterations reaches a user-defined threshold value) otherwise, go to step (8).
4. After optimizing mincut and delay for the partitions, calculate the time complexity of the algorithm

The Steps for calculating Time Complexity of PSO algorithm are as under :

There are a number of modules e.g. sleep\_matrix, sleep\_test,Mincut, Delay, initial position, read\_adja, which contribute to time complexity of algorithm i.e. PSO algorithm.

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

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

1. Initializing ‘cutsize’ will execute only once.
2. First ‘for loop’ will execute length (partiiton1) times i.e. N/2 times.
3. Second ‘for loop’ will execute length (partiiton2) times i.e. N/2 times.
4. ‘If’ statement will execute only when there is an interconnection between partition1 and partition2 i.e. 3N times.
5. Cutsize increment will take only one operation.
6. Time complexity of mincut module is 1+3N3/4.

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

1. Finding sizes of partition1 and partition2 will take only one operation.
2. Firstand second ‘for loop’ will execute N/2 times because partitions are equal.
3. ‘If loop’ will execute the number of times interconnection between two partitions is high.
4. This combination of ‘for and if loops’ will execute (N/2) (N/2) (3N) times.
5. Then evaluating area, a and Assigning values to alpha,beta,gamma,r\_unit,c\_unit as 1.1,2.0,0.5,0.115&0.0015 respectively will take only one operation, total of 5 operations.
6. Calculating unit\_r, unit\_c,l\_avg, re,ce,edel will also take one operation each, a total of 5 operations.
7. Time complexity of delay\_model module is 15+3N3/4.

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

1. Assigning array ‘m’ to an array of zeros will execute once.
2. Initialization of Partition1 and partition2 to zero will be of one operation.
3. ‘For loop’ will execute 3N times.
4. Assigning x, y to ‘0’ will take only one operation, x, y are being used for indexing of partition1 and partition2.
5. ‘For loop’ to differentiate the partitions, will execute N (N/2) (N/2) times.
6. Length of partition1 and partition2 will be executed one time.
7. (N-7) nodes will be exchanged to have balanced partitions.
8. Time complexity of initial\_position module is 3+9N+2+N (N2/4) + (N-7).

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

1. Area file will be given as an input only once.
2. Its path name and file name will be stored in ‘file’ only once.
3. fopen will open the stored area file.
4. Each and every character and digit is read from that file.
5. Size of area\_array is calculated once.
6. 0s are assigned to area\_matrix.
7. ixx is assigned a value once.
8. ‘For loop’ will execute N (N-2) times.
9. Time complexity of read\_area module is N (N-2) +6.

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

1. ‘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.
2. Then, corresponding nodes in both positions are exchanged.
3. After that, new positions are stored in newpartition1 and newpartition2.
4. Total time complexity of nodes\_swapping is 8.

**Steps for calculating Time complexity of sleep\_matrix:**

1. Calculation of node execute once.
2. Assignment of clock will execute once.
3. St Matrix initialization will be of one operation.
4. ‘For’ loop to calculate the st matrix will take time.
5. Time complexity of sleep\_matrix module is.

**Steps for calculating Time complexity of sleep\_test**

1. Calculation of nodes execute once.
2. Calculation of length of partition1 execute once.
3. Calculation of length of partition2 execute once.
4. ‘for’ loop to calculate sleep time for partition1 will take times.
5. Similarly, ‘for’ loop to calculate sleep time for partition2 will take times.
6. Total time complexity of sleep\_test module is

**Steps for finding Time complexity of PSO:**

1. Adjacency matrix created once.
2. Area file read once.
3. Size of adja calculated once.
4. u is evaluated once, where ‘u’ represents length of the either partition.
5. Tolerance, t taken once.
6. Time complexity of initial\_position is to be taken into consideration.
7. Time complexity of ‘mincut’ module is 1+3N3/4.
8. Time complexity of 'delay\_model’ module is 15+3N3/4.
9. Ys­,Yc, c, d, s, cc are evaluated once each, taking total of 6 operations.
10. NP is assigned a value.
11. ‘for loop’ is executed NP times.
12. Inside ‘for’ loop: Particle is generated via random concept. Absolute of random value is taken. Then, remainder is stored after dividing that absolute random value with the length of 1st partition. This remainder is rounded towards the infinite value and is taken as the node to be swapped. Then, this value is swapped between two partitions. Mincut, delay and sleep of the new partitions (obtained after swapping) is calculated. Accordingly, Cc value is calculated using this new mincut, delay and sleep value. cc value is stored as pbest and maximum of this is stored as gbest. Corresponding fit partitions are stored. Then, this gbest value is stored and its corresponding mincut, delay and sleep values are stored.NP(74+9N3/4)
13. Next ‘for’ loop executes (M-1) times. (M-1).Within these loop, ‘for’ loop for PSO is started. Within this for loop, particle\_velocity and particle\_position equations are implemented. This new particle position value is used further for swapping the nodes applying the same procedure as discussed above. Thus, new partitions and their corresponding mincut, delay values are compared with those of previous fitness values (obtained before applying PSO). If new ones are optimum, then these are retained and plotted against number of iterations(NP)(51+9N3/4+4NP).
14. Adding time complexities of all the modules results to, 33+9N3/4-6NP+37M-4NP2+41.NP.M+9N3.M.NP/4+4NP2.M. Finally, the highest term in this expression is N3.NP.M.This algorithm's time complexity is O(N3.NP.M)

Initialize the array ‘m’ to zero to store all the particles present in particular netlist

Initialize Partition1 & Partition2 to zero

Start a loop to define random position for each particle and creating an array ‘m’ containing the position values of each particle

If position of a particle is ‘0’, put it in Partition1; else if it’s ‘1’, put it in another partition.

Do the indexing for each node in Partition1 & partition2

Balance the partitions such that the tolerance limit is satisfied

**Figure 3.5:** Flow chart to determine the initial partitions

Initialize cutsize to 0

Find the length of Partition 1 & Partition 2

Check for the interconnections b/w these two partitions

Is

Interconnection=

‘1’?

No change in cutsize

No

Increment cutsize by 1

Go through all the iterations to check interconnection b/w each and every node

Yes

Inverse of mincut=1/mincut

**Figure 3.6:**Flowchart for finding the interconnections between two partitions

Find the size of Partition1 and Partition2

Initialize *area* and *m* to zero

Check for the interconnections b/w *nodes* of Partition1 and Partition2

Is interconnection ‘1’?

Add *area* of corresponding *node* from both the partitions and increment *m* by 1

Make no change to *area* and *m*

No

Yes

Go through all the iterations to check interconnection b/w each *node*

Calculate total *area* and take its square root to find the value of a.

Take α=1.1, β=2.0,γ=0.5,

Unit-resistance=0.115,

Unit-capacitance=0.00015.

Find average length using formula



Calculate wire resistance and capacitance

Finally, calculate delay-using formula



A

Inverse of delay = 1/*Delay(e)*

**Figure 3.7:**Flowchart to find delay for given partitions

No

Yes

No

Yes

Calculate the number of *nodes* of partition 1 & partition 2, and then initialize *clock*

Initialize st matrix of size with *nodes\*clocks* with all elements = 0

Check for st matrix with *nodes\*clocks* of Partition1 and Partition2

Is

st matrix (i,j) complete for all *nodes*& given *clock* signals ?

Update st matrix (i,j)

Increment *clock* signal by 1

Is

*Clock* signals are finished.

Increment no. of *nodes* by 1

**Figure 3.8:**Flow chart to calculate sleep matrix

*Clock* signal is increment by one

Sleep time = Previous sleep

Sleep time increment by one

No

No

Yes

Yes

Check whether all the *nodes* are off?

Is

Numbers of *clocks* are finished?

Initialize sleep time and set to zero

Call the *sleep matrix* function to get the st matrix function and number of *clocks*

Calculate the length of partition 1 & partition 2

i.e. l1 and l2

Start with partition 1 and check for *clock* value, initially set for 1

Repeat the procedure for partition 2

Return S=Sleep time

**Figure 3.9:** Flowchart to calculate the sleep time

Create any no. of particles using100\*rand to 1000\*rand

Divide this number by length of either partition

No

Take remainder after dividing and round off this value towards the highest integer

Use this value as node to be swapped among the initial partitions

**Figure 3.10:** Flowchart for swapping the nodes between partitions

Take position1, position2 and node to swap as input to this function

Temporarily store the *node* to be swapped in part1 & part2 variables.

Select the corresponding node from two partitions & temporarily, store in temp1 and temp2.

Put these temp1 and temp2 values in opposite partitions for actual swapping to take place.

Resultant is the newpartition1 and newpartition2.

**Figure 3.11**: Flowchart to create new particles

Label a particle having maximumCcvalue as and all other as.

Calculate inverse of mincut &inverse of delay of initial partitions using function files ‘mincut’ , ‘delay model’ and ‘sleep test’

Use this inverse of mincut, inverse of delay and sleep time value in combined fitness function

Call ‘initial-position’ function to generate initial partitions

Calculate Combined fitness function

Take Ys = Yc = 0.5; Get initial value of Cc

Randomly generate particles from the initial partitions using the concept of nodes swapping

Store all these Cc, mincut, delay and sleep time values in an array

Enter the netlist file and read its adjacency matrix

Print fitness, mincut, and delay and sleep time values before applying PSO

Generate an array for weight values in the range of 0.2-0.8.Take c1=c2=2

Apply particle velocity & particle position equations for all the particles generated randomly earlier.

Correspondingly, get new values of fitness function, mincut , delay and sleep time for all the particles

No

Yes

Update values of particles during each iteration

Retain previous values in array

Repeat same procedure for all the particles

Iterate back the positions corresponding to better fitness values to input of PSO.

Is fitness function

(after PSO) > fitness function (before PSO)?

Print all the fitness values, mincut, delay and sleep time values after applying PSO

**Figure 3.12:** Flowchart for maximizing sleep time and reducing the cutsize& delay in Bipartitioning using PSO

1. **Results And Discussion**

All of the parameters were adjusted concurrently in this study since the objective of our work is to minimize the interconnections, latency and maximize intra partition sleep time. Before using the suggested PSO technique, the interconnections, delay and sleep duration were determined; the proposed method was then applied to simultaneously optimize all of the parameters with a 50 percent weight to mincut and sleep. The algorithm's time complexity was also estimated. For this, several netlists provided by ISPD’98 benchmark containing 10-60 nodes were employed. All of the aforementioned netlists were subjected to the PSO algorithm; the results of a select few (showing the best performance) are presented in Table 3.1. The plot between different parameters is shown in Figure 3.13 – Figure 3.19. From Table 3.1, it can be observed that the suggested technique has been thoroughly tested, with results achieved on circuit netlist files of varied sizes. The suggested method outperforms other approaches in terms of overall size ranges. In Table 3.2 the mincut result is compared to that obtained by GA [7]. Our algorithm also takes lesser runtime as compared to GA [7]. The runtime are shown in Table 3.3.

**Table 3.1**: Results on Mincut, delay and sleep time results for various circuits using the PSO technique

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.N.** | **File Name** | **Before optimization** | | | **After optimization** | | | **Percentage improvement** | | | |
|  | **Min-cut** | **Delay**  **(μs)** | **Sleep** | **Min-cut** | **Delay**  **(μs)** | **Sleep** | **Min-cut** | **Delay** | **Sleep** | **Proportion of Energy Saved** |
| 1 | Spp\_N10 Series | 6 | 2.33 | 22 | 4 | 0.24 | 40 | 33.33 | 89.83 | 81.81 | 8.64 |
| 2 | Spp\_N20 Series | 14 | 20.37 | 37 | 9 | 7.83 | 49 | 35.71 | 61.55 | 32.43 | 5.76 |
| 3 | Spp\_N25 Series | 27 | 62.02 | 48 | 18 | 32.16 | 63 | 33.33 | 48.14 | 31.25 | 7.2 |
| 4 | Spp\_N30 Series | 14 | 20.37 | 32 | 6 | 2.33 | 35 | 57.14 | 88.57 | 9.38 | 1.44 |
| 5 | Spp\_N40 Series | 25 | 55.08 | 23 | 18 | 32.16 | 29 | 28 | 41.62 | 26.09 | 2.4 |
| 6 | Spp\_N50 Series | 38 | 102.29 | 22 | 28 | 65.54 | 44 | 26.32 | 35.92 | 100 | 10.56 |
| 7 | Spp\_N60 Series | 52 | 157.08 | 15 | 45 | 129.29 | 36 | 13.46 | 17.69 | 140 | 10.05 |

**Table 3.2**: Comparison of Mincut results for various circuits using the PSO technique with GA

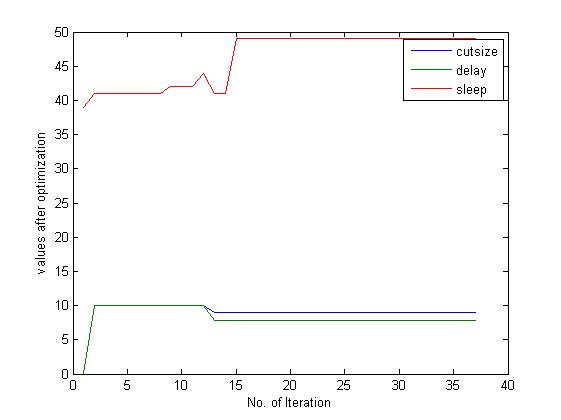
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No.** | **Circuit series** | **Number of nodes** | **Minimum cut using GA[7]** | **Minimum cut obtained by Proposed PSO** | **% Improvement** |
| 1 | Spp\_N10 Series | 10 | 4.05 | 4 | 1.23 |
| 2 | Spp\_N20 Series | 20 | 7.12 | 9 | 26.40 |
| 3 | Spp\_N25 Series | 25 | 8 | 8 | 0 |
| 4 | Spp\_N30 Series | 30 | 7.8 | 6 | 23.08 |
| 5 | Spp\_N40 Series | 40 | 8.5 | 8 | 6.25 |
| 6 | Spp\_N50 Series | 50 | 10.75 | 10 | 7.5 |
| 7 | Spp\_N60 Series | 60 | 11.4 | 11 | 3.51 |

**Table 3.3:** Runtime (in seconds)for the proposed PSO

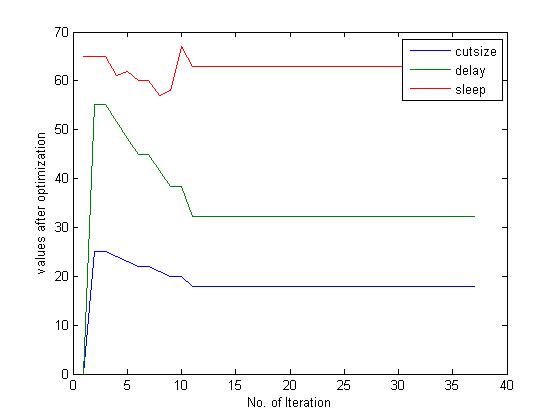
|  |  |
| --- | --- |
| **Circuit series** | **Runtime (s) Proposed PSO** |
| Spp\_N10 Series | 2.56 |
| Spp\_N20 Series | 5.32 |
| Spp\_N25 Series | 8.67 |
| Spp\_N30 Series | 11.45 |
| Spp\_N40 Series | 14.23 |
| Spp\_N50 Series | 17.01 |
| Spp\_N60 Series | 20.03 |



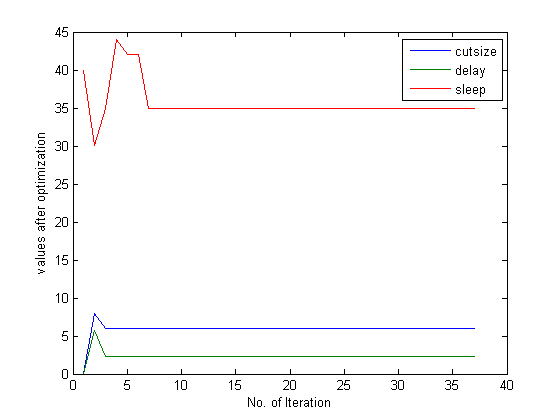
**Figure 3.13:** Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N10 Series



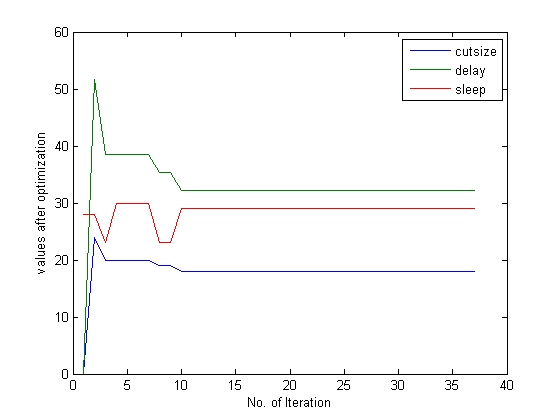
**Figure 3.14:** Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N20 Series



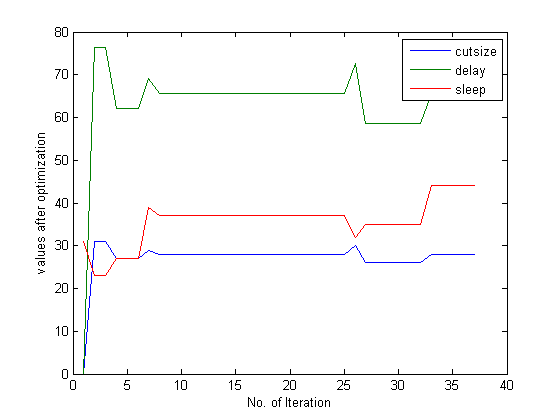
**Figure 3.15:** Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N25 Series



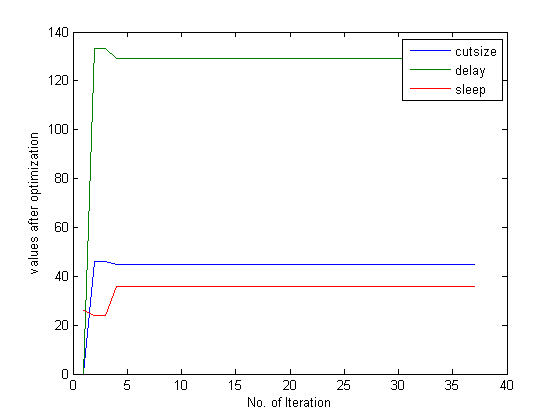
**Figure 3.16:** Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N30 Series



**Figure 3.17:** Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N40 Series



**Figure 3.18:** Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N50 Series

****

**Figure 3.19:** Plot between mincut, delay, sleep time and no. of iterations for netlist Spp\_N60 Series

The results of mincut, delay and sleep time for various netlist files are shown in Figures 3.13 to 3.19. Table 3.1 shows the minimal cut sizes, delay and sleep values that were obtained using 20 particles and 35 iterations. The value of ranges from 0.2 to 0.8, whereas *c1* and *c2*are both set to 2. Table 3.2 compares the minuct obtained by the GA [7] with our algorithm and found that an average of 9.71% improvement has been achieved.

As seen in Figures 3.13 to 3.19, as the iteration progresses, the interconnections/delay decreases and the sleep duration increases, eventually becoming constant. It implies that there is no more room for optimization and that the connectivity delay and sleep time are the final findings. When the fitness function is optimized as a multi-objective fitness function, the connection, delay and sleep time are all optimized at the same time.

1. **Summary**

This chapter provides a novel approach to bipartition a larger system into smaller subsystems. The PSO technique has been successfully applied to achieve the optimized values of multiple objectives at the same time. When the inter partition delay and sleep time are optimized as a multi-objective fitness function with 50% weight to sleep and mincut, the two quantities, numbers of mincut and delay strive to decrease while sleep time tries to maximize, eventually becoming stationary as the number of iterations grows. As can be observed from Table 3.1, there has been a significant improvement in mincut, latency, sleep time and power saving. There is an average 32.47% improvement in mincut, an average 54.76% reduction in delay, an average 60.14% improvement in sleep duration and a reduction of power consumption by an average 6.58%. The suggested approach has a time complexity of approximately.

**CHAPTER 4**

**Chapter 4**

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

In the integrated circuit (IC), designing floorplanning is an important phase in the process of obtaining the layout of the circuit to be designed. The floorplanning determines the performance, size, yield and reliability of VLSI ICs. The obtained results in this step are necessary for the other consecutive process of chip designing. VLSI floorplanning from the computational point of view is a non-polynomial hard (NP-hard problem) and hence cannot be efficiently solved by the classical optimization techniques. In this chapter, we have proposed a metaheuristic approach to address the problem by using the parallel particle swarm optimization (P-PSO) technique. The P-PSO uses a new greedy operation on the sequence pair (SP) to explore the search space to find an optimal solution. Experimental results on the Microelectronic Centre of North Carolina and Gigascale Systems Research Center benchmark shows that the applied parallel PSO (P-PSO) may be used to produce an optimal solution.

1. **Introduction**

Floorplanning is the second step of the physical design process, after the partitioning process is completed; the exact position of each block is to be determined on the chip. The floorplanning determines the performance, size, yield and reliability of VLSI ICs. The obtained results in this step are necessary for the other consecutive process of chip designing. VLSI floorplanning from the computational point of view is a non-polynomial hard (NP-hard problem) and hence cannot be efficiently solved by the classical optimization techniques. As the circuit size increases, the search space also increases simultaneously thus making it impossible to find a global solution. An optimum floorplanner is capable of determining the location of the modules on the chip to be designed. In the physical design process after the circuit partitioning, the position of different blocks on a die is determined in the floorplanning process. Such circuits with more complexity are to be designed hierarchically. Circuit placement within each level of the hierarchy is a complex block placement problem. A good block placement should not only minimize the interconnection delay but also minimize the overall chip area by minimizing the dead space, which is crucial in determining circuit performance in deep submicron designs. Although block placement is a classical problem with many previous algorithms [28-31], it remains to be a hard problem [58, 59]. Floorplanning takes in some of the geometrical constraints in the physical design process and it is a technique for placement arrangement of various functional blocks, called modules. Some design constraints such as wirelength and the overall white space (or dead space) isto be minimized to enhance the performance of the integrated circuit [60-66].

1. **Problem Statement**

Let be a collection of blocks, with N equaling the total number of modules. Assume that the block is represented by where &are width and height of module.

The goal of the floorplan is to place each module in such a way that no two modules overlap and the space filled by the modules represented by set B is as little as possible.

In floorplanning, there are two kinds of modules:

1. *Hard module:*a module's dimension is predetermined*.*
2. *Soft module:*A module's form (width and height) can be changed while the area (A) remains constant (within the permitted aspect ratio established by the benchmark circuits).
3. **The floorplanning techniques**

Various studies have recognized two types of floorplanning structures: slicing and non-slicing. The majority of the floor layout is completed without the use of slicing. A variety of techniques may be used to detect non-slicing structures, such as B-tree, SKB-Tree, sequence pair and many more. One of the non-slicing floorplan methods is the sequence-pair approach. Although this method has drawbacks, such as a high computational cost in the order of O(n2) and a difficult transition from floorplan to appropriate placement, It offers several advantages, including a larger search field for finding the best solution. Sequence-pair technology is also extremely adaptable and can deal with slicing structures. Several studies on floorplanning have been completed to optimize the area and wirelength limits. Saurabh N. Adya and Igor Markov [28] advocated utilizing Simulated Annealing across the sequence pair to tackle the problem and demonstrated that fixed-outline floorplans are more difficult to build than standard floorplans. Lin & Hung [32] tested their technique of allowing motions based on the idea of floorplan slack. Soft modules were also used to implement this fixed-outline floorplan. A soft module has a fixed area and an aspect ratio that may vary within a defined range handle the problem of not employing the hard module (a module whose size and shape are fixed) [48], however the area achieved in that instance is substantially bigger. The authors effectively solved the problem by optimizing the floorplan's wirelength.

1. **Sequence-pair**

VLSI floorplanning consists solely of the arrangement of non-overlapping rectangles. The arrangement of blocks on a chip is divided into two types: slicing floorplan and non-slicing floorplan.The sequence-pair (SP) approach was used to investigate the non-slicing floorplan in this work.Sequence pair is a technique for packing bocks that use a pair of modules known as sequences.The ability to have a limited solution space is essential for successful optimization. Murata et. al.[33]has demonstrated that the SP's searching space results in an effective rectangular packing of the modules.

Tang et.al. [49] Proposed a method called Fast Longest Common Subsequence (fast LCS) to encode a sequence pair to its corresponding floorplan. The first order of sequence-pair is formed by arranging the lines drawn from the chip's southwest corner to its northeast corner in a linear fashion.These are non-intersecting, non-overlapping lines that each pass through one module. The second-order sequence pair () is obtained by drawing similar lines from the chip's southeast corner to its northwest corner.

Fast LCS is a quick and easy way to calculate LCS for a given sequence pair, where n represents the number of items and the weights is not limited to 1 or integers like LCS.

Slide2.TIFSlide1.TIF

**Figure 4.1:**The sequence-pair for the specified placement is (132645, 245136)

If the blocks are and the input sequence pair is , then both and are permutations of The array , of block positions is used to store the coordinates of block based on the their weight vector, which corresponds to the width or height of block . The array is created to be and the length array represents the length of candidates for the longest common subsequence. The following is the algorithm:

**Pseudo code of Fast LCS [49]**

1. Initialize Match Array
2. Initialize Length Array

There are three distinct sorts of procedures that may be used to transform a sequence pair to another, which are as follows:

Op1: Swap the names of two modules in any of the two sequences.

Op2: Swap the names of two modules in each sequence.

Op3: Rotate a module as the third option.

To demonstrate the impact of a perturbation on a sequence pair, consider an example with seven modules of dimension as and the initial sequence pair as (.

2475136,7216534902(2,6).tif6475132,7612534(110).tif6475312,7612534 (100).tif2475136,7216534902(3 rotated).tif

OP1: Swapping modules 1 & 3 only in S1

OP2: Swapping modules 2 & 6 both in S1&S2

OP3: Rotating the module 3

(d)

(c)

(a)

(b)

**Figure 4.2: A**Floorplan for the Initial sequence pair(. **b** Floorplan for the sequence pair (after exchanging module 1 & 3 only in Sequence). **c** Floorplan for the sequence-pair (After exchanging module 2 and 6 in both Sequences. **d** Floorplan forthe sequence pair, when module 3 is rotated

Figure numbers 4.2 shows the effect of these operations. The floorplan's dimensions change from to to torespectively. This example demonstrates how allowing these three valid operations on a sequence pair can significantly alter the floorplan area.

1. **Present work**

We used a multidimensional PSO approach to solve the issue optimally. PSO is a general-purpose global stochastic meta-algorithm.PSO was first proposed by Kennedy and Eberhart [67] in 1995 as a population-based and stochastic search optimization approach. A fish school or a flock of birds can be mathematically modeled in which each particle (or living thing) has a true desire to reach its goal (for example, food). PSO follows in the footsteps of previous evolutionary algorithms (EAs) [68], Genetic Algorithms (GA) [69], Genetic Programming (GP) [70], Simulated Annealing (SA) [71] and Evolution Strategies (ES)[72]. All evolutionary algorithms are common in the sense that they are all based on nature and try to stay away from the local optimal solution; nevertheless, obtaining an optimal solution is never guaranteed.

The fundamental PSO method generates a swarm of particles, each particle representing a potentialresult to the optimization problem; searches throughout the search space are generated.The particles are initially given a random velocity and position in the search space. Each particle's goal is to find the best global solution to the optimization problem. Every individual particle in the swarm has influenced by the movement of each particle in the search space (referred to as in [67]), as well as the global best solution obtained (the so-called in [67]).The velocity of the particle in all the iteration is determined by the previous social best (position of the particles as the social component), individual best ( as the cognitive component), location of the swarm and the previous. Both the social and cognitive components influence the position of the individual particle in the next iteration.

The social interaction behavior shown by the flying birds and fish schools inspired Particle Swarm Optimization (PSO).It is a population-based self-adaptive and powerful stochastic optimization technique.In a multidimensional space, it is essentially the search for the optimal solution. A PSO generates a population of particles and the cost function to be optimized is defined by their current location. The particle's new location and velocity are adjusted in each iteration based on the sum of the effect of each particle's current velocity, the particle's distance from its own best performance () thus far and the distance of the leading particle, which is now the global best (). The following equations developed by Kennedy and Eberhart [67] regulate the particle's position and velocity:

(4.1)

(4.2)

Whereis the particle's previous best position yielding the highest fitness value and is the swarm's historical best position and&are two acceleration constants used to calculate the relative importance of and . The dimension of the search space is represented by &, which are random numbers uniformly distributed in.

To balance the local and global search throughout the optimization process, the concept of inertia weight was introduced to the original PSO by Kennedy and Eberhart [67]. Since then, nearly all PSO versions have incorporated the idea of inertia weight. They demonstrate that a high inertia weight is better for global search while a low inertia weight is better for local search. They proposed using a linearly changing inertia weight across generations to balance the local and global search, which resulted in a significant improvement in performance when searching for the optimal value of the fitness function.It is calculated as follows:

(4.3)

Where& are the start and ultimate values of the inertia weight, is the maximum iteration and is the current iteration.

PSO (as a stochastic search algorithm) has some significant disadvantages when compared to other optimization techniques in the multi-dimensional (MD) solution space. Some of the disadvantages are as follows:

* 1. Any stochastic evolutionary algorithm is controlled by the system parameters in the optimization process to which the optimization algorithm is to be applied; the first disadvantage of the PSO stems from this fact, where variations in the problem's parameters can lead to a significant performance shift.
  2. The global best (gbest) in the PSO directs the rest of the swarm, which could lead to the formation of identical particles with less variety. The likelihood of being caught in the local optimal solution increases because of this PSO phenomenon [73].When the search space is large or the issue to be optimized necessitates concurrent computations of the target solution, PSO frequently converges to an early solution(s).
  3. The basic PSO entails selecting only one swarm at a time and it can only be used in a search space with a defined dimension. In many optimization situations, however, the optimal dimension is unknown; additionally, several swarms may be required to solve the problem properly.

The parallel PSO (P-PSO) technique effectively addresses these issues by avoiding the premature convergence problem. Furthermore, Parallel PSO eliminates the need to determine the dimensions of the solution space in advance.

1. **The Parallel-Particle Swarm optimization**

The search is guided towards the best solution by the two acceleration components & (the cognitive component and the social component).According to Kennedy & Eberhart [67], the relatively high value of the cognitive and social components will result in excessive wandering around the search space, while the opposite will end up causing the particle to move to an untimely local minimum. In order to have a successful optimization strategy, in the early stages of the search, the particle should roam around the whole search space and then focus on the local optimum values in the latter stages. This problem might be solved by mixing a strong social component with a low cognitive component early on and a low social component with a high cognitive component later on. The two components may be taken as:

(4.4)

(4.5)

Where& are constant. To wander across the whole search space, at first, a high value of the social component and a low value of the cognitive component are required, followed by a low value of the social component and a high value of the cognitive component.

The PSO equation may be recreated by incorporating these adjustments and adding linearly variable inertia weight:

(4.6)

(4.7)

The principles that govern particle movement in the search space of a problem may also be considered as a model of human social behavior, in which people change their ideas and attitudes to fit in with their peers [67].

The so-called swarm in PSO is made up of a collection of particles. A particle's location denotes a potential solution to the optimization problem at hand, which is represented by an objective function, in this example, the floorplan's minimal area. At every time step has a position and a velocity   associated with it. The best place that particle (in regard to) has ever visited until time step is called the particle's personal best. The particle's location and velocity are connected with it. The original form of PSO works well when the particle (or swarm) updates its current location inside a single search region, but it fails to give a way to transition between multiple search spaces if the search space encompasses additional one-dimensional space.

In the current situation, we can only get the perturbation operation Op3 using this method (explained in section 4.2). Op1 and Op2 demand the switching of two particles inside the same search space; to do these operations, two distinct sets of particles with the same number of particles are defined parallelly. The two parallel PSOs run at the same time to search inside their respective search areas, yielding two distinct particles, one for each search space. The particles exchange locations inside their separate search spaces to complete the operations indicated in Op1 and Op2, while the particle with the best positions in each of the parallel PSO spins its dimension to perform Op3. Only one perturbation may be performed at any one moment in the traditional PSO, which might lead to sluggish convergence. The proposed approach successfully overcomes this issue and all three procedures are performed in everyiteration to obtain the minimum area (or the objective function).

**Algorithm forthe Proposed P-PSO to minimize the floorplan area**

1. Initialize the two different sets of PSO parameters parallelly with the same number of particles corresponding to each node of the sequence pair
2. Randomly initialize position vector of each pair of particles , where j is used to distinguish between two different sets of PSO.
3. Generate initial velocity vector for each pair of particles
4. Using the objective function(area of the chip in this case), determine the fitness value of each pair of particles
5. set t and in the swarm for both pair of particles
6. update the inertia weight
7. update
8. (here is used to distinguished between the two sets of swarms)
9. update the velocity vector
10. update the position vector

**Op1:** swap the positions of the modules corresponding to in and calculate the fitness function as.

**Op2:** swap the positions of the modules corresponding to in both the sequence and calculate the fitness function as.

**Op3:** rotate the modules corresponding to and calculate the fitness function as

Chose the best fitness among

1. is better than the

Update

1. **if**  is better than the

Update

1. **Parameters of the proposed algorithm**

The values of the parameter are crucial since it ensures that the suggested PSO algorithm balances exploration and exploitation. is a cognitive parameter that highlights personal best performance while the social parameter prioritizes the global best. Despite the lack of a well-defined procedure for selecting these numbers, several scholars have underlined the importance of maintaining and values so that in which a good outcome is obtained in a different environment. To keep the total of cognitive and social factors at 4, we experimented with setting the values of to respectively, which resulted in an improved solution.A linearly varying inertial weight of 0.1 to 1 is used.A smaller inertia weight value prioritizes the search in the local best's neighborhood, whereas a larger value global best promotes the search globally.As a result, early on in the process, local search is strong and as the operation progresses, global search becomes more sophisticated.

1. **Experimental results**

The MCNC Benchmark and the GSRC Benchmark are used to analyze and compare the proposed strategy. The MCNC benchmark includes five (05) soft and five (05) hard modules, whereas the GSRC benchmarks include six (06) soft and six (06) hard modules.A soft module has an area and a range of aspect ratio, so it may have many shapes, however, a hard module simply allows rotation, and thus we do not have control over the shape of the module. We used the sequence pair to test the hard module in our experiment.

Using the MCNC benchmarks, the proposed P-PSO algorithms were compared to other approaches [28, 35, 51, 52, 53 and 86] in terms of the area gained.Whereas the results obtained on GSRC benchmark are compared with the algorithms in [29, 54, 55 and 87]. In [28], a floorplanning program, parquet is combined with the sequence Pair technique and fixed-outline floorplanning.A tool that is commercially available in, BloBB[29]  is utilized for both slicing and non-slicing floorplans. In [35], the initial module placement strategy was B\*-tree and the optimization mechanism was SA.To determine the area of soft modules in [51], the slicing floorplan was developed using PSO.Hybrid Simulated Annealing (HSA) is used to build a non-slicing B\*-tree representation in [52]. Based on sequence pair representation, [54]implements a quick approach for addressing alignment and cluster limits. The Ant system, together with simulated annealing, is utilized to create a 3D floorplan in [55].[53] Presents the DOTFloor (Diffusion Oriented Time-improved Floorplanner) technique, which is a time-efficient and reliable floorplan algorithm based on a SA (Simulated Annealing) engine. Harmony Search Algorithm which is inspired by the music playing phenomenon of musicians is implemented to achieve the area optimization is reported in [86]. A nature inspired Lion Optimization Algorithm has been investigated by [87].

**Table 4.1:**Characteristics of MCNC Benchmark circuits

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Circuit** | **Modules** | **Nets** | **I/O Pads** | **Pins** | **Area(mm2)** |
| apte | 9 | 97 | 73 | 287 | 46.5616 |
| xerox | 10 | 203 | 2 | 698 | 19.3503 |
| hp | 11 | 83 | 309 | 309 | 8.08306 |
| ami33 | 33 | 123 | 42 | 522 | 1.1564 |
| ami49 | 49 | 408 | 22 | 953 | 35.4454 |

**Table 4.2:**Characteristics of GSRC Benchmark circuits

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Circuit** | **Modules** | **Nets** | **Pins** | **Area(mm2)** |
| n10 | 10 | 118 | 248 | 2211679 |
| n30 | 30 | 349 | 723 | 208240 |
| n50 | 50 | 485 | 1050 | 182962 |
| n100 | 100 | 885 | 1873 | 179501 |
| n200 | 200 | 1583 | 3599 | 175696 |
| n300 | 300 | 1893 | 4358 | 273170 |

1. **The validity of the proposed algorithm over classical PSO**

We experimented with six numbers of GSRC and five numbers of MCNC circuits to see how effective the suggested modified PSO is compared to the traditional PSO. In the two benchmarks, the number of blocks varies from nine to up to 300. The suggested technique and the traditional PSO are both based on sequence pair floorplans and the identical beginning sequences. are the values of the traditional PSO parameters. On the MCNC and GSRC benchmarks, Tables 4.3 and 4.4 compare the results with traditional PSO and several alternative techniques.

Based on equivalent solution quality, Table 4.5 and Table 4.6 shows the percentage improvement of dead space as compared to PSO [51] and ACO-SA[55] for MCNC and GSRC benchmarks respectively. Table 4.7 and Table 4.8 compares the runtime of the traditional PSO [51] for MCNC benchmark and with Hierarchical [29]foe GSRC benchmark with the suggested approach. (dead space percentage in the experiment). The number of particles in both the traditional PSO and the proposed P-PSO are set to the same number and the runtime is compared using the same number of iterations.We compared the amount of dead space achieved by the suggested algorithm to that of the traditional PSO and discovered that the proposed method achieves lesser amount of dead space in a considerably shorter period.

**Table 4.3:** Area optimization results for MCNC Benchmark Circuits (in mm2)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | | **SP (Parquet-1)**  **[28]** | **SA with B\*-tree[35]** | **PSO[51]** | **HSA[52]** | **DOTFloor [53]** | **PSO-GA [86]** | **Ours**  **(P-PSO)** |
| apte | Area (mm2) | 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 (mm2) | 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 (mm2) | 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 (mm2) | 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 (mm2) | 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 |

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

**Table 4.4:** Area optimization results for GSRC Benchmark Circuits (in mm2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | | **Hierarchical [29]** | **SA with Cluster Constraints [54]** | **Hybrid ACO-SA[55]** | **LOA [87]** | **Ours**  **(P-PSO)** |
| n10 | Area (mm2) | ---- | ---- | ---- | 2.3 | 2.24 |
| DS (%) | ---- | ---- | ---- | 3.91 | 1.34 |
| n30 | Area (mm2) | ---- | ---- | ---- | ---- | 2..18 |
| DS (%) | ---- | ---- | ---- | ---- | 4.59 |
| n50 | Area (mm2) | 2.06 | ---- | ---- | 2.15 | 2.03 |
| DS (%) | 11.65 | ---- | ---- | 15.35 | 10.34 |
| n100 | Area (mm2) | 1.92 | 1.88 | 1.8 | 2.08 | 1.84 |
| DS (%) | 8.38 | 5.41 | 0.56 | 11.17 | 2.23 |
| n200 | Area (mm2) | 1.91 | 1.85 | 2.0 | 1.97 | 1.79 |
| DS (%) | 6.77 | 4.79 | 12.5 | 13.94 | 2.72 |
| n300 | Area (mm2) | 2.97 | 2.93 | 3.4 | 3.07 | 2.81 |
| DS (%) | 8.08 | 6.83 | 19.71 | 11.07 | 2.85 |

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

**Table 4.5:**Area (in mm2) comparison of PSO and proposed algorithm for MCNC benchmark circuits

|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **PSO[51]** | **Ours (P-PSO)** | **% Improvement** |
| apte | 47.31 | 46.74 | 1.21 |
| xerox | 20.20 | 19.52 | 3.37 |
| hp | 9.5 | 8.91 | 6.21 |
| ami33 | 1.28 | 1.17 | 8.59 |
| ami49 | 38.8 | 36.33 | 6.37 |

**Table 4.6:**Area (in mm2) comparison of ACO-SA and proposed algorithm for GSRC benchmark circuits

|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **ACO-SA[55]** | **Ours (P-PSO)** | **% Improvement** |
| n10 | 2.3 | 2.24 | 2.61 |
| n50 | 2.15 | 2.03 | 5.58 |
| n100 | 2.08 | 1.84 | 11.54 |
| n200 | 1.97 | 1.79 | 9.14 |
| n300 | 3.07 | 2.81 | 8.47 |
|  | | |  |

**Table 4.7:**Runtime (in seconds) comparison on MCNC Benchmark for the proposed P-PSO

|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **Runtime**  **(in seconds) PSO [51]** | **Runtime**  **(in seconds)**  **Proposed**  **P-PSO** | **% improvement** |
| apte | 5.72 | 6.31 | -10.31 |
| xerox | 8.1 | 6.988 | 13.73 |
| hp | 6.25 | 5.73 | 8.32 |
| ami33 | 18.36 | 17.30 | 5.77 |
| ami49 | 32.14 | 30 | 6.66 |

**Table 4.8:** Runtime (in seconds) comparison on GSRC Benchmark for the proposed P-PSO

|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **Runtime**  **(in seconds)**  **Hierarchical [29]** | **Runtime**  **(in seconds)**  **Proposed**  **P-PSO** | **% improvement** |
| n10 | ---- | 3.81 | ---- |
| n30 | ---- | 15.73 | ---- |
| n50 | ---- | 26.47 | ---- |
| n100 | 125 | 73.67 | 41.06 |
| n200 | 522 | 316.2 | 39.43 |
| n300 | 1007 | 568 | 43.59 |

apte.tif

**Figure 4.5:** Floorplan for apte (0.40% dead Space)

HP.tif

**Figure 4.6:** Floorplan for xerox (0.87% dead Space)

xerox.tif

**Figure 4.7:** Floorplan for hp (8.91% dead Space)

ami49.tifami33.tif

**Figure 4.8:** Floorplan for ami33 (1.7% dead Space)

**Figure 4.9:** Floorplan for ami49 (2.45% dead Space)

n30.tifn10.tif

**Figure 4.10:** Floorplan for n10

(1.34% dead Space)

**Figure 4.11:** Floorplan for n30

(4.59% dead Space)

n50.tifn100.tif

**Figure 4.12:** Floorplan for n50 (10.34% dead Space)

**Figure 4.13:** Floorplan for n100 (2.23% dead Space)

n200.tifn300.tif

**Figure 4.14:** Floorplan for n200 (2.72% dead Space)

**Figure 4.15:** Floorplan for n300 (2.85% dead Space)

1. **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 showsthat 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.

**CHAPTER 5**

**Chapter 5**

**Placement for Area & Wirelength Optimization using hybridized PSACO technique**

The process of identifying an appropriate physical position for each cell in the netlist is known as placement. Placement does not just place the standard cell available in the synthesized netlist it also optimizes the design. Various factors such as timing requirements of the system, interconnect length is to be taken into account while assigning a location to any cell. In this chapter, we have proposed a hybrid PSACO approach to optimize the area and wirelength of a layout simultaneously. The results are obtained and compared with the benchmark circuits available with MCNC and GSRC.

1. **Introduction**

The third stage in the physical design cycle is placement. A poor placement may lead to consumption of larger areas that leads to performance degradation. It may also sometime leads to difficult routing task. The inputs to this phase are:

1. A set of blocks of a specific shape and size.
2. The total number of terminals in each block, as well as
3. The linkages between these blocks are specified by the netlist.

The placement phase is critical since ahigh-quality routing cannot improve a badly placed layout;therefore,this phase defines the chip layout's overall quality in terms of performance and area.

1. **Problem Formulation**

Let be a collection of blocks, where ‘n’ is the total blocks. Assume that the block be depicted by*, where*  are the width &height of the module.

Let be the set nets, ‘m’ represents the total number of nets.

Let be the approximatedwirelength of net,, the placement objective is to identify a group of rectangles represented by for each of the block represented by set such that,

1. Each block fits inside the rectangle, which has the dimension.
2. There are no overlaps between the rectangles, that is.
3. takes up the smallest amount of space.
4. The total wirelength as given by is to be kept minimum.
5. **The placement techniques**

In floorplanning, we divide the bigger design into manageable blocks (of standard cells) and macros(memory instances) and arrange it in the chip, while in placement we go deep inside and place the smallest element (digital gate for standard cell and several memory blocks). In floorplanning, the inputs are a set of blocks with the fixed area but having different aspect ratios, while in the placement phase both the size and shape of the block are fixed. The objective of floorplanning is restricted to finding the minimum area of the chip keeping the aspect ratio of the blocks within an allowed range, while in placement the goal is to find the minimum area as well as finding the minimum wirelength of the chip. Thus, both are the same (as both are placement), but the complexity is different so is the terminology.

The technique for the placement is same as that of floorplanning. In this work, we have used the sequence pair technique (explained in detail in chapter 4) for placement also.

1. **Effect of placement on routability**

The topological congestion, often known as rat's nest, approximates the placement area and routability of the placement. Consider the scenario in Figure 5.1 to show routability. Figure 4.2 depicts two possible locations for this example. Figure 5.2(a) shows a considerably smaller rat's nest than Figure 5.2(b) (b). As a result, the location in figure 5.2(a) is more readily routable as compared to the placement in figure 5.2(b).



**Figure 5.1:**A collection of blocks and interconnected networks [1]

**Figure 5.2:**Two distinct solutions to the same problem [1]

(b)

(a)

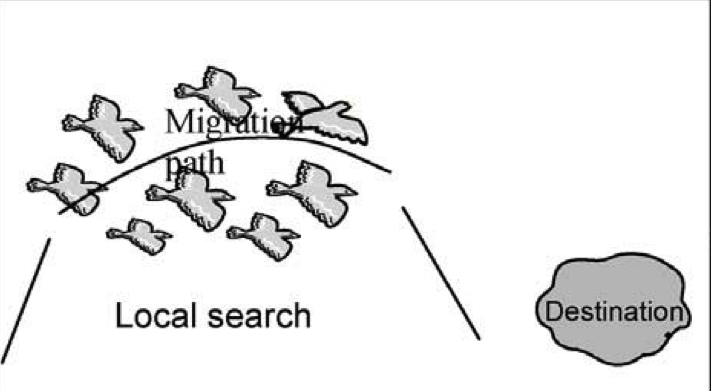
The purpose of the traditional placement technique is to reduce the area occupied by the layout; the problem of multi-objective placement, on the other hand, is significantly more difficult. One of the most challenging aspects of multi-objective optimization is that there is no one best solution for every target in the solution space. Additionally, adopting an ideal solution for one goal may necessitate receiving a suboptimal result for another. Hence, it is difficult to define what constitutes a good answer. In this example, our goal is to reduce the layout area while also reducing the wirelength.However, because the two objectives are so different, maximizing alone does not always indicate that is also reduced and vice versa. That is why, for multi-objective optimization, we use a combination-based formulation: The combined goal will be a scalar combination metric, as shown in the equation below:

(5.1)

Where,

1. **Solution Methodology**
2. **Particle Swarm Optimization**

The PSO, a method based on the choreography of a flock of birds was proposed by Kennedy and Eberhart in 1995 [67]. The goal of this method is to replicate the motions of a flock of birds (Population) on the hunt for food. The method may be thought of as a multidimensional search distributed behavioral algorithm. The behavior of each individual is influenced by either the best local (i.e. within a certain neighborhood) or the best global individual. The approach then employs the population concept as well as a performance indicator similar to the fitness value used in evolutionary algorithms, individual adjustments are also comparable to the usage of a crossover operator. However, this technique uses lying potential solutions across hyperspace (to speed up convergence), which does not appear to be a process employed in standard algorithms.Another significant distinction is that PSO allows individuals to benefit from their earlier experience, whereas other evolutionary algorithms just allow individuals to use the present population as "memory". PSO has long been used for nonlinear and discrete binary optimization.



**Figure 5.3:** Particle Swarm Optimization

1. **PSO Algorithm and its parameters**

PSO, as previously said, replicates bird flocking behavior. Consider the following example: A flock of birds is seeking food at an undetermined location. In the area being searched, there is just one piece of food. The birds are all in the dark about the exact position of the food. They do, however, know how far the meal has traveled in each iteration. To search the meal, following the bird closest to the meal is the most efficient method. PSO applied the lessons learned from the events to the optimization problems.In PSO, each solution is a "bird" that flies through the search area. Particle is the term used for it. All particles have fitness values that the fitness function evaluates in order to maximize particle flight, as well as velocities that guide particle flight. Through the problem space, the particles follow the current optimal particles.

The fitness function and the solution's representation are two essential elements in using PSO to solve optimization issues, as shown in the example above. One of PSO's benefits is that it treats real numbers as particles unlike GA, which convert to binary encoding or employ specific genetic operations. The searching is done in a loop and the stopping criterion is either that the maximum allowed iterations arecompleted or the minimal fallacy limit is reached. The PSO deals with very few parameters that need to be tackled. These parameters and their usual values are as follows:

The average **Number of Particles**is between 20 and 40 [74].

The**Particle’s dimension**indicates the problem that has to be solved.

**Particle range** is influenced by the optimization goal to be solved; may provide multiple ranges for different particle dimensions.

The **constant of inertia weight** w is created in the range [0.2, 1.2] [74].

The **learning factors** are generally the same. Other settings, on the other hand, were utilized in other publications. However, in most cases, c1 = c2 and falls between [0.4, 2] [61]

**Stopping Conditions:**It is the maximum allowable iterationsor the lowest error requirements. The maximum number of iterations can be any value that the user specifies. This halt condition is determined by the problem that has to be solved [75].

PSO begins with a collection of random particles (solutions) and then updates generations to search for optimum conditions. Each particle is updated in every iteration by comparing the most effective remedy (fitness) it has achieved so far () andthe best value that any particle in the population has ever achieved (gbest).

With the following equations (5.2) and (5.3), the particle updates its velocity and positions after selecting the two best values

(5.2)

(5.3)

Where represents particle velocity, represents inertia weight andrepresents the current particle (solution) position., and are defined as previously mentioned. Random numbers between 0 and 1 are represented by and. Learning factorsn most cases equals to 2.The simplest PSO pseudo code is:

**Steps in PSO**

1. **For**every particle initialize the swarm for the solution space
2. **For** every particle calculate fitness function
3. **If** the fitness value is better than the best fitness value in history make it the current the new
4. As the particle, select the particle that has the highest fitness value among all the particles.
5. **For** every particle
6. Using the equation (5.2), determine particle velocity
7. Particle position should be updated based on the equation (5.3)
8. **While**the iterations reach either its maximum value or the minimal allowed error criterion is not met.
9. **Swarms and Particles**

During the paradigm's simplification, it became clear that the behavior of the population of agents is now more like a swarm than a flock. It uses the term in particular in accordance with a research by Millonas, who developed models for artificial life applications and articulated five important swarm intelligence concepts. These are listed as [78]:

1. The first principle states that the population should be able to do simple space and time computations (The proximity Principle).
2. The second concept states that people should be able to adapt to quality variables in the environment (The quality principle)
3. The third principle states that people should not direct their actions through too limited channels (the concept of varied response).
4. The fourth principle is that the population's pattern of conduct should not change every time the environment changes (Concept of stability)
5. The fifth principle is that the population must be able to change behavior mode when the computational cost is justified (adaptability).

It is worth noting that principles four and five are two sides of the same coin. All five principles appear to be followed by the particle swarm optimization idea and paradigm provided. The paradigm is based on n-dimensional space computations performed in a sequence. The population is responding to the quality criteria and . The response distribution between and ensures a wide range of responses. Only when *gbest* changes do the population's state (mode of behavior) change, conforming to the concept of stability. The population is adaptable because it changes in response to changes. As a compromise, the term particle was chosen. PSO is a basic technique that appears to be efficient for improving a broad variety of functions. [67]

1. **Comparisons between PSO and other algorithms**

PSOs have many similarities with evolutionary computation systems such as GA, in which the system is populated with a population of random solutions and the search for optimum solutions is carried out through generations of updating. PSO, unlike GA, lacks evolution operators such as crossover and mutation. Potential solutions, referred to as particles in PSO, fly through the issue space by following the ideal particles at the time.

The benefits of PSO include its simplicity of implementation and the fact that there are very fewparameters to change. PSO has been effectively utilized in a variety of domains, such as parameter optimization, artificial neural network, fuzzy system control andmany more areas where GA may be used. PSO and GA have numerous similarities. Both techniques begin with a randomly formed population group and use fitness values or position vectors to assess the population. Using random methods, both update the population and look for the optimal. Both approaches are not guaranteed to be successful. PSO, on the other hand, lacks genetic operators such as crossover and mutation. The intrinsic velocity of particles causes them to update themselves. They also contain memory, which is necessary for the method. PSO's information exchange method differs considerably from that of genetic algorithms (GAs). Chromosomes communicate with one another in GAs. As a result, the entire population advances in unison toward an ideal region. Only in PSO divulges knowledge to others.

1. **Ant Colony Optimization**

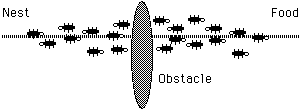
In this, the metaheuristic optimization was inspired by ant behavior. The ant colony optimization algorithm (ACO) is a probabilistic technique for solving problems that may be reduced to finding optimum paths across graphs.Based on the behavior of ants seeking a passage between their colony and a food supply, the technique was created to determine the optimum way through a network. The initial concept was later broadened to tackle a larger class of numerical problems and therefore, other challenges based on various aspects of ant behavior have emerged.

**5.3.2.1 Realants’ behavior**

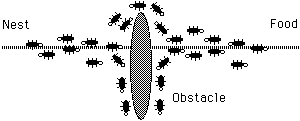
http://iridia.ulb.ac.be/dorigo/ACO/images/Ant1.gifWithout the use of visual cues, real ants can determine the quickest path between the food source and their colony. They may also adjust to environmental changes, such as discovering a new shortest route when the prior one is not available due to a new blockage. Consider Figure 5.4, which depicts ants traveling in a straight path to the food source from the nest:

**Figure 5.4:** Real ants on their way from the nest to the food source.

A pheromone trail is generally recognized to be the principal mechanism by which ants establish and maintain the line. While traveling, ants dump a fixed quantity of pheromone andall the ants preferentially pursue a pheromone-rich path over a pheromone-poor direction. This simple ant activity may be used to illustrate the way theylook for the shortest path that connects a broken chain when an unexpected barrier disturbs the initial path (Figure 5.5).

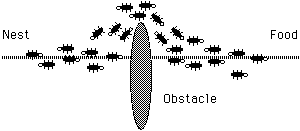
****

**Figure 5.5:** When an obstruction arises on the path, the ants have an equal chance of turning left or right

In reality, once the obstacle appears, the ants immediately in front of it are unable to continue following the pheromone trail and must decide whether to turn left or right. In such a case, we may expect that half of the ants will turn to the right and the others will turn to the left. On the other side of the obstruction, an identical scenario exists (Figure 5.6).

**Figure 5.6:**Pheromone deposited more quickly on the shorter path

It is worth noting that ants who take the shorter route around the barrier will swiftly restore the broken pheromone trail, whilst those who take the longer route would take much longer. As a result, the shorter path will get a higher quantity of pheromone in the time unit, prompting more ants to choose the shorter path. Because of this positive feedback mechanism, all of the ants will eventually choose the shorter path (Figure 5.7).

****

**Figure 5.7:**All the ants are selecting the shortest path.

Discovering the shortest way around the barrier appears to be an emergent characteristic of the interaction between the obstacle's geometry and the ants' dispersed behavior, which is the most exciting component of this positive feedback system.Although all ants travel at roughly the same speed and leave the same pheromone trail, avoiding obstacles on the longer side takes longer than on the shorter side, causing the pheromone trail to build faster on the shorter side.

Ants use their surroundings as a means of communication. They communicate using pheromones, which all explain the current condition of their "job".Only an ant near where the pheromones were deposited has any notion of what they are, therefore the knowledge is restricted. This mechanism is known as "Stigmergy" and it is seen in many social animal groups. A self-organized system is a mechanism for solving an issue that is too complicated for single ants to handle. This strategy is based on positive feedback (a pheromone deposit attracts other ants, who improve it) and negative feedback (a pheromone deposit attracts other ants, who strengthen it) (dissipation of the route by evaporation prevents the system from thrashing).In principle, no path would be chosen if the quantity of pheromones on all edges remained constant over time. Due to feedback, however, a minor alteration on an edge will be exaggerated, allowing for edge selection. The algorithm will go from an unstable state where no edge is stronger than another stable state in which the route is made up of the strongest edges.

**Steps in ACO**

1. Begin with initialization of Ants
2. While termination condition is not true
3. Search solutions
4. Update pheromone
5. Daemon actions
6. End while
7. End process
8. **Artificial ants**

In this study, an artificial ant is an agent that goes from one node to another. It chooses the node to explore based on a probabilistic function of both the edge trail and a heuristic value, which in this case was chosen to be a function of edge length.Artificial ants areasnodes that are connected by edges have many pheromone trails and are close to one another. Initially, artificial ants are deployed in nodes chosen at random. At each time step, they move to a new node and update the pheromone trail on the edges they utilize, a process called a local trail update. When all of the ants have finished a tour, the ant that did the shortest tour alters the margins of the tour by adding an amount of pheromone trail that is inversely proportional to the tour duration (a process known as global trail update). We have taken three concepts from natural ant behavior and applied them to our artificial ant colony:

1. A preference for pathways with a high pheromone concentration,
2. On shorter pathways, the amount of pheromone grows at a faster pace and
3. The trail facilitates ant communication.

Artificial ants have a working memory that can be utilized to memorize nodes that have already been visited and they can assess how far away any nodes are (the working memory is emptied at the beginning of each new tour and is updated after each time step by adding the new visited node). [76]

1. **Hybridized PSACO**

The PSACO (Particle Swarm Ant Colony Optimization) method is used to solve several non-convex optimization problems. Two global search swarm intelligence metaheuristics that operate together are particle swarm optimization (PSO) and ant colony optimization (ACO).PSO is based on the social behavior of flocking birds or schooling fish, whereas ACO is based on the foraging behavior of real-life ants [77]. A simple pheromone-driven method is examined to improve the performance of the PSO technique for optimization of multimodal continuous functions. For multimodal continuous optimization problems, the PSO method is a powerful optimization tool.While the ACO approaches that imitate real-life ant foraging behavior are efficient and robust in tackling combinatorial optimization problems [79].

PSACO is a new particle swarm optimization technique that combines ACO & PSO to optimize multimodal continuous functions. The suggested technique employs PSO for global optimization and the ACO to update particle positions for fast reach the feasible solution space. There are two steps to implementing the PSACO algorithm. In the first step, PSO is used and in the second stage, ACO is used. ACO is a type of local search in which ants employ a pheromone-guided strategy to update locations identified earlier in the process by particles. The inclusion of ACO in the PSACO second stage is based on studies that show that it is effective.

1. PSO is significantly quicker than other evolutionary algorithms in finding decent quality solutions and,
2. If the swarm maintains equilibrium, the development process will become stagnant over time. As a result, as the number of generations increases, so does the inability to enhance the quality of the answers [80].

Initially two different sets of the particles are defined parallelly with the same number of particles. The two parallel PSO runs simultaneously to search within their respective search space and obtains two different particles each corresponding to the two different search spaces. As we get two different particles corresponding to two nodes, we perform the operations as either exchanging the position of the two nodes either in one sequence or in the sequences or rotating the orientation of the nodes to minimize the area and wirelength simultaneously. We store the best sequence after performing the above-mentioned operations. Next to improve the global best; ACO is applied in the second stage as mentioned below:

ACO works in three steps:

1. Construct Ant Solutions (which has been found using proposed P-PSO)

2. Daemon action and

3. Updating of Daemon

An Ant will move from node i to j with probability

(5.4)

Where,

α and β has been taken equal to 1 for our experiment

Amount of pheromone is updated according to the equation

(5.5)

Where,

is the rate of pheromone evaporation, for our experiment we have taken it equal to 0.1

is the amount of pheromone deposited given by,

(5.6)

Where is the cost of kth ant’s tour

The initial value of τ and η is taken as 10 and inverse of the cost matrix (or the *pbest* obtained from the proposed P-PSO).

All the ants that have completed the tour as update the pheromone values;

(5.7)

The number of Ants is taken equal to the number of Particle of P-PSO.

If the fitness value calculated by ACO is better than that of P-PSO then the Ant is chosen, else the particle of P-PSO is chosen. The detailed pseudo code for the proposed algorithm is as:

**Algorithm forthe Proposed PSACO to optimize area and wirelength**

1. Initialize the two different sets of PSO parameters parallelly with the same number of particles corresponding to each node of the sequence pair
2. Randomly initialize position vector of each pair of particles
3. Generate initial velocity vector for each pair of particles
4. Generate the sequence pair for given netlist
5. Using the objective function in equation (5.1), determine the fitness value of each pair of particles (area of the chip in this case)
6. set t and in the swarm for both pair of particles
7. update the inertia weight
8. update
9. update the velocity vector
10. update the position vector
11. **for Particles**

**Op1:** swap the positions of the modules corresponding to in and calculate the fitness function as.

**Op2:** swap the positions of the modules corresponding to in both the sequence and calculate the fitness function as.

**Op3:** rotate the modules corresponding to and calculate the fitness function as

Chose the best fitness among and name it as

1. generate the two dimensional ants using
2. calculate the fitness function using the ants and name it as
3. chose ants
4. chose particles
5. consider set of solutions as particles
6. is better than the

Update

1. **if**  is better than the

Update

Choose the ant

Initialize PSO particles in two-dimension space

Generate the sequence pair for given netlist

Calculate fitness function (Area & wirelength)

Select pbest & gbest

Generate ANTS using gbest.

Choose the particle

Consider set of solutions as particles

Update Velocity & position of each particle

Yes

No

No

Select pbest & gbest by comparing fitness functions

Print gbest position and corresponding fitness function

Yes

Is fitness function (particle) < fitness function (ant)?

Is termination condition true?

**Figure 5.8:** Flow Chart for minimizing area and wirelength using PSO-ACO

1. **Parameters of the proposed algorithm**

The values of the parameter are chosen same as that of chosen in chapter 4.

These values fine-tuned experimentally to achieve a better result and to minimize the runtime.

1. **Experimental Results**

The proposed algorithm is tested and compared over two standard benchmark suites MCNC Benchmark and GSRC Benchmark. The details about MCNC and GSRC benchmark are given in chapter 4. The results with the proposed algorithm are tested using hard modules only where only the rotation of modules is allowed. The positions of the modules are determined by using the sequence pair (SP) technique as discussed in chapter 4. Here, the results of the proposed algorithm are tested with other algorithms such as PSO [51], SA [83], ASOS [84], LOA [87] and BCSA [88] on MCNC benchmark, while with SA [32] and NPE[85] on GSRC benchmark.

1. **The validity of the proposed algorithm over classical PSO**

To test the effectiveness of the proposed PS-ACO we experimented on the six numbers of GSRC and five numbers of MCNC circuits. The number of blocks in the two benchmarks varies from nine blocks to up to 300 blocks. The proposed algorithm has been tested based on the sequence pair floorplan and the same initial sequences. The values of parameters of the PS-ACO are taken as. Table 5.1 and Table 5.2 compare the result with PS-ACO and some other algorithms on MCNC and GSRC benchmark respectively. The algorithm is tested by taking three different weights for the area as Y=0.0, 0.5 and 1.0. The result thus obtained is found to be performing better than other algorithms in terms of overall chip area and wirelength.

**Table 5.1:** Area & Wirelength results for MCNC Benchmark Circuits

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | | **PSO [51]** | **SA [83]** | **ASOS [84]** | **LOA**  **[87]** | **BCSA**  **[88]** | **Ours** | | |
| **Y=0** | **Y=0.5** | **Y=1** |
| Apte | Area (mm2) | 47.44 | 47.31 | 46.32 | 48.03 | 48.05 | 52.1 | 47.3 | 47 |
| Wire (mm) | 263 | 263 | 360 | 316.49 | 474.62 | 326 | 364 | 476 |
| xerox | Area (mm2) | 20.2 | 20.2 | 20.03 | 20.5 | 20.42 | 21.6 | 20.01 | 19.6 |
| Wire (mm) | 477 | 477 | 436 | 337.04 | 385.21 | 182 | 416 | 410 |
| Hp | Area (mm2) | 9.5 | 9.5 | 9.45 | 9.89 | 9.25 | 10.6 | 9.5 | 9.2 |
| Wire (mm) | 136 | 136 | 67.12 | 142.95 | 157.07 | 158 | 64 | 44.5 |
| ami33 | Area (mm2) | 1.3 | 1.28 | 1.29 | 1.23 | 1.23 | 1.26 | 1.2 | 1.17 |
| Wire (mm) | 89 | 69 | 46.26 | 52.22 | 53.55 | 43.5 | 47.4 | 57 |
| ami49 | Area (mm2) | --- | 38.8 | 38.25 | 38.45 | 38.1 | 45.9 | 36.7 | 36.1 |
| Wire (mm) | --- | 880 | 611 | 822.77 | 705.20 | 405 | 592 | 692 |

**Table 5.2:**Area & Wirelength results for GSRC Benchmark Circuits

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | | **SKB with SA [32]** | **SA using NPE [85]** | **Ours** | | |
| **y=0** | **y=0.5** | **y=1** |
| n10 | Area (mm2) | 2.32 | 2.35 | 2.48 | 2.34 | 2.29 |
| Wire (mm) | 16098 | 14224 | 14737 | 18607 | 17770 |
| n30 | Area (mm2) | 2.14 | 2.1 | 2.33 | 2.24 | 2.16 |
| Wire (mm) | 38843 | 35434 | 28981 | 32176 | 37420 |
| n50 | Area (mm2) | 2.07 | 2 | 2.26 | 2.14 | 2.07 |
| Wire (mm) | 79063 | 84020 | 61659 | 62149 | 76398 |
| n100 | Area (mm2) | 1.86 | 1.81 | 1.96 | 1.86 | 1.83 |
| Wire (mm) | 127045 | 12973 | 115448 | 119464 | 148144 |
| n200 | Area (mm2) | 1.83 | 1.87 | 2.06 | 1.99 | 1.86 |
| Wire (mm) | 270*88*6 | 331704 | 335575 | 316482 | 292618 |
| n300 | Area (mm2) | 2.81 | 2.87 | 3.01 | 2.81 | 2.8 |
| Wire (mm) | 425769 | 510535 | 397561 | 424713 | 483215 |

**Table 5.3:** Runtime (in seconds) comparison on MCNC Benchmark for the proposed PSACO

|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **Runtime**  **(in seconds)**  **BCSA [88]** | **Runtime**  **(in seconds)**  **Proposed**  **PSACO** | **% improvement** |
| apte | 3.3 | 7.42 | -124.85 |
| xerox | 7.2 | 7.89 | -9.58 |
| hp | 8.94 | 8.18 | 8.50 |
| ami33 | 34.29 | 19.20 | 44.01 |
| ami49 | 132.3 | 32.36 | 75.54 |

**Table 5.4:** Runtime (in seconds) comparison on GSRC Benchmark for the proposed PSACO

|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **Runtime**  **(in seconds)**  **SKB with SA [32]** | **Runtime**  **(in seconds)**  **Proposed**  **PSACO** | **% improvement** |
| n10 | 20 | 5.76 | 71.20 |
| n30 | 42 | 17.32 | 58.76 |
| n50 | 44 | 29.21 | 33.61 |
| n100 | 107 | 97.56 | 8.82 |
| n200 | 872 | 689.35 | 20.95 |
| n300 | 2440 | 2269.28 | 7.00 |

**Figure 5.9:** Comparison of Area of MCNC benchmark circuits with other algorithms

**Figure 5.10:** Comparison of Wirelength of MCNC benchmark circuits with other algorithms

**Figure 5.11:** Comparison of Area of GSRC benchmark circuits with other algorithms

**Figure 5.12:** Comparison of Wirelength of GSRC benchmark circuits with other algorithms

**Table 5.5:**Area & wirelength comparison of PSO and proposed algorithm for MCNC benchmark (Y=0.5)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Benchmark** | **PSO[51]** | | **Proposed PSACO** | | **% Improvement** | |
|  | Area (mm2) | Wire (mm) | Area (mm2) | Wire (mm) | Area | Wire |
| apte | 47.31 | 263 | 47.3 | 364 | 0.02 | -38.40 |
| xerox | 20.20 | 477 | 20.01 | 416 | 0.94 | 12.79 |
| hp | 9.5 | 136 | 9.5 | 64 | 0 | 52.94 |
| ami33 | 1.28 | 69 | 1.2 | 47.4 | 6.25 | 31.30 |
| ami49 | 38.8 | 880 | 36.7 | 592 | 5.41 | 32.73 |

**Table 5.6:**Area & wirelength comparison of SKB-SA and proposed algorithm for GSRC benchmark (Y=0.5)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Benchmark** | **SKB with SA [32]** | | **Proposed PSACO** | | **% Improvement** | |
|  | Area (mm2) | Wire (mm) | Area (mm2) | Wire (mm) | Area | Wire |
| N10 | 2.32 | 16098 | 2.34 | 18607 | -0.86 | -15.59 |
| N30 | 2.14 | 38843 | 2.24 | 32176 | -4.67 | 17.16 |
| N50 | 2.07 | 79063 | 2.14 | 62149 | -3.38 | 21.39 |
| N100 | 1.86 | 127045 | 1.86 | 119464 | 0.00 | 5.97 |
| N200 | 1.83 | 270886 | 1.99 | 316482 | -8.74 | -16.83 |
| N300 | 2.81 | 425769 | 2.81 | 424713 | 0.00 | 0.25 |

**Figure 5.13:** Placement for

apte (Y=0.0)

**Figure 5.14:**Placement for xerox (Y=0.0)

**Figure 5.17:** Placement for hp (Y=0.5)

**Figure 5.16**: Placement for apte(Y=1.0)

apte(Y=0.5).tifapte(Y=1.0).tif

xerox(Y=0.0).tifapte(Y=0.0).tif

apte(Y=0.5).tifapte(Y=1.0).tif

**Figure 5.15:** Placement for apte(Y=0.5)

**Figure 5.16**: Placement for apte(Y=1.0)

**Figure 5.14:** Placement for xerox (Y=0.0)

**Figure 5.13:** Placement for

apte (Y=0.0)

xerox(Y=1.0).tifxerox(Y=0.5).tifhp(Y=0.5).tif

**Figure 5.19:** Placement for

xerox (Y=1.0)

**Figure 5.18:** Placement for

xerox (Y=0.5)

**Figure 5.17:** Placement for hp (Y=0.5)

hp(Y=0).tif

**Figure 5.20:** Placement for hp (Y=0.0)

hp(Y=1.0).tif

ami33(Y=1.0).tifami49(Y=0).tifami33(Y=0.0).tifami33(Y=0.5).tif

**Figure 5.25:** Placement for

ami49 (Y=0.0)

**Figure 5.24:** Placement for

ami33 (Y=1.0)

**Figure 5.23:** Placement for

ami33 (Y=0.5)

**Figure 5.21:** Placement for hp (Y=1.0)

**Figure 5.22:** Placement for

ami33 (Y=0.0)

n10(Y=0.0).tifn10(Y=0.5).tifami49(Y=1).tifami49(Y=0.5).tif

**Figure 5.26:** Placement for

ami49 (Y=0.5)

**Figure 5.27:** Placement for

ami49 (Y=1.0)

**Figure 5.28:** Placement for

n10 (Y=0.0)

n10(Y=1.0).tifn30(Y=0.0).tif

**Figure 5.30:** Placement for

n10 (Y=1.0)

**Figure 5.31:** Placement for

n30 (Y=0.0)

**Figure 5.29:** Placement for

n10 (Y=0.5)

n50(Y=0.0).tifn50(Y=0.5).tifn30(Y=0.5).tifn30(Y=1.0).tif

**Figure 5.33:** Placement for

n30 (Y=1.0)

**Figure 5.34:** Placement for

n50 (Y=0.0)

**Figure 5.35:** Placement for

n50 (Y=0.5)

**Figure 5.32:** Placement for

n30 (Y=0.5)

n50(Y=1.0).tifn100(Y=0.0).tif

**Figure 5.36:** Placement for

n50 (Y=1.0)

**Figure 5.37:** Placement for

n100 (Y=0.0)

n100(Y=1.0).tifn100(Y=0.5).tif

**Figure 5.39:** Placement for

n100 (Y=1.0)

**Figure 5.38:** Placement for

n100 (Y=0.5)

n200(Y=0.5).tifn200(Y=0.0).tif

**Figure 5.41:** Placement for

n200 (Y=0.5)

**Figure 5.40:** Placement for

n200 (Y=0.0)

n300(Y=0.0).tifn200(Y=1.0).tif

**Figure 5.42:** Placement for

n200 (Y=1.0)

**Figure 5.43:** Placement for

n300 (Y=0.0)

n300(Y=1.0).tifn300(Y=0.5).tif

**Figure 5.44:** Placement for

n300 (Y=0.5)

**Figure 5.45:** Placement for

n300 (Y=1.0)

1. **Summary**

This chapter provides a novel hybridized PSACO algorithm to determine the area occupied by a chip along with the total wirelength required to interconnect different blocks as per the netlist provided. Through investigation, it has been found that the proposed algorithm is able to find a better solution both in terms of area occupancy and wirelength requirements. With the obtained results, we can conclude that the proposed algorithm provides a potential solution when problem is NP-Hard and requires multiple objectives to be optimized at the same time.

The obtained results are compared with those of other algorithms for determining improvement. Table 5.3 shows the comparison results for MCNC benchmark with PSO [51], result shows an average improvement of 2.52% on area occupied and 18.27% on total wirelength required to connect all the nodes as per the netlist provided. Table 5.4 shows the comparison results for GSRC benchmark with SKB-SA [32], it shows an average degradation of 2.94% on area occupied while an average improvement of 2.06% on total wirelength required connecting all the nodes as per the netlist provided.

**CHAPTER 6**

**Chapter 6**

**Simultaneous optimization of the area, wirelength and TSVs ina 3D IC design**

The technology of a three-dimensional integrated circuit (3D-IC) is an emerging approach for improving performance. In comparison to a standard 2-D IC design, which arranges all of the devices on a single planar layer, a 3D-IC stacking of many tiers enables more devices to be placed close together, resulting in the significant area and wirelength reduction. Designing a 3D-IC introduces an extra parameter to be considered while assigning a layer to any circuit component where different layers are connected by Through Silicon Vias. In this chapter, we have applied the Parallel-PSO approach to optimize the area, wirelength of the layout and the number of TSVs to connect the different layers simultaneously. The results are obtained and compared with the benchmark circuits available with MCNC and GSRC.

1. **Introduction**

A three-dimensional IC is an Integrated circuit manufactured by laying several vertical silicon wafers or die. The different layers are connected either by Through Silicon Vias (TSVs) or by Cu-Cu interconnects [89, 90, 91, 92, 93]. While the Cu-Cu interconnection with silicon generates sheer force between them, there is a high chance of failure when IC is heated and hence TSVs are widely used to connect these different layers. Recently 3-D integration has attracted researchers as it provides a higher device density as well as higher bandwidth. It is also possible to integrate heterogeneous technologies in a layered die stack structure, which counteracts system-on-chip integration. Other benefits of 3-D ICs include smaller footprints; lower interconnect delays, higher performance, as well as lower power consumption. 3-D IC changes the wirelength distribution from 2-D layout. Nets can be made shorter in 3-D layout, but TSVs are not free and therefore cannot be used at random. There are two approaches used to design 3-D ICs, via-first and via-last. The TSVs in the Via-first approach only interfere with the device layers [Fig: 6.1(a)], while in the Via-Last approach: the TSVs interact not only with the device layers but also with the metal layers [Fig: 6.1(b)].

(a)

(b)

**Figure 6.1:** (a) Via-First TSVs & (b) Via-Last TSVs [61]

The inputs to 3D-ICs are:

1. A set of blocks with a specific shape and size,
2. A list of number of terminals on each block and,
3. The netlist describing the interconnections between these blocks.
4. **Problem Formulation**

We have focused our work in two-layer 3D-IC, where we have partitioned the set of blocks in two different layers such that their area difference is as small as possible.

Let be a set of ‘n’ modules.Themodulecould be represented by*, where* is itswidth and is its height.

Let be the set nets, where 'm' denotes the total number of nets that link the blocks,

Takingas the estimated length of net ,, The placement objective is to identify a group of rectangles represented by for each block represented by set such that,

1. Each block can be placed either in the two layer
2. Each block can be put in the rectangle, which has the dimensions.
3. No two modules overlap each other, that i.
4. The total area occupied by is minimized
5. The total wirelength determined by is minimized.
6. The total number of TSVs should be minimized.
7. **TSVs Optimization**

The 3D-IC design problem involves division of the circuit netlist into multiple parts (in our case two or four parts) such that there are some connections between these parts. The number of edges in the two parts of the circuit is the number of TSVs in the 3D-IC and this number can be calculated as follows:

(6.1)

Where,are the edge's vertices.

The design problem in first stage is a partitioning problem where the netlist, say is to partition into, such that,

(6.2)

And,

(6.3)

As the problem involves bipartitioning of a circuit, so equality condition must be satisfied as

(6.4)

1. **Wirelength Estimation**

In 3-D floorplanning, there is a high chance that all the terminals of a net may lie in multiple layers and hence the lateral wirelength calculation becomes necessary. Most of the works related to the calculation of lateral wirelength suggested using Half Perimeter Wirelength (HPWL), Wirelength of a net is determined by measuring half perimeter of the bounding box of all its terminals, assuming they are all in the same plane, as shown in figure 6.2 (a) [94].

n1

n2

n3

(a)

(b)

(c)

**Figure 6.2:** wirelength estimation models (a) Bounding box of all terminals of a net, (b) Bounding box of all terminals of a net and TSVs associated with them and (c) a net divided into subnets and summing up individual subnet wirelength

The drawback of this method is that it estimates the lateral wirelength without any information of TSVs locations. Although this is unavoidable as the floorplan do not take care of the TSV placements. In the other technique to estimate the wirelength, the bounding box is chosen such that it covers all the terminals of the net as well as the TSVs associated with that net as illustrated in figure 6.2(b). However,it underestimates the total wirelength when a net has terminals in multiple dies [94].

To overcome the drawbacks of these two methods, in our technique wehave estimated the total lateral wirelength by first calculating wirelength on each individual dies then summing up them.Alsoin ourproposed technique, the wirelength of a net (say n) on a particular die is calculated by the half perimeter wirelength of all its terminals on that die and any TSV of n in the same die or die above/below it. In figure 6.2(c), the lateral wirelength of a 3D net n is obtained by summing up the estimated wirelength of subsets n1, n2 & n3 [74].

Here we have bipartitioned the netlist hence the total wirelength will be calculated as the sum of wirelength of partition 0 (), wirelength of partition 1 () and the wirelength due to TSVs (),

Hence total wirelength (6.5)

To estimate the wirelength due to TSVs we have taken the TSV size as 3 *μ*m as in [90-93] unless otherwise specified.

Hence,

(6.6)

Where, T is the total number of TSVs

The TSV size is 3 *μ*m as in [93] unless otherwise specified.

1. **Combined Area, Wirelength and TSVs optimization**

While the purpose of traditional 3D-IC design methods is to reduce the number of TSVs, integrating it with the area and wirelength reduction makes this work much harder. Designing multi-objective 3D-ICs is substantially more complicated.One of the most challenging aspects of multi-objective optimization is that there is no one best solution for every target in the solution space. Additionally, adopting an ideal solution for one goal may necessitate receiving a suboptimal result for another. Hence, it is difficult to define what constitutes a good answer. The formulation of a good solution is adopted as it appears in [58]:

1. Allow for more precise handling of the tradeoffs between goals.
2. Produce partitioning that is predictable.
3. Provide a method for dealing with objectives that correspond to amounts of various types.

In this case, a designtechnique is requiredthat can optimize objective, the number of TSVs, minimization of objective the total area occupied by the modules in two/four-layer and minimizing objective the total wirelength. However, the three objectives are dissimilar objectives, which means that optimizing alone does not necessarily imply that and are optimized and vice-versa. That is why we adopt a combination-based formulation for multi-objective optimization: The combined objective will be a scalar combined metric given by the following equation:

(6.7) Where,

,

,

Minimizing equation (6.1) seeks to calculate a 3D layout that is as close as possible to any of the best in terms of any beginning goal. The area weight may be used to traverse the distance between each objective's best solution locations, resulting in a predictable structure based on the area weight as well as fine-tuned management of the tradeoff between the three objectives.

1. **The 3D-IC design technique**

In the present work, a P-PSO algorithm (as proposed in Chapter 4) for the optimization of multimodal continuous functions is proposed. P-PSO is used for global optimization by updating particle locations to achieve quick convergence. To determine the layout in each of the two layers the sequence pair (SP) technique with LCS as described in chapter 4 has been used. Firstly, the given netlist has been bi-partitioned and then four different types of operations are allowed to perturb the bi-partitioned circuit and in the given sequence pair to another sequence pair listed as:

Op1: Swap two module names in all the partitions.

Op2: Swap two module names in only one sequence of each partition.

Op2: Swap two module names in both sequences of each partition.

Op3: Rotate a module in each partition.

To start with the designing of 3D-IC we first converted the information provided by the netlist into a matrix known as *adjacency matrix*, where column and row represents the nodes. Then we ha randomly bipartition the given netlist by calling the *initial\_position* function. In the next step the total number of interconnections (TSVs) between different layers of 3D-IC is calculated. Total area (A) occupied along with the amount of wirelength required to connect different nodes are calculated. The combined objective function as given in equation 6.5 in initiated. The proposed parallel Particle Swarm Optimization is then applied to achieve the minimum of all the three parameters viz. Area, Number of TSVs and the Wirelength requirement. The steps for the proposed approach for the 3D design problem under consideration are presented as under:

**Algorithm for 3D IC design using P-PSO**

1. Start at the beginning of netlist and convert it into matrix form.
2. Bipartition the circuit into 0 and 1 partitions as

(6.8)

Also, from (6.4),

1. Calculate their TSV using

(6.9)

1. Determine the position of modules in each partition and determine the corresponding area and wirelength using Sequence pair (SP) technique with the help of LCS. The total area and wirelength are as:

(6.10)

(6.11)

Where,

and

1. Initialize the two different sets of PSO parameters parallelly with the same number of particles corresponding to each node of the sequence pair
2. Correspondingly evaluate fitness function, , for all the particles using (6.5), taking weight , (for 50% weight to Area and Wirelength objectives).
3. Randomly initialize position vector of each pair of particles
4. Generate initial velocity vector for each pair of particles
5. evaluate the fitness value of each pair of particles using the objective function (using equation 6.5)
6. set t and in the swarm for both pair of particles
7. update the inertia weight
8. update
9. update the velocity vector
10. update the position vector

Op1: Swap two module namescorresponding to in both the partitions and calculate the fitness function as.

Op1: Swap the positions of the modules corresponding to in and calculate the fitness function as.

Op2: Swap the positions of the modules corresponding to in both the sequence and calculate the fitness function as.

Op3**:** Rotate the modules corresponding to and calculate the fitness function as

Chose the best fitness among

1. is better than the

Update

1. **if**  is better than the

Update

Generate initial velocity vector for each pair of particles

Evaluate the fitness function Cc, and store TSV, Area and Wirelength values in an array

Set t and in the swarm for both pair of particles

Enter the netlist file and read its adjacency matrix

Call ‘initial-position’ function to bi-partition the netlist, where initial\_position function takes the netlist as input and return initial partitions

Calculate TSVs (T), Area (A) of each partition and total wirelength (W) of initial partitions using function files ‘TSV’ , ‘AREA’ and ‘WIRELENGTH’

Randomly generate sequence pair for both the partitions

Calculate combined fitness function

Taking

Randomly generate two different sets of particles from the initial partitions using the concept of nodes swapping

No

Yes

Set maximum number of iteration as

Is

Update the inertia weight as

Update

Apply particle velocity & particle position equations for all the particles generated randomly earlier.

Op1: Swap two module names corresponding to in both the partitions and calculate the fitness function as.

Op2: Swap the positions of the modules corresponding to in and calculate the fitness function as.

Op3: Swap the positions of the modules corresponding to in both the sequence and calculate the fitness function as.

Op4**:** Rotate the modules corresponding to and calculate the fitness function as

Chose the best fitness among

Iterate back the positions corresponding to better fitness values to input of P-PSO.

Print all the fitness values, TSV, Areaand Wirelength values after applying P-PSO

Repeat same procedure for all the particles

Retain previous values in array

Update&values of particles during each iteration

NO

YES

Is fitness function (after P-PSO) > fitness function (before P-PSO)?

**Figure 6.3:** Algorithm for minimizing TSV, Area and Wirelength in two-layered 3D-IC using P-PSO

1. **Experimental Results**

We simulated the proposed 3-D floorplans with are, wirelength &TSV co-optimization. We used the MCNC &GSRChard benchmark suites as our test cases. We have the unit in the MCNC &GSRC benchmarks to 1 *μ*m. The TSV sizeis 3 *μ*m as in [93] unless otherwise specified. The IO padlocations are assigned randomly.First, we tested our approach for two-layered 3D-IC then for a fair comparison the four-layered results were obtained and compared with [10, 93 and 94]; the comparison results are shown in table 6.1. The Tabu Search and Simulated Annealing techniques were used and designed to optimize the area and TSVs by [10].In [93] authors applied the shuffling frog leaping method to tackle the optimization issue of 3D ICdesign in terms of area and TSVs count.SA [94] was used by to optimize the wirelength and TSVs. They presented their technique in two stages, stage one planned the hard macros and TSV-blocks at the same time. The wirelength is improved in stage two by reassigning signal TSVs. Further in Table 6.2 and Table 6.3 we have presented the optimization results of Area, Wirelength and TSVs simultaneously for two layered 3D IC design for MCNC and GSRC Benchmarks respectively.

**Table 6.1:** 4-layered 3D IC Parameters optimization comparison of results on MCNC & GSRC Benchmark Circuits

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Benchmark** | **Algorithm** | **Tabu Search [10]** | **SFLP [93]** | **SA [94]** | **ours** | **% improve-ment over [10]** | **% improve-ment over [94]** |
| ami33 | WL) | ---- | ---- | 45179 | 28750 | ---- | 36.36 |
| Avg. Area | 573000 | 586000 | ---- | 339136 | 40.81 | ---- |
| TSV | 116 | 108 | 141 | 159 | -37.07 | -12.76 |
| ami49 | WL) | ---- | ---- | 585804 | 365666 |  | 37.58 |
| Avg. Area | 7481931 | 7481895 | ---- | 9167900 | -22.53 | ---- |
| TSV | 292 | 263 | 436 | 171 | 41.44 | 60.78 |
| n30 | WL) | ---- | ---- | ---- | 17906 | ---- | ---- |
| Avg. Area | ---- | ---- | ---- | 55025 | ---- | ---- |
| TSV | ---- | ---- | ---- | 310 | ---- | ---- |
| n50 | WL) | ---- | ---- | ---- | 24628 | ---- | ---- |
| Avg. Area | ---- | ---- | ---- | 51622 | ---- | ---- |
| TSV | ---- | ---- | ---- | 446 | ---- | ---- |
| n100 | WL) | ---- | ---- | 148748 | 53514 | ---- | 64.02 |
| Avg. Area | 48170 | 48112 | --- | 45087 | 6.40 | ---- |
| TSV | 996 | 804 | 1171 | 742 | 25.50 | 36.64 |
| n200 | WL) | ---- | ---- | 291091 | 99728 | ---- | 65.74 |
| Avg. Area | 50646 | 51097 | ---- | 45971 | 9.23 | ---- |
| TSV | 2035 | 1468 | 2179 | 1542 | 24.23 | 29.23 |
| n300 | WL) | ---- | ---- | 391694 | 164364 | ---- | 58.04 |
| Avg. Area | 76223 | 76478 | ---- | 71751 | 5.87 | ---- |
| TSV | 2133 | 1823 | 2730 | 1793 | 15.94 | 34.32 |

**Figure 6.4:** Comparison of No. of TSVs with other algorithms

**Figure 6.5:** Comparison of wirelength with other algorithms

**Table 6.3:**2-layered 3D IC Parameters optimization results for GSRC Benchmark Circuits

**Table: 6.2:** 2-layered 3D IC Parameters optimization results for MCNC Benchmark Circuits

|  |  |  |
| --- | --- | --- |
| **Benchmark** | **Parameters** | **Result** |
| Apte | Wirelength) | 20668 |
| Avg. Area | 53191804 |
| TSV | 9 |
| Xerox | Wirelength) | 205399 |
| Avg. Area | 19701250 |
| TSV | 18 |
| Hp | Wirelength) | 84308 |
| Avg. Area | 8930936 |
| TSV | 6 |
| ami33 | Wirelength) | 64992 |
| Avg. Area | 1164240 |
| TSV | 42 |
| ami49 | Wirelength) | 36812916 |
| Avg. Area | 768986 |
| TSV | 98 |

|  |  |  |
| --- | --- | --- |
| **Benchmark** | **Parameters** | **Result** |
| n10 | Wirelength) | 15528 |
| Avg. Area | 230468 |
| TSV | 12 |
| n30 | Wirelength) | 54914 |
| Avg. Area | 221028 |
| TSV | 57 |
| n50 | Wirelength) | 90182 |
| Avg. Area | 207208 |
| TSV | 127 |
| n100 | Wirelength) | 156698 |
| Avg. Area | 176599 |
| TSV | 251 |
| n200 | Wirelength) | 270722 |
| Avg. Area | 185008 |
| TSV | 485 |
| n300 | Wirelength) | 429918 |
| Avg. Area | 292008 |
| TSV | 639 |

**Table 6.4:**Runtime (in seconds) comparison for 4-Layered 3D IC

|  |  |  |  |
| --- | --- | --- | --- |
| **Benchmark** | **Runtime**  **(in seconds)**  **SA [94]** | **Runtime**  **(in seconds)**  **(Proposed-PSO)** | **%**  **Improvement** |
| ami33 | 42.46 | 63.15 | -48.73 |
| ami49 | 184.63 | 126.33 | 31.58 |
| n30 | ---- | 61.52 | ---- |
| n50 | ---- | 114.35 | ---- |
| n100 | 1306.39 | 306.65 | 76.53 |
| n200 | 8237.10 | 1422.19 | 82.73 |
| n300 | 21450.50 | 2793.42 | 86.98 |

**Table 6.5:** Runtime for 2-Layered 3D IC (in seconds)

|  |  |
| --- | --- |
| **Benchmark** | **Runtime**  **(in seconds)**  **(Proposed-PSO)** |
| apte | 7.58 |
| xerox | 8.59 |
| hp | 6.95 |
| ami33 | 21.62 |
| ami49 | 39.23 |
| n10 | 4.29 |
| n30 | 20.57 |
| n50 | 36.54 |
| n100 | 79.65 |
| n200 | 396.12 |
| n300 | 598.95 |

1. **Summary**

3D integrated circuits (3D-ICs) are a new technology that has a lot of promise. 3D-ICs have a tiny footprint and vertical linkages between dies, allowing for shorter wirelength between gates. As a result, they have lower connection latency and power consumption. The 3D Partitioning and Layer Assignment stage is the first of several in the design flow of 3D integrated circuits. This step is crucial since the outcome will have an impact on the performance of succeeding processes. This issue is NP-hard, much like other partitioning problems. The implementation of iterative heuristics [22] was used to handle this essential problem. When attempting to tackle this problem, several factors have been considered. Layer assignment, TSV reduction, wirelength optimization and area balance are some of these considerations. To do this objective, we have proposed Parallel PSO (P-PSO). The result obtained were compared and found to be giving better solutions when compared to other algorithms. As shown in table 6.1 as compared to SA [95] the average wirelength has an average improvement of 52.35% and the TSV count has an average improvement of 29.64%. Our results show an average improvement of 8.36% over the area occupied by the three-layered 3D IC and an average improvement of 14% over TSV count as compared to the results with Tabu Search [10].

**CHAPTER 7**

**Chapter 7**

**Conclusion & Future Scope**

1. **Conclusion**

In the present dissertation, we have done the physical domain design for VLSI circuits. The work presented in this thesis has considered the design aspects of partitioning, floorplanning and placement for 2D and 3D ICs. We have proposed several algorithms to optimize the design problem in various steps of the practical domain. Through our computation / simulation, it is shown that the various parameters can be simultaneously optimized with our proposed algorithms.

In chapter 3 of this thesis, VLSI circuit bi-partitioning using PSOAlgorithm has been proposed for mincut and circuit delayminimization along with maximization of sleep time. Theadvantages of the proposed PSO approach are:

1. It is fast, thus applicable to large-sized circuits.
2. It performs better suitable partitioning, as it optimizes all the parameters with some cutsize, delay and sleep time tradeoff.

The Particle Swarm Optimization algorithm applied to VLSI partitioning produces a very good result of these three objectives simultaneously. This triple goal function was first defined independently and subsequently integrated into one. Because the combined issue is NP-hard, a heuristic technique has been successfully implemented.

In chapter 4 ofthe present thesis, we have proposed Parallel-PSO, a modified version of the classical PSO algorithm to minimize the total area occupancy of a circuit on the wafer. The drawbacks of PSO are

1. The variations in the parameters of the problem may significantly result in a large performance shift using PSO [17].
2. In the PSO the global best (*gbest*) guides the rest of the swarm which may result in the creation of similar particles with loss of diversity. This phenomenon of the PSO increases the probability of being trapped in the local optimum solution [18]. PSO often converges to a premature solution especially when the search space is of high dimension [19] or the problem to be optimized requires parallel computations of the target solution(s).
3. The basic PSO includes the process that can be applied only to a search space with a fixed dimension and at a time, it chooses only one swarm. However, in many optimization problems, the optimum dimension is unknown; also, it may require multiple swarms to effectively solve the problem.

These problems are effectively addressed by the parallel PSO (P-PSO) technique, which can avoid the premature convergence problem. Furthermore, Parallel PSO avoids the need for fixing the dimension of the solution space in advance. The resultpresented on area optimization using P-PSO shows that proposed algorithm can be successfully implement in the Sequence-Pair based floorplanning where multiple perturbation operation is needed to be performed in every step.

In chapter 5, the PSACO, a hybridized PSO and ACO algorithm is presented to optimize area occupancy and total wirelength consumption of a circuit. The algorithm was tested on both MCNC and GSRC benchmark circuits.As the floorplanning/placement problem is NP-Hard, it can solved by using heuristic approaches. The inclusion of ACO in PSO provides the benefits such as:

1. PSO is significantly quicker than other evolutionary algorithms in finding decent quality solutions and,
2. If the swarm maintains equilibrium, the development process will become stagnant over time. As a result, as the number of generations increases, so does the inability to enhance the quality of the answers.

Chapter 6, of the present work, provides the 3D-IC design technique to optimize the three design parameters; Area occupancy, wirelength requirements and TSVs requirements simultaneouslyWe have presented our results for two and four-layered 3D ICs. The benchmarks MCNC & GSRC have been used to show the results. The obtained results on MCNC & GSRC benchmark suggest that the proposed P-PSO can be applied to design 3D-IC where we need to optimize the multiple parameters simultaneously.

1. **Future Scope**

The current dissertation addresses the design issues of partitioning, floorplanning, placement and 3D IC design. The proposed algorithms have been tested over various benchmark circuits. Future research of the current work proposed in this thesis can be categorized as below:

1. The useddelay optimizedforthe partitioning in our course of work is the net-based delay. The same approach can be used to optimize path-based delay. After finding path delay, combined net and path-based delay can be calculated by giving weights to each delay. Then, the proposed PSO approach can be applied for mincut and combined delayminimization with sleep time maximization. The results can be improved by combining PSO with other evolutionary algorithms such as GA, ACO and many more.
2. We have strict our work for Hard modules only, the soft modules with allowable aspect ratios can further be used to optimize the area and wirelength constraints,
3. In our dissertation, we have used only rectangular modules, U-shaped or L-shaped modules may also be used to investigate the proposed algorithms.
4. The chip routability and temperature are of the major concerns while designing 3D ICs, it may be combined with TSV counts, total Area occupancy and, wirelength utilization to design the 3D IC in such a way that the peak temperature may be reduced to improve the chip performance.

**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.M. Fidduccia and R.M. Mattheyes, “A linear time heuristic or improving network partitions,” In Proc. *of the 19th ACM/IEEE Design Automation Conference*, IEEE Press, Piscataway, NJ, USA, pp.175-182, June 1982.
12. Sung Kyu Lim, “Practical Problems in VLSI Physical Design Automation,”Springer Science & Business Media, Jul-2008.
13. Krishnamurthy, “An Improved Min-Cut Algonthm for Partitioning VLSI Networks,” *IEEE Transactions on Computers,* vol. C-33, no. 5, pp. 438-446, May 1984.
14. L. A. Glasser and L. P. J. Hoyte, “Delay and Power Optimization in VLSI Circuits,” in Proc.21st Design Automation Conference Proceedings,Albuquerque, NM, USA, pp. 529-535, June 1984.
15. L. A. Sanchis, “Multiple-way network partitioning,” in IEEE Transactions on Computers, vol. 38, no. 1, pp. 62-81, 1989.
16. G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, “Multilevel hypergraph partitioning: applications in VLSI domain,”*in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 69-79, 1999.
17. 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.
18. 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.
19. 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.
20. 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.
21. S. M. Sait, A. H. El-Maleh and R. H. Al-Abaji, “Enhancing performance of iterative heuristics for VLSI netlist partitioning,” In proc. of10th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2003), Sharjah, United Arab Emirates, pp. 507-510, Dec. 2003.
22. M. E. Ekpanyapong and S. K. Lim, “Simultaneous delay and power optimization for multi-level partitioning and floorplanning with retiming,” in Proc. *the Int. Symp. on Circuits and Systems (ISCAS 2004)*, Vancouver, Canada, pp.1-9, May 2003.
23. 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. in proc. Fifth International Workshop on System-on-Chip for Real-Time Applications (IWSOC'05), Banff, AB, Canada, pp. 368-371., July 2005.
24. S. Peng, G. Chen and W. Guo, “A Discrete PSO for Partitioning in VLSI Circuit,” in proc. International Conference on Computational Intelligence and Software Engineering,Wuhan, China, pp. 1-4, Dec. 2009.
25. S. S. Gill, R. Chandel and A. Chandel, “Swarm Intelligence based circuit partitioning,” in proc. Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia),Shanghai, China, pp. 41-44, Sept. 2010.
26. I. H. Shanavas, R. K. Gnanamurthy and T. S. Thangaraj, “A Novel Approach to Find the Best Fit for VLSI Partitioning - Physical Design,”in proc. International Conference on Advances in Recent Technologies in Communication and Computing, Kottayam, India, pp. 330-332, Dec. 2010.
27. J. J. Cong and Kwok-Shing Leung, “Optimal wiresizing under Elmore delay model,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 14, no. 3, pp. 321-336
28. S. N. Adya and I. L. Markov, “Fixed-outline floorplanning: enabling hierarchical design,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 6, pp. 1120-1135, 2003.
29. Hayward H. Chan and Igor L. Markov, “Practical slicing and non-slicing block-packing without simulated annealing,” in Proc. *of the 14th ACM Great Lakes symposium on VLSI (GLSVLSI '04)*, New York, United States pp. 282–287, April 2004.
30. Tung-Chieh Chen and Yao-Wen Chang, “Modern floorplanning based on B/sup \*/-tree and fast simulated annealing,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 4, pp. 637-650, 2006.
31. P. G. Sassone and Sung Kyu Lim, “Traffic: a novel geometric algorithm for fast wire-optimized floorplanning,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 6, pp. 1075-1086, 2006.
32. J. Lin and Z. Hung, “SKB-Tree: A Fixed-Outline Driven Representation for Modern Floorplanning Problems,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 3, pp. 473-484, 2001.
33. H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, “VLSI module placement based on rectangle-packing by the sequence-pair,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, no. 12, pp. 1518-1524, 1996.
34. C. Long, L. J. Simonson, W. Liao and L. He, “Microarchitecture Configurations and Floorplanning Co-Optimization,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 15, no. 7, pp. 830-841, 2007.
35. T. Chen, Y. Chang and S. Lin, “A New Multilevel Framework for Large-Scale Interconnect-Driven Floorplanning,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 2, pp. 286-294, 2008.
36. Li Li, Yuchun Ma, Ning Xu, Yu Wang and Xianlong Hong, “Floorplan and Power/Ground network co-design using guided incremental floorplanning,” in proc. IEEE 8th International Conference on ASIC,Changsha, China, pp. 747-750, Oct. 2009.
37. P. Ghosh and A. Sen, “Power efficient voltage islanding for Systems-on-chip from a floorplanning perspective,” in proc. Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, Germany, pp. 654-657, Mar. 2010.
38. N. Liu, S. Chen and T. Yoshimura, “Floorplanning for high utilization of heterogeneous FPGAs,” in proc. 12th International Symposium on Quality Electronic Design, Santa Clara, CA, USA, pp. 1-6, Mar. 2011.
39. A. B. Chong, “Gross Die Estimator's caveats for ASIC floorplanning,” in proc. 10th IEEE International Conference on Semiconductor Electronics (ICSE),Kuala Lumpur, Malaysia, pp. 431-435, Sept. 2012.
40. C. Hoo, K. Jeevan, V. Ganapathy and H. Ramiah, “PPTP: Pre-Post Terminal Propagation in modern fixed-outline soft module VLSI floorplanning design,” in proc. 10th IEEE International Conference on Semiconductor Electronics (ICSE), Kuala Lumpur, Malaysia, pp. 448-452, Sept. 2012.
41. Iksoon Lim, Donghoon Yeo, Wang Yu, Hyunchul Shin and Hyounseok Song, “Optimal flip-chip floorplanning with area IO,” in proc. 7th International Conference on Computing and Convergence Technology (ICCCT), Seoul, pp. 1335-1338, Dec. 2012.
42. R. K. Nain and M. Chrzanowska-Jeske, “Fast Placement-Aware 3-D Floorplanning Using Vertical Constraints on Sequence Pairs,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 9, pp. 1667-1680, 2011.
43. Z. Li et al., “Efficient Thermal via Planning Approach and Its Application in 3-D Floorplanning,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no. 4, pp. 645-658, 2007.
44. H. Hong, J. Lim and S. Kang, “Process variation-aware floorplanning for 3D many-core processors,” in proc. IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS), pp. 193-196, Dec. 2012.
45. J. Z. Yan and C. Chu, “SDS: An Optimal Slack-Driven Block Shaping Algorithm for Fixed-Outline Floorplanning,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 2, pp. 175-188, 2013.
46. D. Gracia Nirmala Rani andS. Rajaram, “A novel differential evolution based optimization algorithm for non-sliceable VLSI floorplanning” *Iranian Journal of Science & Technology*, vol. 39, pp. 375–382, 2015
47. M. Rabozzi, G. C. Durelli, A. Miele, J. Lillis and M. D. Santambrogio, “Floorplanning Automation for Partial-Reconfigurable FPGAs via Feasible Placements Generation,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 1, pp. 151-164, 2017.
48. Zhigang He, Yuchun Ma, Ning Xu, Yu Wang and Xianlong Hong, “On handling fixed blocks in incremental fixed-outline floorplanning,” in proc. International Conference on Communications, Circuits and Systems (ICCCAS), Chengdu, China, pp. 876-880, July 2010.
49. Xiaoping Tang, Ruiqi Tian and D. F. Wong, “Fast evaluation of sequence pair in block placement by longest common subsequence computation,”IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 12, pp. 1406-1413, 2001.
50. Yingxin Pang, Chung-Kuan Cheng and Takeshi Yoshimara, “An enhanced perturbing algorithm for floorplan design using the O-tree representation,” in proc. *of the 2000 international symposium on Physical Design (ISPD’00),*San Diego California USA, pp. 168-173, May 2000.
51. G.Chen, W. Guo&Y. Chen,“A PSO-based intelligent decision algorithm for VLSI floorplanning,” *Soft Comput* vol. 14, pp. 1329–1337, 2010.
52. J. Chen, W. Zhu and M. M. Ali, “A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning,” in IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 4, pp. 544-553, July 2011.
53. S. R. Choudhury and S. N. Pradhan, “DOTFloor–A Diffusion Oriented Time-Improved Floorplanner for Macrocells,”*in*IEEE Access, vol. 7, pp. 172074-172087, 2019.
54. S. Pinge, R. K. Nain and M. Chrzanowska-Jeske, “Fast floorplanning with placement constraints,” in proc. IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS),Cusco, Peru, pp. 1-4, Mar. 2013.
55. Q. Xu, S. Chen and B. Li, “Ant system based 3D fixed-outline floor planning,” in proc. 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Guilin, China, pp. 1-3, Oct. 2014.
56. P. Zarkesh-Ha, J. A. Davis and J. D. Meindl, “Prediction of net-length distribution for global interconnects in a heterogeneous system-on-a-chip,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 8, no. 6, pp. 649-659, 2000.
57. Kirk Schloegel, George Karypis and Vipin Kumar, “A New Algorithm for Multi-objective Graph Partitioning,”in Proc. *5th International Euro-Par Conference on Parallel Processing (Euro-Par '99),*Toulouse, France, pp. 322–331, Aug. 1999.
58. Sivaranjani, P., Senthil Kumar, “A. Thermal-Aware Non-slicing VLSI Floorplanning Using a Smart Decision-Making PSO-GA Based Hybrid Algorithm,”  *Circuits Syst Signal Process*vol. 34, pp. 3521–3542, 2015.
59. Priyajit Mukherjee, Sandeep D'souza, Santanu Chattopadhyay, “Area Constrained Performance Optimized ASNoC Synthesis with Thermal‐aware White Space Allocation and Redistribution,” *Integration*, vol. 60, Pages 167-189, 2018.
60. Amarbir Singh and Leela Jain, “VLSI Floorplanning Using Entropy Based Intelligent Genetic Algorithm,” in proc. *International conference onComputing, Analytics and Networks.*, Rajpura, India , pp. 53-71, July 2017.
61. S. Banerjee, A. Ratna and S. Roy, “Satisfiability modulo theory based methodology for floorplanning in VLSI circuits,” in proc. Sixth International Symposium on Embedded Computing and System Design (ISED), Patna, India, pp. 91-95, Dec. 2016.
62. S. Chen, J. Huang, X. Xu, B. Ding and Q. Xu, “Integrated Optimization of Partitioning, Scheduling and Floorplanning for Partially Dynamically Reconfigurable Systems,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 1, pp. 199-212, 2020,
63. S. -E. D. Lin and D. H. Kim, “Construction of All Rectilinear Steiner Minimum Trees on the Hanan Grid and Its Applications to VLSI Design,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 6, pp. 1165-1176, June 2020.
64. Z. Huang, Z. Lin, Z. Zhu and J. Chen, “An Improved Simulated Annealing Algorithm With Excessive Length Penalty for Fixed-Outline Floorplanning,” inIEEE Access, vol. 8, pp. 50911-50920, 2020.
65. A. Sadeghi, M. Zolfy Lighvan and P. Prinetto, “Automatic and Simultaneous Floorplanning and Placement in Field-Programmable Gate Arrays With Dynamic Partial Reconfiguration Based on Genetic Algorithm,” in Canadian Journal of Electrical and Computer Engineering, vol. 43, no. 4, pp. 224-234, 2020.
66. M. Rabozzi, G. C. Durelli, A. Miele, J. Lillis and M. D. Santambrogio, “Floorplanning Automation for Partial-Reconfigurable FPGAs via Feasible Placements Generation,”*in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 151-164, 2017.
67. J. Kennedy and R. Eberhart, “Particle swarm optimization,” in proc. ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, pp. 1942-1948, Dec. 1995.
68. T. Bäck and H. Schwefel, “An Overview of Evolutionary Algorithms for Parameter Optimization,” in Evolutionary Computation, vol. 1, no. 1, pp. 1-23, 1993.
69. David E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning,” (1st. ed.). *Addison-Wesley Longman Publishing Co*., Inc., USA, 1989.
70. J. Koza, “Genetic programming: On the programming of computers by means of natural selection” *Cambridge, MA: MIT Press*, 1992.
71. D. Delahaye, S. Chaimatanan and M. Mongeau, “Simulated Annealing: From Basics to Applications,”*Handbook of Metaheuristics. International Series in Operations Research & Management Science,* vol 272. Springer, Cham, 2018.
72. [Thomas Back](https://www.worldscientific.com/doi/abs/10.1142/9789812830753_0001)and [Frank Kursawe](https://www.worldscientific.com/doi/abs/10.1142/9789812830753_0001), “Evolutionary algorithms for fuzzy logic: A brief overview*”, fuzzy logic and soft computing Singapore: World Scientific*, Singapore, pp. 3-10, 1995.
73. J. Riget and J. S. Vesterstrom, “A diversity-guided particle swarm optimizer—The ARPSO” *Technical Report, Department of Computer Science*, University of Aarhus, Aarhus, Denmark, 2002
74. C. Li, W. Mak and T. Wang, “Fast Fixed-Outline 3-D IC Floorplanning With TSV Co-Placement,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 3, pp. 523-532, 2013.
75. Andries P. Engelbrecht, “Computational Intelligence: An Introduction”, *2nd Edition, Wiley*, 2007.
76. M. M. Millonas, “Swarms, Phase Transitions and collective Intelligence”, *arXiv: Adaptation and Self-Organizing Systems,*pp. 1-33, *1993.*
77. M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,”IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53-66, 1997.
78. [V.Selvi](https://www.researchgate.net/scientific-contributions/VSelvi-50820488?_sg%5B0%5D=hJoAwDjcU7zddmilIcVxNLKUtNqAmWVR30X0qrWkEML4qMGvQagyRZbXfZkME0zhumC1CYk.urLqcex7B890DzLZvN9StfzpcL3isdKtAiaeQhR1D48UC39347IXfW92_epmw91fXBrDccczcN3SVt-pyk10Nw.RnoyxSUw5-sjnWXvblXIHO1hbuu5SZPaSQZj0gKuSwkY8A_HAndE03npe7hQY8ghhH0ZWP7p-9WF_53kCMvJlg&_sg%5B1%5D=RWAxdv0_ulFg7n9rO1YEx_MQGNpEjm8c_FebGCaTnxSrS8Lq0XKKziKbf4MvZ0Jk0aV1yto.FsjahTF8u9mFO_J63Z1YX7y9Ud6hrJTDqadyD9ZKXURGePMk3ZM_cHmNsHevm_oZKuKoh2bDA0zZZrjOVUF9wA) and [Dr. R. Umarani](https://www.researchgate.net/scientific-contributions/DrRUmarani-43960524?_sg%5B0%5D=hJoAwDjcU7zddmilIcVxNLKUtNqAmWVR30X0qrWkEML4qMGvQagyRZbXfZkME0zhumC1CYk.urLqcex7B890DzLZvN9StfzpcL3isdKtAiaeQhR1D48UC39347IXfW92_epmw91fXBrDccczcN3SVt-pyk10Nw.RnoyxSUw5-sjnWXvblXIHO1hbuu5SZPaSQZj0gKuSwkY8A_HAndE03npe7hQY8ghhH0ZWP7p-9WF_53kCMvJlg&_sg%5B1%5D=RWAxdv0_ulFg7n9rO1YEx_MQGNpEjm8c_FebGCaTnxSrS8Lq0XKKziKbf4MvZ0Jk0aV1yto.FsjahTF8u9mFO_J63Z1YX7y9Ud6hrJTDqadyD9ZKXURGePMk3ZM_cHmNsHevm_oZKuKoh2bDA0zZZrjOVUF9wA), “comparative analysis of ant colony and particle swarm optimization techniques” *International Journal of computer Application*, volume 5 no. 4 , pp.1-6,2010.
79. P.S Shelokar, V.K Jayaraman and B.D Kulkarni, “An ant colony approach for clustering”, *Analytica Chimica Acta*, vol. 509, Issue 2, Pages 187-195, 2004.
80. P. Angeline, “Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences”, in proc. *Int. Conf. on Evolutionary Programming VII*, San Diego, California, pp. 601-610, Mar. 1998.
81. P. N.Guo, C. K. Cheng and T. Yoshimura, “An O-tree representation of non-slicing floorplan and its applications,” in proc.*36th ACM/IEEE conference on design automation conference,* New Orleans, Louisiana, USA, pp. 268–273, June 1999.
82. Lin J. M, Chang Y. W., “TCG: a transitive closure graph-based representation for non-slicing floorplans” in proc. *38th conference on design automation, ACM*, Las Vegas, Nevada, pp 764–769, June 2001.
83. R. B. Singh and A. S, Baghel, “IC Floorplanning Optimization using Simulated Annealing with Order-based Representation” *I. J. Intelligent Systems and Applications*, 2, pp. 62-70, 2021.
84. Lalin L Laudis, N.Ramadass, Shilpa Shyam, R.Benschwartz and V.Suresh, “An Adaptive Symbiosis based Metaheuristics for Combinatorial Optimization in VLSI”, *Procedia Computer Science*, Elsevier, vol. 167,pp. 205-212, 2020.
85. Q. Ma and E. F. Y. Young, “Multivoltage Floorplan Design,”IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 4, pp. 607-617, 2010.
86. S. Karimullah, D. V. Vardhan and S. J. Basha, “Floorplanning for Placement of Modules in VLSI Physical Design Using Harmony Search Technique,” in *Lecture Notes in Electrical Engineering*, Springer, Singapore, pp. 1929-1936, 2020.
87. M. Shunmugathammal, C. C. Columbus and S. Anand, “A nature inspired optimization algorithm for VLSI fixed-outline floorplanning,” in *Analog Integrated Circuit and Signal Processing*, vol. 103, pp. 173-186, 2020.
88. M. Shunmugathammal, C. C. Columbus and S. Anand, “A Novel B\*tree Crossover‑Based Simulated Annealing Algorithm for Combinatorial Optimization in VLSIFixed‑Outline Floorplans,” in *Circuit, System and Signal Processing*, vol. 39, pp. 900-918, 2020.
89. M. Jung, T. Song, Y. Peng and S. K. Lim, “Design Methodologies for Low-Power 3-D ICs With Advanced Tier Partitioning,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems*,* vol. 25, no. 7, pp. 2109-2117, 2017.
90. J. Lin and J. Yang, “Routability-Driven TSV-Aware Floorplanning Methodology for Fixed-Outline 3-D ICs,” *in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 11, pp. 1856-1868, 2017.
91. W. Liu, Y. Wang, G. Chen, Y. Ma, Y. Xie and H. Yang, “Whitespace-Aware TSV Arrangement in 3-D Clock Tree Synthesis,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 23, no. 9, pp. 1842-1853. 2015.
92. T. Song, S. Panth, Y. Chae and S. K. Lim, “More Power Reduction With 3-Tier Logic-on-Logic 3-D ICs,”*in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 35, no. 12, pp. 2056-2067, 2016.
93. J. Kadambarajan and S. Pothiraj, “TSV Aware 3D IC Partitioning with Area Optimization,” *Arabian Journal for Science and Engineering (online)*, 2021.
94. 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, Aug. 2011,
95. S. Pothiraj, J. Kadambarajan and P. Kadarkarai, “Floor Planning of 3D IC Design Using Hybrid Multi-verse Optimizer” *Wireless Personal Communication,* vol. 118, pp. 3007–3023, 2021.
96. ISPD’98 Benchmark, Retrieved from <https://vlsicad.ucsd.edu/UCLAWeb/cheese/ispd98.html>
97. MCNC/GSRC Benchmark, Retrieved from <http://vlsicad.cs.binghamton.edu/benchmarks.html>.

**Annexure I**

**Benchmark Circuits**

1. **ISPD’98 Benchmark Details [96]**

In netlist representation a circuit is simply represented as interconnections of components where component can be any entity say, a logic gate or a block depending upon the level of partitioning. It is the simplest form of representation but becomes cumbersome for large circuits making the partitioning process very slow and difficult to handle.

The MCNC routinely issued and maintained benchmark suites for the Design Automation community from 1985 through 1993. The Collaborative Benchmarking Laboratory is actively maintaining these benchmarks.

There are three types of netlist files: ".net",".are" and ".netD". The first two formats have been available since the launch of MCNC benchmark, but the third is a newer format that is similar to ".net" but additionally includes signal direction.

**Net Format**

Each netlist header has five entries, which are   
ignored   
#Pins  
#Nets  
#Modules  
pad offset

The list of nets follows. Each net is simply a subset of modules, which are either cells or pads. Cells are numbered from 0 to pad offset (inclusive). Pads are numbered from 1 to (#Modules - pad offset - 1). Cells are prefaced by an "a", pads by a "p". The beginning of each net is denoted by an "s".

**1.tif**

**Figure1.8:** Circuit for Netlist Representation

Consider the above example with 4 cells, 3 pads,6 nets and 14 pins, the net file is given by   
0   
14   
6  
7   
3   
P0 s 1   
a0 l   
P1 s 1   
a0 l   
a1 l   
a0 s 1   
a2 l   
a3 l   
a1 s 1   
a3 l   
a2 s 1   
p2 l

a3 s 1

p2 1

The .are file is a list of the size of each module. It is simply a list of cells and pads, each followed by its area. In the example, the .are file is given by  
a0 4  
a1 4   
a2 3  
a3 2   
p1 0   
p2 0

p3 0

finally, “.netD” is similar to “.net”, except that each module in a net is identified as an input (I), output (O), or bidirectional (B) pin for that net. In other words, if the module is labeled with ‘I’, it is a net drain and if it is labeled with ‘O’, it is a net source. This allows one to determine signal directions across the circuit. The example's “.netD” file is provided by

0   
14   
6   
7   
3   
P0 s 1 O  
a0 l I  
P1 s 1 O  
a0 l I  
a1 l I  
a0 s 1 O   
a2 l I  
a3 l I  
a1 s 1 O   
a3 l I  
a2 s 1 O   
p2 l I

a3 s 1 O

p2 1 I

1. **MCNC Benchmark details [97]**

The YAL (Yet Another Language) was created by ByranPreas and Ken Roberts at Xerox PARC (Palo Alto Research Center) in 1987 to express the MCNC Benchmark netlist circuits [ref]. The following is a representation of each module in the MCNC Benchmark circuit:

MODULE *<module name>*;

TYPE *<module type>*;

DIMENSIONS *< X*1 *>< Y* 1 *>< X*2 *>< Y* 2 *> . . . < XN >< YN >*

IOLIST;

*<signal name><terminal type>*

[*<Xposition> | < side ><Yposition> | < position >*[*< width >< layer >*]

[*CURRENT < current >*][*VOLTAGE < voltage >*];

.

.

ENDIOLIST;

NETWORK;

*<instance name><module name><signal name> . . .* ;

.

.

ENDNETWORK;

ENDMODULE;

Where

**UPPERCASE** refers to the YAL language's keywords.

*<angle brackets>*refers language’s intermediate definitions, such as module or signal names

*option1|option2 denotes that either option1 or option2 has been selected.*

[square brackets] includes optional elements.

*. . .* Indicates that a previous field or line should be continued as needed.

Each module is identified by its own name and kind. The following are the different module types:

STANDARD: A standard cell is a leaf module (cell) arranged according to the row layout style.

**PAD:** A chip (I/O) pad.

**GENERAL:** A macro cell is a leaf module that is put according to the general layout style.

**PARENT:**A soft module: contains a group of cells.

Each module contains an IOLIST that describes the module's external net connections. The net is identifiable by the Signal name for each connection. The connection's terminal type is one of the following:

**I** : The module accepts the connection as an input.

**O :**The module accepts the connection as an output.

**B :**The bidirectional connection, can either be input or output

**PI :** pad terminal input.

**PO :** pad terminal output.

**PB :** bidirectional pad terminal.

**F :**feed-through signal. The module does not utilize this signal; however, the connection is used to transfer the signal via it.

**PWR :** power connection (VDD).

**GND :** ground connection (VSS).

Depending on its categorization, each module description may include additional information. If a module's properties (e.g., size, terminal positions) are defined, it is characterized as *hard*. Soft is a module with unclear attributes.. A module with undefined characteristics is *soft*.

1. **GSRC Benchmark Details [97]**

The \*.blocks file format contains the name of each block or terminal node in the specified floorplan, as well as additional optional metadata. A single block/terminal node is specified by each line. The format is as follows:

Standard Header

NumSoftRectangularBlocks:  *soft rectangular block nodes number*

NumHardRectilinearBlocks:  *hard rectilinear block nodes number*

NumTerminals:  *terminal (pad etc.) nodes number*

Each soft rectangular block node has one line with the following format:

*Node\_Name* soft\_rectangular *area* *min\_Aspect\_Ratio* *max\_Aspect\_Ratio*

Then, for each hard rectilinear block node, one line (or more if the vertex-list is too large) in the following format:

*Node\_Name* hardrectilinear *vertex\_Number* *vertex1, vertex2, ..., vertexN*

If NumTerminals is greater than zero, then, one line with the following format for each terminal node:

*nodeName* terminal

The default dimension for terminal node is (0, 0)

Where,

***nodeName:***node names are case sensitive comprising of any number of characters from: {(a-z), (A-Z), (0-9), '\_', '/', '\', '+' and '-'}

**softrectangular:**defines the node to be a soft rectangle block.

***area*, *min\_Aspect\_Ratio* and *max\_Aspect\_Ratio:*** The soft rectangular block's area, minAspectRatio and maxAspectRatio are specified.

**Hardrectilinear:** defines the node to be a hard rectangle block.

***vertexNumber:***the hard rectilinear block's number of vertices.

***vertex1, vertex2, ..., vertex:*** list of all vertices of hard rectilinear block in clockwise direction, starting from the left-bottom corner of the hard rectilinear block’s bounding box.

***nodeDimensions:*** indicates width and height when in N orientation.

***terminal* :** indicates that the node is a terminal.

The second file format .Net provides the interconnection between the blocks and terminals. The format is as follows:

Standard Header

NumNets : indicates the number of nets in the specific floorplan

NumPins: indicates the number of pins

Then the details of each net are mentioned as follows:

NetDegree: indicates the number of the blocks/terminals present in the net

<Block/Terminal name><B/U: indicating the block/terminal is bi-directional/unidirectional>

Both the file formats are case sensitive.

**List of Publications**

**International Journals**

1. Atul Prakash & R. K. Lal, “Hybrid PS-ACO Algorithm in Achieving Multiobjective Optimization for VLSI Partitioning” in *International Journal of Control Theory and Applications,* vol.8, No.1, pp. 227-242, 2015*.* **[SCOPUS]**
2. Atul Prakash & R. K. Lal, “Multiobjective VLSI Circuit Partitioning Using ACO for Optimal Solution” in *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)* vol. 4, Issue 5, pp. 1352-1358, 2015*.*
3. Atul Prakash & R. K. Lal, “Floorplanning for Area Optimization Using Parallel Particle Swarm Optimization and Sequence Pair,” in *Wireless Personal Communication,*vol. 118, pp. 323-342, 2021.**[SCIE]**

**International Proceedings**

1. Atul Prakash & R. K. Lal, “PSO: An Approach to Multiobjective VLSI Partitioning” in proc. *IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication systems (ICIIECS*), Coimbatore, India, pp. 1-7, Aug. 2015.