Biosc.Biotech.Res.Comm. Special Issue Vol 13 No 13 (2020) Pp-144-121

BBRC
Bioscience Biotechnology Research Communications

VLSI Floorplan Area Optimization using Swarm Intelligence

Ashwini Desai[1] and Vishal Pattanad[2]
[1]*Department of Electronics and Communication Engineering*
[2]*Department of Electronics and Communication Engineering*
[1,2]*KLE Dr. M. S. Sheshgiri College of Engineering and Technology, Belagavi, India*

ABSTRACT

Floor-planning is a very important stage in the VLSI Physical design process. It determines size, performance, reliability and yield of the VLSI chips. VLSI floor-planning is considered as NP-hard in computational point of view. The modern VLSI technology is associated with fixed outline floor-plan constraint and the objective is to minimize the area and wire-length between the modules. In this paper Particle Swarm Optimization (PSO) algorithm is proposed. PSO is used to optimize the floor-plan area and to represent the floor-plan for non-slicing structure, where area is the physical quantity that is considered for optimization. PSO is an effective swarm intelligence search method which explores the search space during the optimization process to find a near optimal solution. The proposed PSO algorithm is tested with the Microelectronics Centre of North Carolina benchmark circuits (MCNC). The obtained results show that the proposed PSO has better optimization of area of floor-plan with optimal run-time compared to other existing optimization schemes. An area improvement of 7.8% and 11.9% is obtained with MCNC benchmarks ami33 and XEROX10 as compared to the existing methods.

KEY WORDS: OPTIMIZATION, PHYSICAL DESIGN PROCESS, PSO, VLSI FLOORPLANNING.

INTRODUCTION

VLSI physical design process is used to map the circuit components into layout of the circuit. This is accomplished in multiple stages as partitioning, floor-planning, placement, routing and compaction (Sherwani, 1999). Floor-planning is an important stage in the physical design process. It deals with placement of rectangular areas allocated to explicit circuit modules which are to be included on a chip. Every module has millions of cells which perform some arithmetic or logical operation. The goal of floor-planning is to optimize the total chip area which includes area of components and interconnects.

Floor-planning is considered to be NP hard. Many heuristic algorithms are proposed in the literature to solve the floor-planning problem. Non-deterministic methods such as simulated annealing (SA) have been proposed. An iterative approach is used; relaxation of modules and then simulated annealing is used to generate a near optimal solution. The best floor-plan can be chosen from the generated near optimal solutions (Ashwini Desai and Uday Wali, 2020). A modified SA algorithm is discussed by Yifan Weng et al., (2019). The authors applied a two-step strategy, finding a feasible solution and optimizing. The algorithm is proved to be efficient for the fixed-outline floor-planning. De-xuan ZOU et al., (2016) propose another modified SA algorithm which is applied for fixed outline floor-planning. Their results show a stronger capacity to exploit the solution space compared to SA.

ARTICLE INFORMATION

*Corresponding Author: ashwinipri19@gmail.com
Received 11th Oct 2020 Accepted after revision 27th Dec 2020
Print ISSN: 0974-6455 Online ISSN: 2321-4007 CODEN: BBRCBA

Thomson Reuters ISI Web of Science Clarivate Analytics USA and Crossref Indexed Journal

Clarivate Analytics crossref

NAAS Journal Score 2020 (4.31)

Online Contents Available at: http//www.bbrc.in/
Doi: http://dx.doi.org/10.21786/bbrc/13.13/16

Many computational methods viz Particle swarm optimization techniques are used to optimize a floor-planning problem using iterative approach to improve a candidate solution. Zhenyi Chen et al., (2012) have proposed a hybrid PSO that can handle fixed outline floor-planning. The authors applied a linear decrease process to balance between global and local exploration abilities. This reduced the number of computations and iterations to locate the optimum. Tsung-Ying Sun et al., (2006) discuss PSO with B* tree for floor-planning problem to explore the solution space more efficiently than SA. The authors show that the proposed approach exhibits rapid convergence and leads to near optimal solutions.

Venkatraman and Sundhararajan (2017) describe PSO using polish expression to fix modules with fixed outline constraint and a Hybrid Ant colony optimization technique which can reduce the calculation time and can produce the enhanced floor-plan arrangement, it mainly concentrates on giving more suitable floor-plan in less time. Paramasivam S et al., (2016) propose a hybrid method using PSO and Harmony Search (HS) to achieve global optima and local optima. The results obtained show an optimal solution for larger number of modules. Guolong Chen et al., (2008) discuss a Discrete PSO (DPSO) algorithm to explore the search space and to find an acceptable solution. Amarjot Kaur et al., (2016) propose hybrid particle swarm and ant colony optimization (PSO/ACO) algorithm for non-slicing floor-plans to achieve optimized solution. Area optimization during floor-planning is considered to be very important as the area utilized by the floor-plan will decide the overall chip size. It is also difficult to arrive at a lower bound on the estimated area. Hence, a near optimal solution is chosen to be satisfactory.

This paper proposes a PSO algorithm for VLSI floor-plan area optimization. Non-slicing floor-plans are considered for optimization as they are comparatively difficult to optimize than the slicing type of floor-plans. The proposed PSO for the floor-plan optimization can be implemented for any number of modules or blocks. PSO method is easy to understand; each parameter in the PSO can be controlled and can be varied according to the requirements easily. If the number iterations are to be increased or decreased it can be done by changing the assigned value for the variable. The only parameters that vary from one floor-plan to other are the number of blocks and the velocity assigned to each particle. The feasibility of the solution can be known easily and PSO has higher probability in finding the global optima compared to other methods as only few parameters are used.

When there is no overlapping between modules, the solution is said to be feasible until than the solution is infeasible, it helps in knowing if the solution obtained is better or not. The solution can be obtained for any number of modules but at the cost of time, the time required to obtain feasible solution goes on increasing as the number of modules increase. The results in this paper show near optimal solutions can be obtained with proposed PSO as compared to other algorithms.

A. Problem definition: Given a set consisting of n blocks B= $\{b_1, b_2.....b_n\}$, where every block is rectangular in shape with predefined width (w_i) and height (hi) respectively. The overall width (W) and height (H) of the floor-plan design are defined. The main objective of floor-planner is to assign a position to each block within the predefined area of floor-plan (F), such that the blocks do not overlap with each other and there should be a minimum space between each block for the interconnects. Finally the floor-plan has to be optimized so as to obtain a near optimal solution.

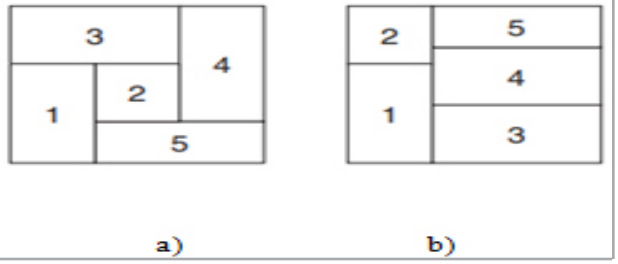
The conditions to be considered during the optimization process are

1. The number of blocks remains the same before and after the optimization process.
2. There should be no overlap between any two or more blocks.
3. Every block must lie in the predefined rectangular region with a predefined aspect ratio.
4. All the blocks should be parallel to the coordinate axis of the predefined floor-plan area F.
5. Taking into consideration the above conditions, PSO algorithm for VLSI floor-plan with area optimization is to be implemented.

B. Floor-planning model: Floor-planning is the placement of flexible blocks with fixed area but unknown dimensions. Floor-planning can be classified as slicing floor-plans and non-slicing floor-plans. The slicing floor-plan can be cut horizontally or vertically, whereas a non-slicing floor-plan cannot be cut either horizontally or vertically. B*-tree can be used to represent a non-slicing floor-plan, where the modules are at cut leaves and the cut types are at the internal nodes.

The floor-plan is divided horizontally (H cut) or vertically (V cut). In horizontally sliced floor-plan, the top or bottom sub-floor-plan is represented by the left or right child whereas; in the vertically sliced floor-plan the left or right sub-floor-plan is represented by the left or right child. The slicing floor-plan may correspond to more than one slicing trees, depending on the order of the cut-line sections. Figure 1.a and b represents the non-slicing floor-plan and the slicing floor-plan respectively.

Figure 1: Floor-plan representations a) non-slicing, b) slicing



Non-slicing floor-plans can be represented by Horizontal constraint graph (HCG) and Vertical constraint graph (VCG). The HCG defines the relations of the modules with respect to horizontal interconnections, and VCG defines the relations of the modules with respect to vertical interconnection of the modules. Figure 2 represents a slicing floor-plan and its slicing tree. V refers to the vertical cut and H refers to the horizontal cut.

Figure 2: Slicing floor-plan and its tree

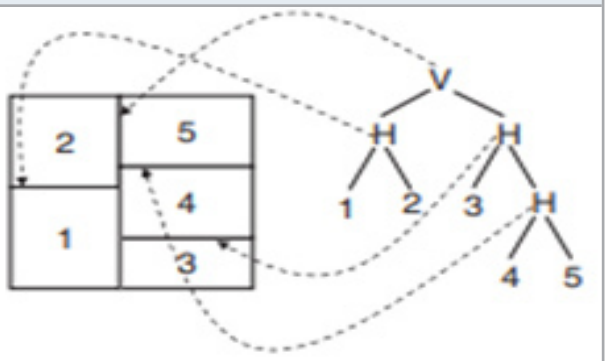
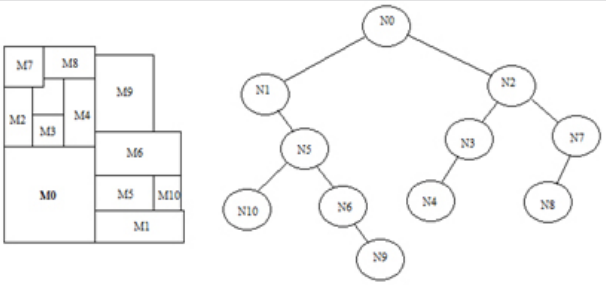


Figure 3: Floor-plan and its B*-tree representation



C. B*-tree representation: B*-tree representation is the binary tree representation of the compact floor-plan. No modules can be moved towards left or bottom in the representation of the compact floor-plan. B*-trees representation is simple. It can be said that the area-optimal floor-plan always refers to some B*-tree. B*-trees are easy to implement; they inherit some properties of the ordered binary trees. They can perform primitive tree operations like search, insert, and delete. The module at the left bottom corner represents B*-tree root. To obtain the root of the B*-tree, sub-tree is constructed first.

The B*-tree has geometric relationship between all the blocks with respect to the nodes of all the trees corresponding to it. In order to obtain the optimized floor-plan, the B*-tree is used for performing certain operations like rotation of the blocks, deletion of blocks, swapping of the blocks and flipping of the blocks according to the floor-plan requirements. Figure 3 shows the floor-plan with different blocks M0 to M10 with its corresponding B*-tree representation.

D. Basic PSO Algorithm: The basic PSO is a stochastic population based search algorithm. It is considered to be an alternate solution for most of the complex non-linear optimization problems. It was introduced by Dr. Kennedy and Dr. Eberhart in the year 1995. The analogy is taken from the flock of birds or insects that migrate from one place to the other in search of food, in the search space. These birds or insects are unaware of the best position in the search space. If any member could find the desirable path from their social behavior, the remaining members will also follow the same path quickly. This algorithm is derived from the behavior or activity of animals to solve the optimization problems. Every member included in the population is a particle and the population of particles is called as a swarm. Initially the population moves in a random direction, every particle moves in the search space in search of best position and it remembers the previous best position of itself and its neighboring particles.

In PSO, particles communicate with every other particle in the swarm for good positions. They dynamically adjust their positions as well as velocity which are derived from the best position of all the particles in the swarm. Until the swarm moves close to an optimum fitness function, all the particles try to find better positions in the search space. The main reason for the popularity of PSO algorithm is its simplicity in the implementation and its ability to converge to a better solution. It uses only the mathematical operators and without any gradient information of the function to be optimized.

The PSO method is more efficient, cheaper and faster when compared with the other existing optimization methods. The problems like non-convex, non-linear, discrete, continuous, integer variable can be easily solved by PSO. It is known that the PSO is a technique impressed by the swarm intelligence. It is the population based evolutionary formula initialized with a population of random solutions. Since the population initialized is random, the particles tend to occupy random places or positions within the boundary initialized for the floor-plan. Each particle in the swarm looks for the best position by communicating with the neighboring particles and occupies the position if it is best suited for it and its neighboring particles.

If the current position is better than the best position, the current position is updated as the best position. Present best position is represented as P_{BEST} and the global best position is represented as GBEST. The P_{BEST} and G_{BEST} are updated until all the iterations are done. Along with updating the P_{BEST} and G_{BEST}, the fitness value is also updated, which in turn gives the cost value. This updating is done until all the particles tend to fly towards higher and higher positions and until the entire swarm moves close to an optimum solution. When all the particles occupy the best positions once, the solution is said to be feasible until then the solution is not feasible. That is if two or more blocks overlap with each other, then the solution is not feasible. Once all the blocks are at their best suited positions without any overlap between the blocks then the solution is said to be feasible.

METHODOLOGY

The proposed optimization method is inspired from the basic PSO algorithm. In this case the population taken into consideration is the number of blocks to be placed in a floor-plan area. Initially the parameters such as population size, number of iterations for the optimization, inertia weights, and the learning coefficients are to be loaded. The modules have different dimensions and are rectangular in shape, the width and height of each module is to be known before the optimization process. Since every particle moves with a certain velocity, a velocity matrix is constructed with a number of rows and columns equal to the number modules.

The steps in the proposed Particle Swarm Optimization (PSO) algorithm are as follows:
Step1: Initialize the population size, number of iterations for the optimization and the
acceleration coefficients.
Step 2: Load the dimensions of each module (width and height), and the parent dimensions.
Initialize the velocities of each particle randomly.
Step 3: Initially the modules take some random positions within the parent region defined.
Step 4: The fitness value of every particle is calculated and best global position GBEST and
best particle position PBEST are updated.
Step 5: The PBEST value is updated with the fitness value if it is better than the previous value
for all iterations.
Step 6: If the GBEST obtained is better than the previous one, update it with the value obtained
in the current iteration.
Step 7: The modules can swap with other modules, or can move from one position to other or
can be rotated for the better position within the parent region.
Step 8: Once all the modules are at their optimum region within the bound and without any
modules overlapping with each other, the cost function is feasible.
Step 9: If the feasibility is not obtained, then increase the number of iterations or the parent
module dimensions.
Step 10: The steps from 4 to 9 are repeated until all the iterations are done.
Step 11: The termination condition is the number of iterations defined.

The main parameters that are considered in PSO algorithm are w, c1, c2, velocity and the size of swarm. These parameters influence the optimization of the search space. The values of w, c1, c2 can be same for all the optimization problems, whereas the velocity and the swarm size differ. The inertia weight w can control the momentum of the particles. If w<1, momentum preserved is very little so the change in the positions/directions of particles can be observed quickly. If w=0, the particles move without any knowledge of the previous velocity of the particle. If w>1 (high), the particles do not change the directions/positions rapidly. If the swarm size is low (from 10 to 20), the optimization becomes easier. All the particles in swarm can find the best position at a less time thereby reducing the total time to optimize the solution. Whereas for higher swarm size, usually the time to optimize the solution is high since every particle has to search and occupy the best position and then optimize which takes more time.

The PSO performs three main operations on the particles/ modules: swap, rotate and move. In swap, two particles swap their positions with their mutual understanding if the position is best suited for them. In move and rotate the individual particle has to change its position or rotate at right angles if the optimum position can be occupied. All the particles perform these three operations until all the particles are placed at the optimum position or until the termination condition is achieved. The termination condition is the number of iterations itself.

Consider a floor-plan region F, which is the area of floor-plan region and all the modules are to be placed on this region. The set of modules can be defined by M= {b1, b2,....bn}. Each module has to be placed in the defined region F such that no modules overlap with each other and all the modules are placed within the region F and a minimum space is to be maintained between each block for the interconnect. A velocity matrix is defined with a matrix size equal to number of modules; if the number of modules is 10 then a velocity matrix of 10*10 is defined with some random values. The dimension of matrix is proportional to the number of modules. To calculate the fitness value of particles the cost function is used and is given by equation 1.

$$f = \frac{1}{cost(f)}$$

The initializations that are to be made are as follows; number of modules, number of iterations, inertia weight and the acceleration coefficients (c1, c2). Further the floor-plan region F is defined with the dimensions of floor-plan region also referred to as the parent. The width and height of the parent is defined. The modules that are to be placed are defined in a set of values with their widths and heights defined:

$w = [w_1, w_2, w_3 ... wn]$

$h = [h_1, h_2, h_3 ... hn]$

Where, w and h corresponds to the set of width and heights respectively. And w_1, w_2, w_3...wn represent the widths of modules and h_1, h_2, h_3 ...hn represent the heights of modules. In PSO, the particle is referred as a point. In this implementation the rectangular blocks are to be placed in the floor-plan region and thus two random points on each rectangular block is assumed rin and rout. These points are used to perform the operation like swap, rotate and move. Delta (d) is the gap between the modules.

Initially the modules are placed at a random position within the floor-plan region F defined. Using a function x=zeros (1, n), the random 1*n matrix is defined for the positions of the modules. Once the optimization process begins, the modules start to occupy the best position. During the optimization process the solution is considered infeasible if two or more modules overlap each other and when all the modules are at the optimum positions, without any overlapping of modules then the solution is considered to be feasible. The solution obtained from this optimization method is random. And the solution obtained can be used for further applications. Initially the swarm distributes the modules in the random positions in the solution space. The velocity of each particle at mth iteration is given by the equation 2.

$$v_{m,n}(K+1) = w * v_{m,n}(K) + c_1 * r_{1n}(K) * [y_{m,n}(K) - x_{m,n}(K)] + c_2 * r_2(K) * [y_n'(K) - x_{m,n}(K)] \quad (2)$$

Where, $v_{m,n}$ is the velocity of m^{th} particle in the n^{th} dimension, w denotes the inertia weight which lies between 0 and 1, y_m' is the global best, c_1 and c_2 are referred to as acceleration coefficients which are defined randomly, and r_{1m} (k) r_{2n} (K) ~ U(0, 1) is some random value usually ranging in [0, 1] which is sampled from the uniform distribution. For each particle (here in this case modules or blocks), its position is updated for every iteration until the termination condition is reached. And this position can be given by equation 3.

$$x_m(K+1) = x_m(K) + v_m(K+1) \quad (3)$$

The personal best and the global best are updated along with the change in the position of the particle for all the iterations. The global best is updated from the personal best if the current personal best is superior than the previously obtained personal best. ym represents personal best position represented by equation 4.

$$y_m(K+1) \begin{cases} y_m(K) & \text{if } f(x_m(K+1) \geq f(y_m(K)) \\ x_m(K+1), & \text{if } x_m(K+1) < f(y_m(K)) \end{cases} \quad (4)$$

The parameters that are considered during optimization are swarm size, inertia weight, number of iterations and acceleration coefficients,

If the swarm size is large then for all the iterations, the large part of search space is covered. But it degrades the parallel search for the position of the particles and the iteration computational complexity also increases. Therefore the small swarm size is optima for the optimization. The acceleration coefficients c_1 and c_2 are the variables that control the overall velocity of the optimization process. If c_1 and c_2 are both zero, the particles are free to move in search space with no change in velocity. If $c_1 > 0$ and $c_2 = 0$, then each particle will find the best position in search space by local search and if the current position is better it replaces the current position. Conversely, if $c_1 = 0$ and $c_2 > 0$, the particles in the swarm start moving towards one single point which is the best position and hence the overlap of the modules occur.

If $c_1 > c_2$ then each particle can be seen moving towards its personal best and if $c_2 > c_1$, then there is the change in the particles motion where the particles now move towards its global best position. If $c_2 = c_1$ then particles move towards the position which is average of the personal best and global best positions. Through inertia weight, momentum of the particles can be controlled, if $w > 1$, for higher inertia weights the particles generally move with high velocity and in this case the direction of the particle is changed so that a new position can be occupied. Whereas if $w < 1$ the particles move with decreasing acceleration until the velocity is reached to zero. If the number of iterations is very low, then the optimization may or may not be completed before all the iterations are over. And if the number of iterations is too high, it leads to the unnecessary computational complexity. So the number of iterations should be chosen wisely.

RESULTS AND DISCUSSION

The PSO algorithm for VLSI floorplan optimization is implemented using MATLAB R2020a version 9.8.0.1323502 and Simulink. The parameters required for the optimization are set as w=0.9, c_1=0.7, c_2=1.5. Maximum number of iterations is MaxIt=1000. Certain parameters like the population or the swarm size, width and height of the parent module (W and H), width and height of modules (w and h), random points on modules (r_{in} and r_{out}), spacing between the modules (d) and the velocity matrix which is n*n matrix where n corresponds to number of particles are to be defined for every new optimization problem.

The PSO algorithm is applied for a random floor-plan with 10 modules and 29 modules initially. Figure 4.a shows the random initial floor-plan with 10 modules. To search for the best position in the search space, initially all the blocks are placed at random positions without any overlap. Further the best position is attained by checking for the current position of each block and the minimum space between two blocks. For every iteration, if the global best is better than the personal best, the position of the block is updated else the position does not change. Figure 4.b shows the near optimal floor-plan obtained for the random initial floor- plan with 10 blocks. PSO algorithm is also applied for random floor-plan with 29 modules for which the initial and near optimal floor-plans are shown in figure 5.a and 5.b.

Table 1 shows the experimental results for a random floor-plan with 10 modules and 29 modules. For each of the cases the initial floor-plan area and the optimized floor-plan area and time elapsed to complete the optimization is mentioned. For floor-plan with 10 modules and 29 modules, an area improvement of 15.7% and 12.4% respectively is seen as compared to the initial input floor-plan.

Figure 4.a: Random initial Floor-plan for 10 modules

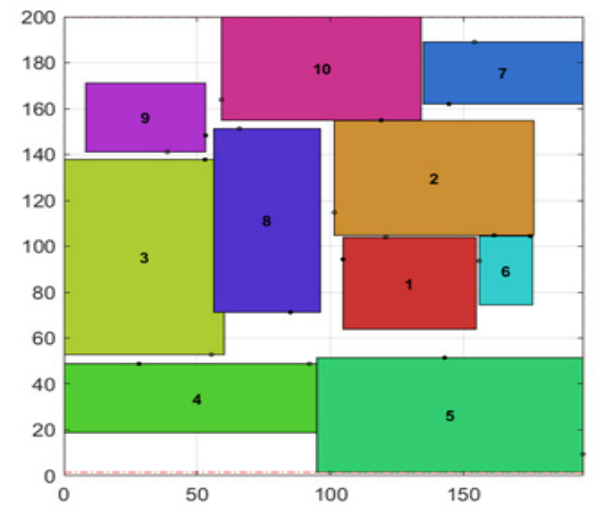


Figure 4.b: Near optimal Floor-plan for 10 modules

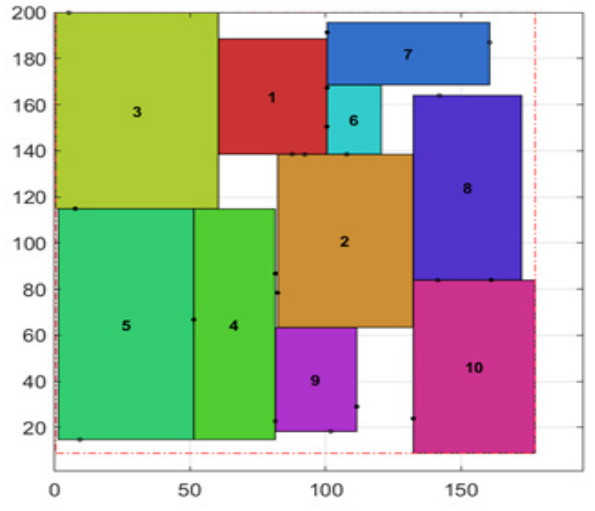


Figure 5.a: Random initial Floor-plan for 29 modules

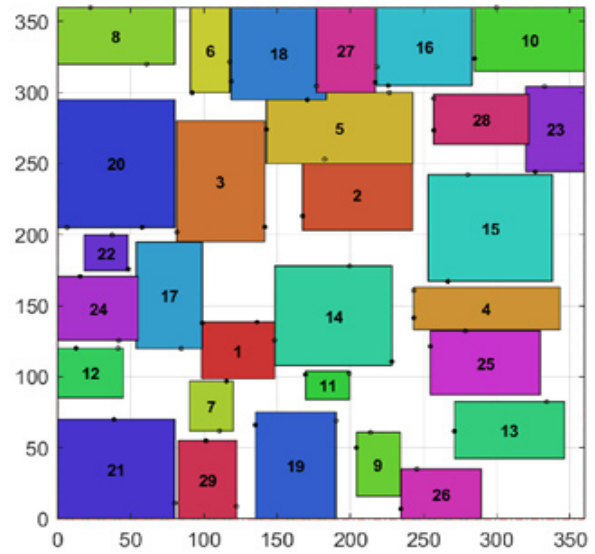


Figure 5.b: Near optimal Floor-plan for 29 modules

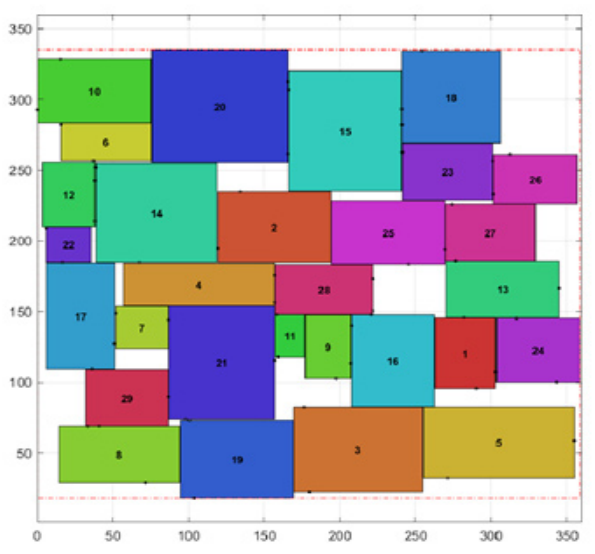


Table 1. Experimental Results Of Proposed Pso For Floor-Plan With 10 And 29 Modules

No. Of Blocks	Parent [height, width] (mm^2)	Initial floorplan area (mm^2)	Area Optimized (mm^2)	Time elapsed (in seconds)
10 blocks input	[200,195]	0.039	0.032872	187.89801
29 blocks input	[360,360]	0.1296	0.113492	509.28092

The proposed PSO algorithm was tested with ami33 and XEROX10 MCNC benchmark circuits. Figure 6 and 7 show the near optimal floor-plan for MCNC ami33 and XEROX10 benchmark circuits. Table 2 shows comparison of the proposed PSO with Hybrid PSO/ACO (Amarjot Kaur et al., 2016) and Discrete Particle Swarm Optimization (DPSO) (Guolong Chen et al., 2009).

The results obtained from the proposed PSO algorithm show an area improvement of 7.8% over Hybrid PSO/ACO and DPSO for MCNC ami33 benchmark. For MCNC benchmark XEROX10 an improvement in area of 4.1% over Hybrid PSO/ACO and 11.9% over DPSO is obtained from the proposed PSO. Therefore it is observed that the results obtained from the PSO method are better compared to other existing methods.

Figure 6: Near optimal Floor-plan for ami33 benchmark

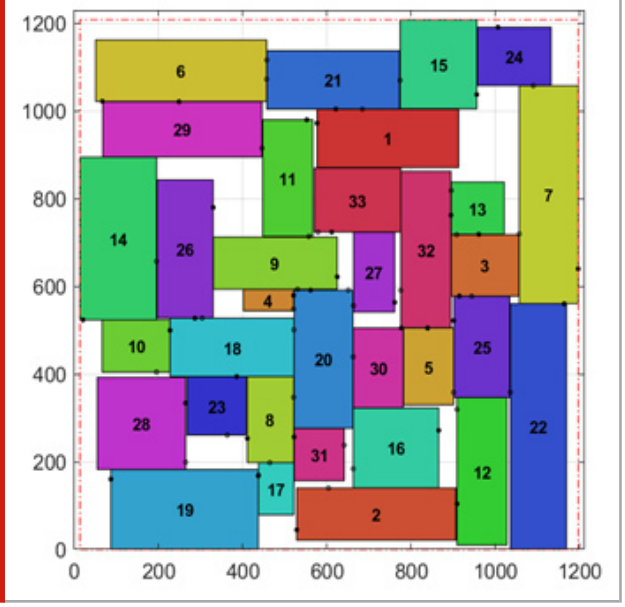


Figure 7: Near optimal Floor-plan for XEROX10 benchmark

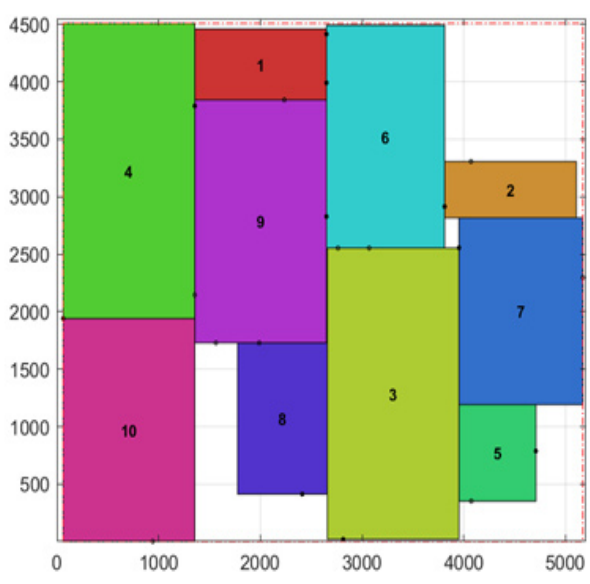


Table 2. Results Of Comparison With Existing Methods and Proposed Method

MCNC Benchmark	Number of modules	Hybrid PSO/ACO	DPSO	Proposed PSO
		Area optimized in mm^2		
ami33	33	1.28	1.28	1.3863
XEROX10	10	21.70	20.2	22.66

CONCLUSION

In this paper, a PSO algorithm for VLSI floor-plan optimization is implemented. The experimental results show that PSO is the best scheme for floor-plan optimization. PSO produces better results when compared to the other existing methods. Initially the implementation was done on few modules with lesser floor-plan area in order to check the feasibility. Table 1 shows parent height and width chosen, the total area optimized and time elapsed for two initial floor-plans with 10 and 29 modules. Area improvement of 15.7% and 12.4% was obtained for floor-plan with 10 and 29 modules respectively as compared to the initial floor-plan area. Since the number of iterations was set to 1000, the elapsed time is high.

The performance of the algorithm was tested with MCNC benchmark circuit ami33 and XEROX10 and the results obtained are shown in Table 2. The proposed PSO algorithm was compared with the existing methods Hybrid PSO/ACO (Amarjot Kaur et al., 2016) and Discrete Particle Swarm Optimization (DPSO) (Guolong Chen et al., 2009). The results show that the area optimization for floor-plan is better in case of PSO method. An improvement in floor-plan area of 7.8% was seen in Hybrid PSO/ACO and DPSO with MCNC benchmark ami33 whereas an area improvement of 4.1% over Hybrid PSO/ACO and 11.9% over DPSO was achieved with MCNC benchmark XEROX10.

REFERENCES

Amarjot Kaur, Dr. Sandeep Singh Gill (2016) Hybrid Swarm Intelligence for VLSI Floor-plan. International Conference on Computing, Communication and Automation (ICCCA2016) ISBN: 978-1-5090-1666-2/16 Pages 224 – 229.

Ashwini Baligatti, Ashwini Desai ,Uday Wali (2014) Free Area Estimator for Simulated Annealing of VLSI Floor Plans. International Journal of Innovative Research in Computer Science & Technology ISSN: 2347-5552 Volume-2 Issue-4 July-2014 Pages 52 – 55.

Ashwini Desai, Ashwini Baligatti, Harsha Pal, Savita Y. Barker, Uday Wali (2014) Development of EDA Tool with Easy Plugin for New VLSI Algorithms. International Journal of Engineering Science and Innovative Technology Volume 3 Issue 4 July 2014 Pages 354 -359.

Ashwini Desai, Uday Wali (2020) Effect of Module Order on VLSI Floor Planning using Simulated Annealing. IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT 2020) Electronic ISBN: 978-1-7281-6828-9 IEEE Xplore 16 September 2020.

De-xuan ZOU, Gai-ge WANG, Gai PAN, Hong-wei QI (2016) A modified simulated annealing algorithm and an excessive area model for floor-planning using fixed-outline constraints. Frontiers of Information Technology & Electronic Engineering ISSN 2095-9184 (print) ISSN 2095-9230 (online) November 2016 Pages 1228 – 1244.

Guolong Chen, Wenzhong Guo, Hongju Cheng, Xiang Fen and Xiaotong Fang (2008) VLSI Floor-planning Based on Particle Swarm Optimization. Proceedings of 2008 3rd International Conference on Intelligent System and Knowledge Engineering IEEE Xplore 30 December 2008 ISBN:978-1-4244-2196-1 Pages 1020 – 1025.

Paramasivam, S., Athappan, S., Natrajan, E.D. and Shanmugam, M. (2016) Optimization of Thermal Aware VLSI Non-Slicing Floor-planning Using Hybrid Particle Swarm Optimization Algorithm-Harmony Search Algorithm. Circuits and Systems Vol 7 No 5 April 2016 doi.org/10.4236/cs.2016.75048, Pages 562-573.

S. Venkatraman, M. Sundhararajan (2017) Particle Swarm Optimization Algorithm for VLSI Floor-planning Problem. Journal of Chemical and Pharmaceutical Sciences ISSN: 0974-2115 JCPS Volume 10 Issue 1 January - March 2017 Pages 311 – 316.

S.Venkatraman, Dr.M.Sundhararajan (2017) Optimization for VLSI Floor-planning Problem by using Hybrid Ant colony Optimization technique. International Journal of Pure and Applied Mathematics Volume 115 No. 6 2017 ISSN: 1311-8080 (printed version) ISSN: 1314-3395 (on-line version) Pages 637-642.

Sherwani N. (1999) Algorithms for VLSI Physical Design Automation. 3rd. Ed., Kluwer Academic Publishers.

Tsung-Ying Sun, Sheng-Ta Hsieh,Hsiang-Min Wang and Cheng-Wei Lin (2006) Floorplanning Based on Particle Swarm Optimization. IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI'06) IEEE Xplore 6 March 2006 ISBN:0-7695-2533-4.

Yifan Weng, Zhen Chen, Jianli Chen, Wenxing Zhu (2019) A Modified Multi-objective Simulated Annealing Algorithm for Fixed-outline Floor-planning. IEEE International Conference on Automation Electronics and Electrical Engineering Electronic ISBN: 978-1-5386- 7861-9.

Zhenyi Chen, Gaofeng Wang, Chen Dong (2012) Hybrid Particle Swarm Optimization Algorithm for Fixed-outline Floorplanning. International Conference on Computer Science and Network Technology IEEE Xplore 12 April 2012 Electronic ISBN: 978-1-4577-1587-7 Pages 1299 – 1302.