MINIMIZATION OF FLOOR PLANNING AREA USING PARTICLE SWARM OPTIMIZATION

A Thesis Submitted

In Partial Fulfillment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

NAME OF THE STUDENTS

1) **BITTU LAL (12/EC/034)**
2) **SWAPNIL DUBEY (12/EC/045)**
3) **SHOBHAN KUMAR BISWAS (12/EC/068)**
4) **POLINA PAVAN KUMAR (12/EC/088)**
5) **ARJUN V ANCHAN (12/EC/090)**
6) **ELITA ASTRID LOBO (12/EC/094)**

Under the Supervision

of

DR. RAJIB KAR

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

National Institute of Technology Durgapur

West Bengal – 713209

India

May, 2016

NATIONAL INSTITUTE OF TECHNOLOGY

DURGAPUR

CERTIFICATE

It is certified that the work contained in the thesis entitled "**Minimization of Floor Planning Area Using Particle Swarm Optimization** " has been carried out by **"Elita Astrid Lobo (12/EC/094),"** under the guidance of **"DR. Rajib Kar"**. The data reported herein is original and that this work has not been submitted elsewhere for any other Degree or Diploma.

Elita Astrid Lobo

Place.....Durgapur.....:

Date...10.5.16.....:

This is to certify that the above declaration is true.

10.05.16

DR. Rajib Kar

Place: .....

Date.....:

Assistant Professor

Department of Electronics & Communication Engg.

National Institute of Technology

Durgapur - 713209, India

NATIONAL INSTITUTE OF TECHNOLOGY

DURGAPUR

Abstract

Floor planning is an essential design step for hierarchical building module design methodology. Floor planning provides early feedback that evaluates architectural decisions, estimates chip area, estimates delay and congestion caused by wiring. As technology advances, design complexity is increasing and the circuit size is getting larger. To cope with the increasing design complexity, hierarchical design and Intellectual Property modules are widely used. This makes floor planning much more critical to the quality of a Very Large Scale Integration (VLSI) design. For many years, floor planning is a critical step, as it sets up the ground work for a layout. However, it is computationally quite hard. The process of determining block shapes and positions with area minimization objective and aspect ratio requirement is referred to as floor planning. Common strategy for blocks floor planning is to determine in the first phase and then the relative location of the blocks to each other based on connection-cost criteria. In the second step, block sizing is performed with the goal of minimizing the overall chip area and the location of each block is finalized. From the computational point of view, VLSI floor planning is NP-hard. The solution space will increase exponentially with the growth of circuits scale, thus it is difficult to find the optimal solution by exploring the global solution space. To handle this complexity swarm based optimization method has opted in this proposed Project. A generalize solution has developed to take care of area as well as interconnection wire length. To achieve this weighted objective function has defined. The advantages of PSO like simplicity in implementation, not depends upon the characteristics of objective function and better performance have given support to include it as a solution method.

Acknowledgement

We would like to articulate our deep gratitude to our project guide **Dr. RAJIB KAR,** who has always been our motivation for carrying out the project. His constant inspiration and effort made this project work a great success. We are thankful to him for his contributions in completing this project work. An assemblage of this nature could never have been attempted without reference to and inspiration from the works of others whose details are mentioned in reference section. We acknowledge our indebtedness to all of them. Last but not the least we would like to thank our parents and the Almighty.

Date:

Bittu Lal (12/EC/034),
Swapnil Dubey (12/EC/045),
Shobhan Kumar Biswas (12/EC/068),
Polina Pavan Kumar (12/EC/088),
Arjun V Anchan (12/EC/090)
Elita Astrid Lobo (12/EC/094)
Department of Electronics and Communication
National Institute of Technology Durgapur
West Bengal – 713209, India

Table of Contents

List of Figures

Literature review 1: Temporal Floorplanning Using the T-tree Formulation

Theory:

Improving logic capacity by time-sharing, dynamically reconfigurable FPGAs are employed to handle designs of high complexity and functionality. In this paper, each task has been modelled as a 3D-box and deal with the temporal floorplanning/placement problem for dynamically reconfigurable FPGA architectures. A tree-based data structure has been presented, called T-trees, to represent the spatial and temporal relations among tasks. Each node in a T-tree has at most three children which represent the dimensional relationship among tasks. For the T-tree, we develop an efficient packing method and derive the condition to ensure the satisfaction of precedence constraints which model the temporal ordering among tasks induced by the execution of dynamically reconfigurable FPGAs.

Result:

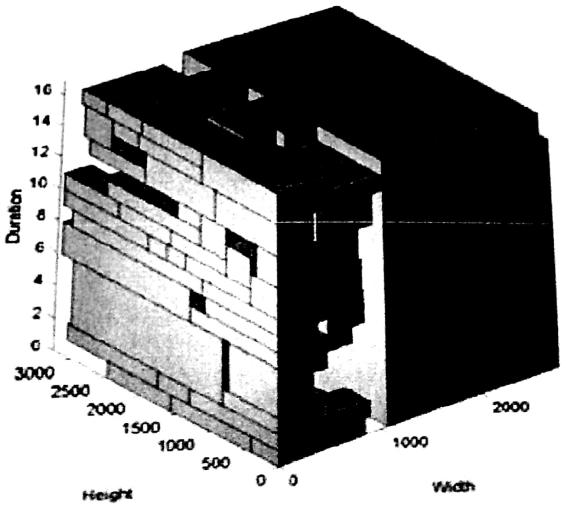

Figure .1.1 The result of 3D-ami49 with simultaneous volume and wirelength optimization

Conclusions:

Compared with the 3D-subTCG, the T-tree can achieve better solution quality in less time. This makes it suitable for large circuits. The main reasons why the T-tree is more efficient than the 3D-subTCG is three-fold: (1) the packing time of T-tree is faster, (2) the size of the solution space is smaller than the 3D-subTCG, and (3) the solution space of the T-tree is well-structured. Thus, the annealing engine can devote more time to explore the solution space, resulting in better solution quality and less running time.

Reference:

[1] Ping-Hung Yuh1, Chia-Lin Yang1, Yao-Wen Chang2, "Temporal Floorplanning Using the T-tree Formulation," Proc. RAW, Apr. 2004.

Literature review 2: Floorplanning Based on B-Tree and Fast Simulated Annealing

Theory:

Floorplanning with bus planning is one of the most challenging modern floorplanning problems because it needs to consider the constraints with interconnect and block positions simultaneously. A B*-tree is an ordered binary tree for modelling nonslicing or slicing floorplans. Given an admissible placement (in which no blocks can move left or down), we can construct a unique B*-tree in linear time to model the placement. Further, given a B*-tree, we can also obtain a legal placement by packing the blocks in amortized linear time with a contour structure. In this paper, the authors study two types of modern floorplanning problems: 1) fixed-outline floorplanning and 2) bus-driven floorplanning (BDF). This floorplanner uses B*-tree floorplan representation based on fast three-stage simulated annealing (SA) scheme called Fast-SA. For fixed-outline floorplanning, the authors present an adaptive Fast-SA that can dynamically change the weights in the cost function to optimize the wirelength under the outline constraint. Experimental results show that this floorplanner can achieve100%success rates efficiently for fixed-outline floorplanning with various aspect ratios. For the BDF, the authors explore the feasibility conditions of the B*-tree with the bus constraints, and develop a BDF algorithm based on the conditions and Fast-SA.

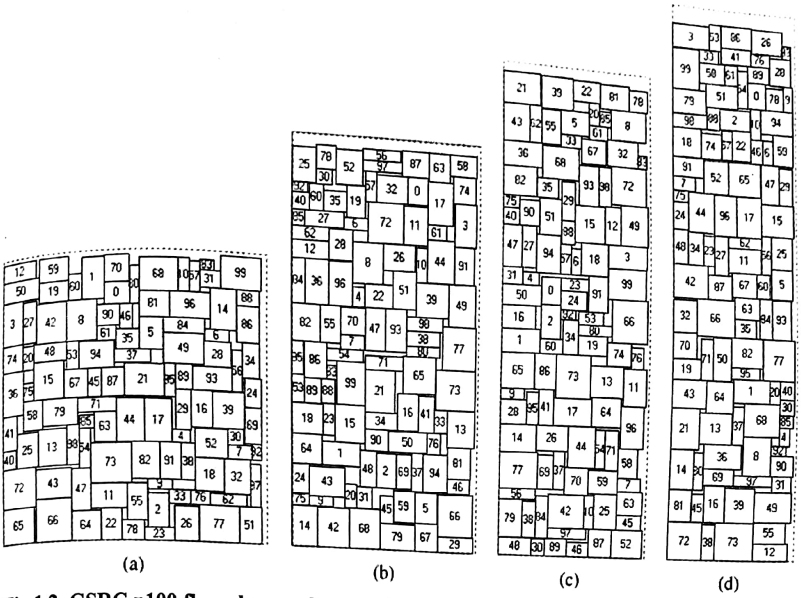

Fig.1.2. GSRC n100 floorplan results with Γ=10%, and desired aspect ratios *Rs are (a) 1, (b) 2, (c) 3, and (d) 4. Dead spaces are (a) 5.57 %, (b) 5.06%, (c) 5.03%, and (d) 4.70%. Dotted line is fixed outline.**

Conclusions:

Algorithms for the modern floorplanning problems with fixed-outline and bus constraints, based on our new Fast-SA scheme and the B*-tree representation have been proposed. Experimental results have shown that the Fast-SA leads to faster and more stable convergence to the desired floorplan solutions.

Reference:

[1] Tung-Chieh Chen, "Modern Floorplanning Based on B*-Tree and Fast Simulated Annealing," Ieee Transactions On Computer-Aided Design Of Integrated Circuits And Systems, v. 25, No. 4, April 2006.

CHAPTER I
INTRODUCTION

Physical design begins with a floor plan. The floor plan estimates the area of major units in the chip and defines their relative placements. The floor plan is essential to determine whether a proposed design will fit in the chip area budgeted and to estimate wiring lengths and wiring congestion, so an initial floor plan should be prepared as soon as the logic is loosely defined Based on the area of the design and the hierarchy, a suitable floor plan is decided upon. Floor planning takes into account the macros used in the design, memory, other IP cores and their placement needs, the routing possibilities and also the area of the entire design. Floor planning also decides the IO structure, aspect ratio of the design. A bad floor plan will lead to wastage of die area and routing congestion. To overcome the problem of floor planning an automated system which taken the input as DFG by the technological help of the PSO has been defined. The problem of the floor planning has been transferred as a problem of constraint optimization to take care of non over lapping requirement. The purposed method is not only taking care of required area for the floor planning but also how to minimize the interconnected wire length is also involved. By considering the following constraints such as:

- Non overlapping constraint: No two modules should overlap.
- Wire length estimation: Wire length between two modules is calculated by calculating the distance between centers of two modules.
- Area Estimation: It's the area of rectangle of minimum size, enclosing all the blocks.
- Cost Function & Constraints: A floor plan has an area cost, i.e., which is measured by the area of the smallest rectangle enclosing all the modules and an interconnection cost, i.e., Wire length, which is the total length of the wires fulfilling the interconnections between the modules.

The goal of floor planning is to optimize a predefined cost function; the floor plan area directly correlates to the chip silicon cost.

Floor planning is an important stage in VLSI design. It's an NP-hard combinatorial optimization problem. The solution space of the problem will increase exponentially with the growth of circuits scale, and thus it is impossible to find the optimal solution by exploring the

global solution space. The random optimization algorithms can be adopted to solve floor planning problem.

In many design methodologies, area and speed are considered to be trade off against each other. The reason is that, there are limited and more routing resources that are used for design. Due to this, designed system will operate slower. Optimizing for minimum area allows the design to use fewer resources, but also allows the sections of the design to be closer together. This leads to shorter interconnect distances, less routing resources to be used, faster end-to-end signal paths, more consistent place and routing times.

The particle swarm optimization (PSO) is an optimization technique inspired by swarm intelligence and theory in general such as bird flocking, fish schooling and even human social behaviour. PSO is a population-based evolutionary algorithm in which the algorithm is initialized with a population of random solutions. However, unlike most of other population-based evolutionary algorithms, PSO is motivated by the simulation of social behaviour instead of the survival of the fitness. The advantages of PSO over many other optimization algorithms are its simplicity in implementation and its ability to converge to a reasonably good solution quickly. Since the PSO algorithm was proposed, it has aroused great interest among the academic community, massive research results have been presented in only a few years. Hence particle swarm optimization technique will be made use to find the optimal solution for the floor planning.

CHAPTER 2
Particle swarm optimization algorithm

The particle swarm optimization (PSO), originally introduced by Kennedy and Eberhart, is an optimization technique inspired by swarm intelligence and theory in general such as bird flocking, fish schooling and even human social behaviour. PSO is a population-based evolutionary algorithm in which the algorithm is initialized with a population of random solutions. However, unlike most of other population-based evolutionary algorithms, PSO is motivated by the simulation of social behaviour instead of the survival of the fitness. The advantages of PSO over many other optimization algorithms are its simplicity in implementation and its ability to converge to a reasonably good solution quickly. Since the PSO algorithm was proposed, it has aroused great interest among the academic community, massive research results have been presented in only a few years.

The basic PSO algorithm consists of three steps namely, generating particle's positions and velocities, velocity update, and finally position update. Here a particle refers to a point in the design space that changes its position from one move (iteration) to another based on velocity updates.

First, the positions x_k^i and velocities v_k^i, of the initial swarm of particles are randomly generated using upper and lower bounds on the design variables values, x_{min} and x_{max}, as expressed in Eq. (1) and Eq. (2) . The positions and velocities are given in a vector format with the superscript and subscript denoting the i^{th} particle at time k. In Eq. (1) and Eq. (2), rand is a uniformly distributed random variable that can take any value between 0 and 1. This initialization process allows the swarm particles to be randomly distributed across the design space [6].

$$x_0^i = x_{min} + rand(x_{max} - x_{min}) \quad \dots \quad (1)$$

$$v_0^i = x_{min} \frac{x_{min} + rand(x_{max} - x_{min})}{\Delta t} = \frac{\text{position}}{\text{time}} \quad \dots \quad (2)$$

The second step is to update the velocities of all particles at time k+1 using the particles objective or fitness values which are functions of the particles current positions in the design

space at time k. The fitness function value of a particle determines which particle has the best global value in the current swarm p_k^g, and also determines the best position of each particle over time p^i, i.e. in current and all previous moves. The velocity update formula uses these two pieces of information for each particle in the swarm along with the effect of current motion v_k^i, to provide a search direction v_{k+1}^i, for the next iteration. The velocity update formula includes some random parameters, represented by the uniformly distributed variables, *rand,* to ensure good coverage of the design space and avoid entrapment in local optima. The three values that effect the new search direction, namely, current motion, particle own memory, and swarm influence, are incorporated via a summation approach as shown in Eq. (3) with three weight factors, namely, inertia factor, w, self confidence factor, c_1 and swarm confidence factor, c_2 respectively [8].

$$v_{k+1}^i = w\, v_k^i + c_1 rand \frac{(p^i - x_k^i)}{\Delta t} + c_2 rand \frac{(p_k^g - x_k^i)}{\Delta t} \quad \ldots \quad (3)$$

The research presented in this paper found out that setting the three weight factors w, c_1, and c_2 at 0.5, 1.5, and 1.5 respectively provides the best convergence rate for all test problems considered. Other combinations of values usually lead to much slower convergence or sometimes non-convergence at all.

Position update is the last step in each iteration. The Position of each particle is updated using its velocity vector as shown in Eq. (4) and depicted in Fig.1.

$$x_{k+1}^i = x_k^i + v_{k+1}^i \Delta t \quad \ldots\ldots \quad (4)$$

In PSO, the design variables can take any values even outside their side constraints, based on their current position in the design space and the calculated velocity vector. This means that the design variables can go outside their lower or upper limits, x_{min} or x_{max}, which usually happens when the velocity vector grows very rapidly; this phenomenon can lead to divergence. To avoid this problem, in this study, whenever the design variables violate their upper or lower design bounds, they are artificially brought back to their nearest side constraint. This approach of handling side constraints is recommended by reference [4] and is believed to avoid velocity "explosion". There has been no recommendation in the literature regarding swarm size in PSO. Most researchers use a swarm size of 10 to 50 but there is no well established guideline.

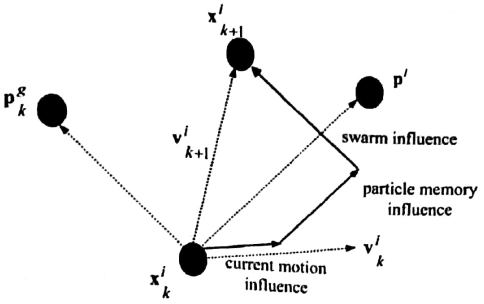

Fig. 2.1. Depiction of the velocity and position updates in Particle [1]

2.1. Selection of Parameters for PSO:

The main parameters of the PSO model are ω, C_1, C_2, Vmax and the swarm size S. The settings of these parameters determine how it optimizes the search-space. For instance, one can apply a general setting that gives reasonable results on most problems, but seldom is very optimal. Since the same parameter settings not at all guarantee success in different problems, we must have knowledge of the effects of the different settings, such that we can pick a suitable setting from problem to problem [4].

2.2. The Inertia Weight ω:

The inertia weight ω controls the momentum of the particle: If $\omega << 1$, only little momentum is preserved from the previous time-step; thus quick changes of direction are possible with this setting. The concept of velocity is completely lost if $\omega = 0$, and the particle then moves in each step without knowledge of the past velocity. On the other hand, if ω is high (>1) we observe the same effect as when C_1 and C_2 are low: Particles can hardly change their direction and turn around, which of course implies a larger area of exploration as well as a reluctance against convergence towards optimum. Setting $\omega > 1$ must be done with care, since velocities are further biased for an exponential growth. This setting is rarely seen in PSO implementation and always together with Vmax. In short, high settings near 1 facilitate global search and lower settings in the range [0.2, 0.5] facilitate rapid local search.

The decreasing ω-strategy is a near-optimal setting for many problems, since it allows the swarm to explore the search-space in the beginning of the run, and still manages to shift towards a local search when fine-tuning is needed.

This was called PSO-TVIW method (PSO with Time varying inertia weight). According to Eberhart and Shi devised an adaptive fuzzy PSO, where a fuzzy controller was used to control ω over time. This approach is very interesting, since it potentially lets the PSO self-adapt ω to the problem and thus optimizes and eliminates a parameter of the algorithm. This saves time during the experimentation, since fine-tuning of ω is not necessary anymore. At each time-step, the controller takes the Normalized Current Best Performance Evaluationl and the current setting of ω as inputs, and it outputs a probabilistic change in ω.

2.3.The Maximum Velocity Vmax:

The maximum velocity Vmax determines the maximum change one particle can undergo in its positional coordinates during iteration. Usually we set the full search range of the particle's position as the Vmax. For example, in case, a particle has position vector x = (x1, x2, x3) and if $-10 \leq xi \leq 10$ for i = 1, 2 and 3, then we set Vmax = 20. Originally, Vmax was introduced to avoid explosion and divergence. However, with the use of constriction factor χ or ω in the velocity update formula, Vmax to some degree has become unnecessary; at least convergence can be assured without it. Thus, some researchers simply do not use Vmax. In spite of this fact, the maximum velocity limitation can still improve the search for optima in many cases.

2.4. The Constriction Factor χ:

An adaptive PSO model that uses a new parameter 'χ' called the constriction factor. The model also excluded the inertia weight ω and the maximum velocity parameter Vmax. The velocity update scheme proposed by Clerc [9], [11]. Constriction coefficient results in the quick convergence of the particles over time. That is the amplitude of a particle's oscillations decreases as it focuses on the local and neighborhood previous best points. Though the particle converges to a point over time, the constriction coefficient also prevents collapse if the right social conditions are in place. The particle will oscillate around the weighted mean of p^i and p^g, if the previous best position and the neighborhood best position are near each other the particle will perform a local search. If the previous best position and the neighborhood best position are far apart from each other, the particle will perform a more exploratory search (global search). During the search, the neighborhood best position and previous best position will change and the particle will shift from local search back to global search.

The constriction coefficient method therefore balances the need for local and global search depending on what social conditions are in place.

2.5. *The swarm size:*

It is quite a common practice in the PSO literature to limit the number of particles to the range 20–60. There is a slight improvement of the optimal value with increasing swarm size, a larger swarm increases the number of function evaluations to converge to an error limit. According Eberhart and Shi [12] illustrated that the population size has hardly any effect on the performance of the PSO method

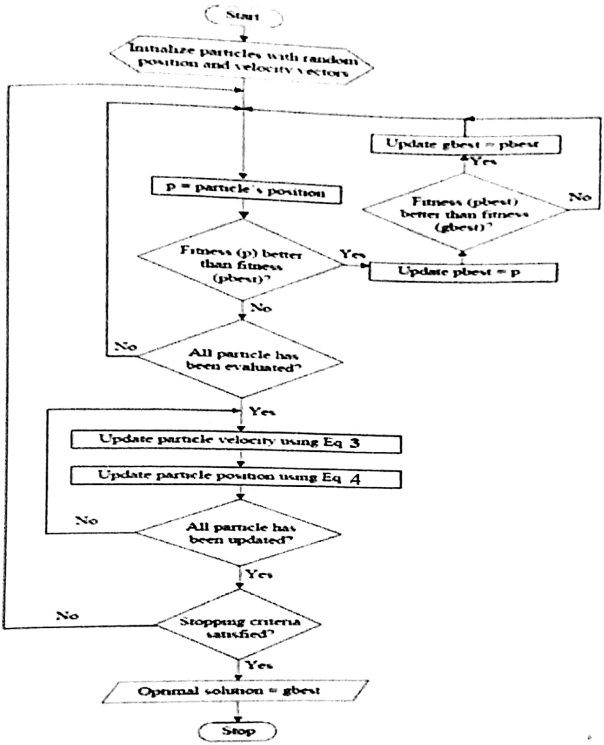

Fig. 2.2. Flowchart of particle swarm optimization algorithm [1]

CHAPTER 3

Floorplanning using particle swarm optimization algorithm

During Floorplanning the information is given to a set of macro cells, the information includes their width, length and cell numbers. The goal of proposed method is to plan all macro cells positions on a chip such that nonoverlap constraint can be satisfied and the area and interconnection cost is minimized. In this paper, the notation c(x, y, i) is used to denote the i^{th} cell location, where x and y are i^{th} cell position on x-axis and y-axis respectively. Note that, the cell positions are defined as the cell lower left corner. Fig. 3.1 illustrates these definitions.

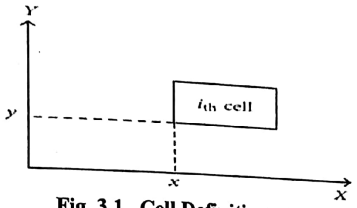

Fig. 3.1. Cell Definition [1]

3.1. Non-Overlapping Constraint:

Two modules Mi and Mj are nonoverlap, if at least one of the following constraints is satisfied.

Mi to the left of Mj : $xi + wi <= xj$

Mi below Mj : $yi + hi <= yj$

Mi to the right of Mj : $xi - wj >= xj$

Mi above Mj : $yi - hj >= yj$

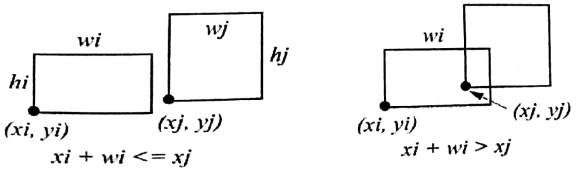

Fig. 3.2. Two modules overlapping [1]

3.2. Area Estimation:

It's the area of rectangle of minimum size, enclosing all the blocks as shown in Fig. 3.3. Therefore the total area will be

$$Area\ (F) = \left(\max(xi + wi) - \min(xi)\right)\left(\max(yi + hi) - \min(yi)\right) \quad \text{..... (5)}$$

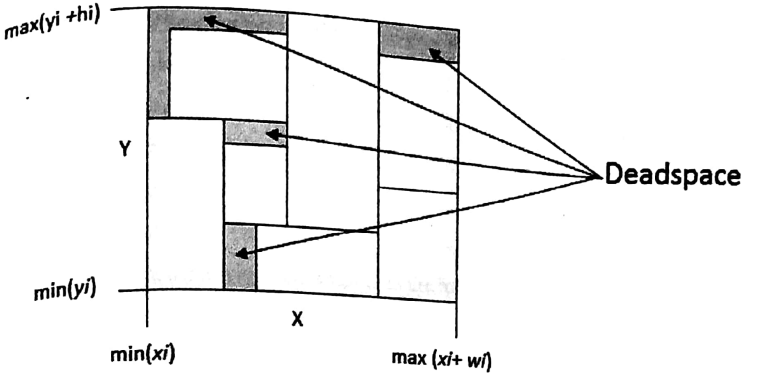

Fig. 3.3. Area estimation of the given floorplan [1]

3.3. Cost Function & Constraints:

A floorplan has an area cost, i.e., which is measured by the area of the smallest rectangle enclosing all the The cost of a floorplan F is defined as follows:

$$Cost(F) = Area(F) + Tv \times 10^{4} \quad \text{..... (6)}$$

Tv is the total overlap violation, which is the total Euclidean distance Between 2 modules which overlap.

3.4. Fitness Function:

The VLSI floorplanning is a minimization problem, and the objective is to minimize the cost of floorplan F, i.e., cost (F). Thus, the fitness of an individual in the population is defined as follows

$$f(x, wh) = \frac{1}{cost(F)} \quad \text{..... (7)}$$

Where, $f(x, w_h)$ is the corresponding floorplan of (x, w_h), $cost(F)$ is the cost of floorplan defined in Eq. (8), x is a matrix which has the (x,y) location of each module and w_h is a matrix which has corresponding width and height of each module.

3.5.Algorithm:

Sort blocks in decreasing order of size

For each block do the following:

1. Apply Particle Search Optimization Algorithm to find the best position of the block with respect to previous blocks:

a. Initialize swarm particles with random position and velocity vectors.

Initialize local best position and fitness of each particle.

Initialize gbest

% gbest = Best particle in the current swarm

b. For each particle in the swarm, do the following:

% p = Particle's position

% pbest = Local best position of particle p

if current block overlaps with previous blocks:

fitness(p) = 0

else

fitness(p) = 1/(Area of outbounding rectangle)

If fitness(p) is better than fitness of particle at pbest and position p does not overlap with previous blocks:

Update pbest = p

if fitness(p) is greater than fitness of gbest at its pbest position:

Update gbest

Update particle velocity

Update particle position

c. Repeat step b until atleast one non overlapping position has been found and number of times step b has been repeated is atleast 10,000.

2. Fix the block at local best position of the best particle in the current swarm (gbest).

Print block positions

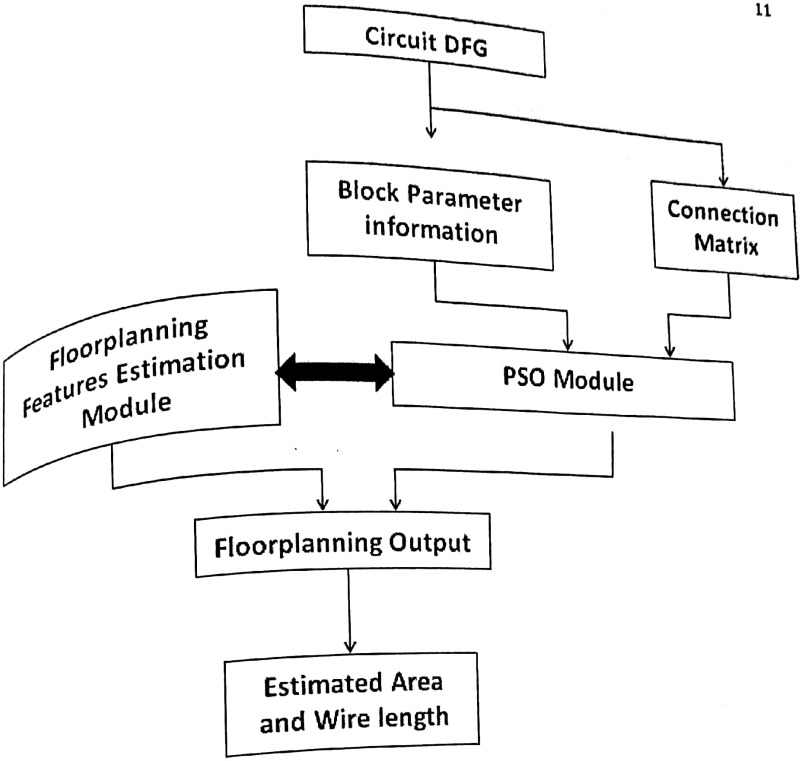

Fig. 3.4. Flow Chart of Floorplanning Using PSO [1]

CHAPTER 4
Experimental results

In the floorplanning modules are represented by the set of rectangular blocks which depends upon the module operation. The following results show the convergence of the blocks to a minimum Area. Each figure represents blocks with different Areas and different no of blocks.

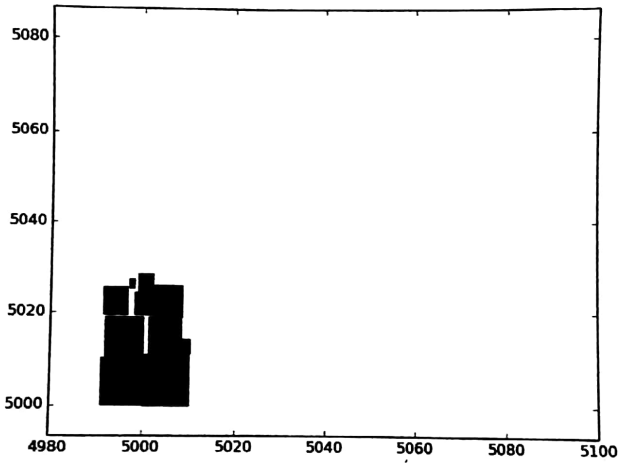

Fig . 4.1. No of Blocks: 10(Dimensions: Random)

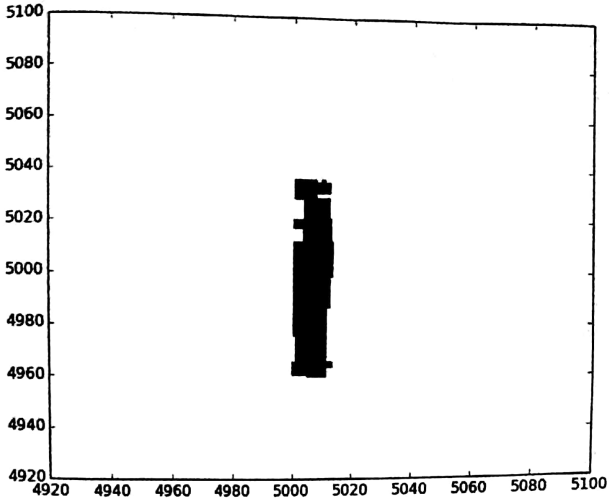

Fig. 4.2. No of Blocks: 13(Dimensions: Random)

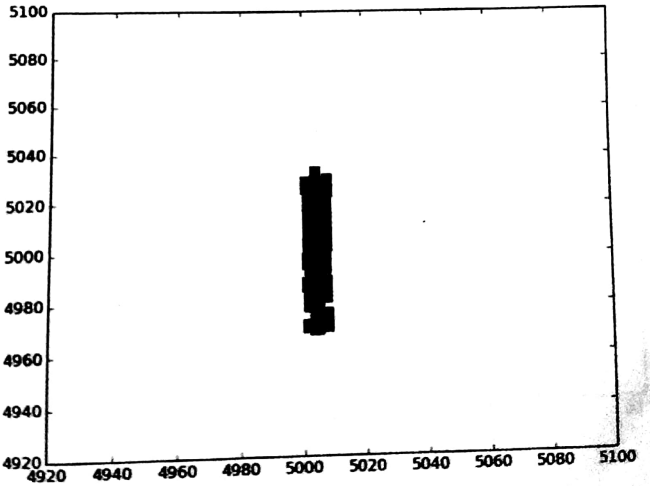

Fig. 4.3. No of Blocks: 15(Dimensions: Random)

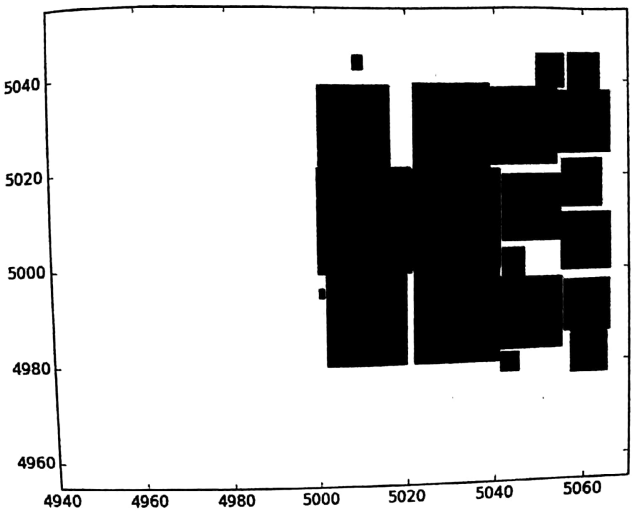

Fig.4.4. No of Blocks: 20(Dimensions: Random)

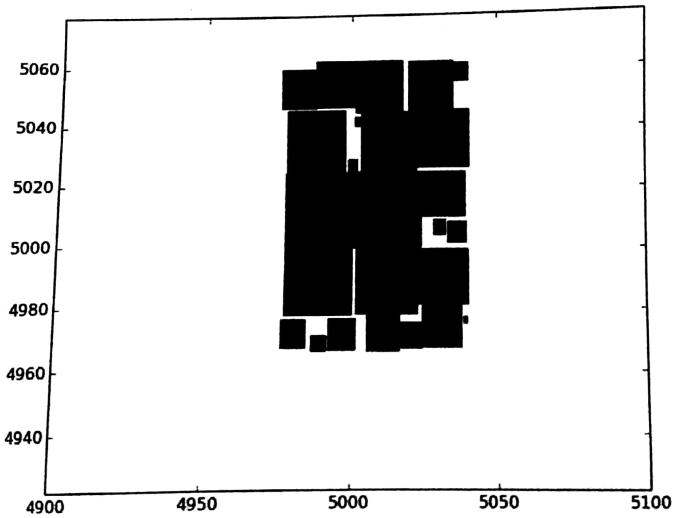

Fig. 4.5. No of Blocks: 25(Dimensions: Random)

CHAPTER 5

Conclusion

The problem in physical design has lot of diversity. Floorplanning is one of the process manually crafted which results in time consumption and less efficiency. To overcome this problem an automated system which takes the input as DFG by the technological help of the PSO has been defined. The problem of the floorplanning has been transferred as a problem of constraint optimization to take care of non overlapping requirement. The proposed method is takes care of required area for the floorplanning.

The concept of the PSO algorithm has been used because, the number of advantages like, simplicity in the math logic, not much problem dependency and efficient solution.

With the proposed solution we hope that researchers will find a more comfort methodology to obtain the optimal floorplanning and cost effective.

The solution has been proposed in the respect to defining the floorplanning by means of particle swarm optimization. Extending this algorithm to include total wirelength required to connect all blocks in cost and try to minimize the wire length cost as well can be considered as one of the future scopes.

References

[1] Dain Palupi Rini, Siti Mariyam Shamsuddin & Siti Sophiyati Yuhaniz —Particle Swarm Optimization: Technique, System and Challengesl. International Journal of Computer Applications (09758887), Vol 14-No.1,January 2011.

[2] Mohammad Syafrullah, NaomieSalim, —Improving Term Extraction Using Particle Swarm Optimization Techniques,l journal of computing, volume 2, issue 2, february 2010.

[3] Sheng-Ta Hsieh, Tsung- Ying Sun Chan-Cheng-Liu and Cheng-Wei Lin Research article — An Improved Particle Swarm Optimizer for Placement Constraintsl. Hindawi Publishing Corporation ,Journal of Artificial Evolution and Applications January 2008.

[4] Swagatam Das, Ajith Abraham and Amit Konar. — Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives l,Studies in Computational Intelligence 2008.

[5] A review of particle swarm optimization. Part II: hybridization, combinatorial, multicriteria and constrained optimization, and indicative applicationsl By Alec Banks Æ Jonathan Vincent Æ Chukwudi Anyakoha Received: 25 August 2006 / Accepted: 4 June 2007 / Published online: 17 July 2007 Springer Science+Business Media B.V. 2007

[6] Rania Hassan, Babak Cohanim, Olivier de Weck, Gerhard Venter. —A Comparison of Particle Swarm Optimization and the Genetic Algorithm.l AIAA-50; 2005-1897. 46th AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Materials Conference, Austin, Texas, 2005.

[7] S. T. Hsieh, C. W. Lin and T. Y. Sun, —Particle Swarm Optimization for Macrocell Overlap Removal and Placement,l in Proc. of IEEE Swarm Intelligence Symposium (SIS'05), pp. 177-180, June 2005

[8] Venter, G. and Sobieski, J., —Particle Swarm Optimization,l AIAA 2002-1235, 43rd AIAA/ASME/ASCE/ AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO., April 2002.

[9] Clarc, M and Kennedy, J (2002). —The particle swarm – explosion, stability, and convergence in a multidimensional complex space.l IEEE Transactions on Evolutionary Computation. pp. 58-73

[10] Kennedy, J., Eberhart, R.C., and Shi, Y.(2001). —Swarm Intelligence,l San Francisco: Morgan Kaufmann Publishers.

[11] Clerc, M. (1999). —The swarm and the queen: towards a deterministic and adaptive particle swarm optimization.l Proc. 1999 Congress on Evolutionary Computation, Washington, DC, pp 19511957. Piscataway, NJ: IEEE Service Center.

[12] Y. Shi and R. Eberhart, —A modified particle swarm optimizer,l in Proceedings of IEEE World Congress on Computational Intelligence, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[13] M. Rebaudengo and M. Reorda, —GALLO: A genetic algorithm for floorplan area optimization,l IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 15, no. 8, pp. 943–951, Aug. 1996

[14] R. Eberhart and J. Kennedy, —A new optimizer using particle swarm theory,l in Proceedings of 6th International Symposium on Micro Machine and Human Science, pp. 39–43, Nagoya, Japan, October 1995.

[15] Yi-Chun Xu, Ren-Bin Xiao & Martyn Amos —Particle Swarm Algorithm for Weighted Rectangle Placement

[16] Andrew Kahng, Jens Leinig, Igor L. Markov, Jin Hu, —VLSI Physical Design: From Graph Partitioning to Timing Closure,l Springer, 2011