ISSN: 0974-2115
www.jchps.com Journal of Chemical and Pharmaceutical Sciences

Particle Swarm Optimization Algorithm for VLSI Floorplanning Problem

S. Venkatraman*, M. Sundhararajan
Department of Electronics and Communication Engineering, Research scholar, Bharath University, Chennai, India
***Corresponding author: E-Mail: apece2008@gmail.com**

ABSTRACT

Floor planning is that the very basic stage in VLSI physical design for classified building module design methodology. The floorplanning is employed to calculate to the relative location of blocks within the fixed outline. The planning obscurity is increasing and therefore the circuit size is obtaining additional. Thus ultimately area of the circuit gets rise and harder to optimize the Wirelength and area. This leads the terribly high attention to tend for VLSI floorplanning. So our objective is to attenuate the chip area and fix the modules or blocks within the fixed outline constraints. During this paper we focus the particle swarm optimisation (PSO) methodology to achieve global solution for fixed outline constraints for this we tend to taken MCNC and GSRC benchmark circuits.

KEY WORDS: VLSI Floor planning, Particle Swarm Optimization (PSO), Chip area.

1. INTRODUCTION

The tense growth in technology for very large scale integration (VLSI) circuit style and producing has managed to entire systems with countless semiconductor unit being placed on one chip. Due to the high complexness of recent chip design, VLSI CAD tools are dynamic for delivering high VLSI system performance. For many issues in layout style, the machine complexness is NP-hard (Sherwani, 1999). The longer term nice growth of VLSI circuits can consider the event of physical design automation tools. A traditional floorplanning formulation close to decide the layout of a given set of modules, specified no overlapping modules. Based on the circuit style and also the hierarchy, an appropriate floorplan is determined. An immoral floorplan can cause wastage of die space and cant ready to succeed fixed outline constraints.

The proposed technique the Particle Swarm optimisation algorithm (PSO) can offer the most effective best resolution for VLSI fixed outline constraints floorplanning. The immoral floorplanning to be thought-about as a drag of constraint optimisation to require care of no overlapping constraint. The purposed technique is giving additional attention of needed space for the floorplanning however additionally the way to minimize white area and the way to attain fixed outline constraints. By considering the subsequent constraints such as: area Estimation: It‘s the area of rectangle of minimum size, enclosure all the blocks. Cost Function & Constraints: A floorplan 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. PSO could be a swarm intelligence technique that roughly models the social behavior of swarms. The consequence of modeling this social behavior is that the search method permits particles to stochastically come back toward antecedently productive regions within the search area. It’s tested to be economical on several issues in science and engineering. Our technique will scale back the maximum amount attainable time period, obtains higher solutions. Moreover, our floorplanner will wide explore the answer house and forestall the answer from falling into the native borderline.

The particle swarm optimization (PSO), is a random improvement technique supported the movement and swarm intelligence. PSO is initialized with the population of random solution and it applies the idea of social interaction to problem determination. In contrast to most of different population-based evolutionary algorithms, PSO is driven by the imitation of shared behavior rather than the survival of the fitness. The benefits of PSO area unit simplicity in implementation and its ability of merging is incredibly fast. (Sun et al. 2006) originally introduces PSO into the floorplanning problem. The paper adopts the B*-tree floor planning structure (Chang, 2000) to come up with an initial stage with overlap free for floorplan and utilizes PSO to seek out the global resolution. However, implementation detail of the algorithmic rule is mentioned, and solely the area improvement is taken into account.

2. EXISTING METHODS

Mathematical programming primarily based floorplanners use a interconnect estimate because the objective perform to be reduced with a constraint on the floorplan area. Kim and Kim planned an applied mathematics approach to optimize area and interconnect at the same time. Their approach uses a linear program with area constraints to optimize interconnect followed by an occasional simulated annealing method exploitation the one normalized weighted total approach to enhance the answer quality. Sheqin et al planned a quadratic programming primarily based floorplanner to optimize interconnect followed by a deterministic algorithm based on Less Flexibility first (LFF) principles to provide the ultimate floorplan.

Sowmya et al handle the complexness of VLSI floorplanning, swarm based optimisation technique has opted in this paper work. A generalize resolution has developed to take care of area further as wire length. To realize this weighted objective perform has outlined. The benefits of PSO like simplicity in implementation, not depends upon

the characteristics of objective perform and higher performance have given support to incorporate it as a solution technique.

Shanavas et al minimizing the Wirelength contend a vital important} role in physical design automation of very large-scale integration finding an optimum resolution for VLSI physical design parts like partitioning and floorplanning. In VLSI circuit floorplanning, the matter of minimizing semiconductor area was additionally a stock. Memetic algorithm (MA) applied some form of local search for optimisation of VLSI partitioning and floorplanning. The formula combined a hierarchical design technique like genetic formula and constructive technique like simulated annealing for local search to unravel VLSI partitioning and floorplanning problem. MA will quickly turn out optimum solutions for the popular benchmark.

Modern VLSI floor planning, as outlined by Kahng, focuses on interconnect optimisation inside a set chip outline. With the chip complexness increasing with the rising integration technology, hierarchical design strategies became imperative. During a hierarchical design flow, floorplanning at the topmost-level may need a versatile chip define. However the floorplans for the modules of the upper levels can fix the floorplan define for the lower level sub-modules. This has crystal rectifier to an accumulated importance for the trendy fixed outline floorplanning problem. It's to be noted that in fashionable floorplanning, interconnect is that the primary objective whereas space isn't any longer an objective however rather a constraint.

VLSI floor planning could be a well-studied drawback that a range of optimisation techniques are applied together with simulated annealing (SA), mathematical programming, and genetic algorithms. Early floorplanners restricted space optimisation alone. However with the arrival of the deep sub-micron regime, floorplanners shifted their focus to optimizing interconnect. However if interconnect is that the solely objective to be optimized, the ensuing floorplan can have lots of unused area. Hence, some floorplanners tried to optimize each area and interconnect. Within the single normalized weighted sum (SNWS) approach to multi-objective optimisation, concurrent optimisation of two objectives implies that the optimizer uses equal weights to multiply the normalized objectives before adding them along to get the one normalized weighted sum. Classical (outline-free) floorplanners supported Simulated tempering use the one normalized weighted sum approach to optimize the two objectives, specifically area and total interconnect. These SA-based floorplanners dissent solely within the system, Sequence pair, or transitive Closure Graph, used to represent the floorplans.

Problem Statement: Let M be the set of modules represented by M=, wherever N is that the number of modules. Every module mi is represented by (Wi, Hi), wherever 1≤i≤N, wi is dimension of the module mi and Hi is that the height of the module mi. The ratio of mi is outlined as Hi / wi. The area Ai of the module mi is given by wi *Hi. There are three completely different types of rectangular modules specifically soft modules, hard modules and pre-placed modules. The soft modules have variable ratio at intervals fixed vary and fixed space. In hard modules, each space and ratio area unit mounted structure. The elaborate description is given as follows:

Slicing floorplan: A slicing floorplan is obtained by cutting the floorplan either horizontally or vertically repetitively. Fig.1 (a) represents slicing floorplan. A slicing tree could be a binary tree. The pre-placed module could be a one during which modules coordinates' area unit given by the floorplanner. Let H denotes set of hard modules, S denotes set of soft modules and P denotes set of preplaced modules. Let M be the union of those three sets of modules. The illustration of floorplanning will be exhausted two layout forms, specifically the slicing structure and non-slicing that is used to represent a slicing floorplan. Generally, there are two cut sorts, + and -. The + (-) represents floorplan horizontal (vertical) cut. Fig.1 (b) shows a slicing tree of

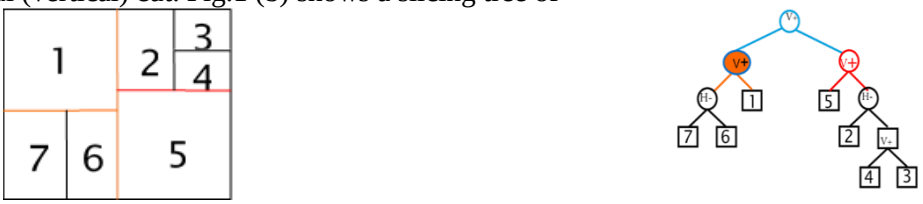

Figure.1. (a) slicing floorplan **(b) slicing tree**

Non slicing floorplan: Non slicing floorplan is more common than slicing floorplan. All the children of the given cell cannot be obtained by bisecting the floorplan. This is called non-slicing floorplan. Horizontal constraint graph and vertical constraint graph can be used to model a non-slicing floorplan. In a constraint graph, a node represents a module.

Figure.2. (a). Non slicing floorplan Vertical and Horizontal Constraint graph

The foremost aim of this paper to minimize the dead space (white space) & fix the module in fixed outline constraint. In this paper we dealt with slicing floorplanning

Representation of Floorplan: Polish Notation is used to model Slicing Floorplans. The Binary Tree is used to mention a Polish notation, E= e1 e2...e2n-1 where ei € {1, 2... n, H+, V-}. Here, each number represents a module and H+, V- represents a horizontal and vertical cut respectively in the slicing floorplan. The Polish expression is the postfix ordering of a binary tree, which can be obtained from the post-order traversal on a binary tree. Polish expression length is 2n1, where n – is number of modules.

Figure.3. Polish Expression

Particle Swarm Optimization (PSO): PSO is an improvement technique impressed by swarm intelligence. PSO could be a population-based evolutionary formula within which the formula is initialized with a population of random solutions. During this algorithm essentially learned from animal's activity or behavior to resolve improvement issues. Every member of the population is named a particle and also the population is named a swarm. Beginning with an at random initialized population and occupation at random chosen directions, every particle goes through the looking area and remembers the simplest previous positions of itself and its neighbors. Every particles of swarm communicate its best positions to every alternative. Consequent step begins once all particles are moved. Finally, all particles tend to fly towards higher and higher positions over the looking method till the swarm move to close to an optimum of the fitness operate.

The procedural flow of Particle Swarm Optimization

Step 1: Load the modules and initialize the parameters of the PSO algorithm.

Step 2: Generate an initial population with particle dimension corresponding to the number of modules to be optimized and initialize its positions.

Step 3: Calculate the fitness value of each particle using area and then assign the fitness values to its corresponding particles. Let the initial global best be the lowest Pbest value.

Step 4: Update the velocity of the particle.

Step 5: In the consecutive iterations check every particle. If its fitness value is better than its corresponding previous Pbest, then update its Pbest along with the fitness value and particle.

Step 6: Update Gbest for each and every iteration. If the earlier Gbest is higher than the Gbest obtained in current iteration, then update newer one as the final Gbest.

Step 7: Repeat step 3 to step 6 till the termination condition is reached. The termination condition or stopping criteria may be the end of number of iteration or the repetitive occurrence of the same output for certain number of iterations specified by the user.

Step 8: halt the process, if termination condition is satisfied.

Area Estimation: It's the area of rectangle of minimum size, enclosing all the blocks as shown in Fig 4.

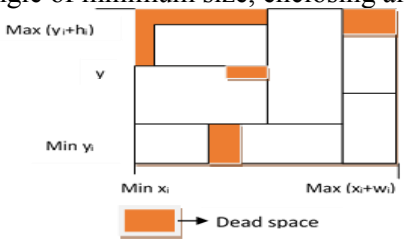

Figure.4. Block position & dead space

Therefore the total area will be *Area (F)* = (max (x_i+ w_i) – min (xi)) (max (y_i+ h_i)–min (yi))

Fitness Function: The VLSI floor planning 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) = 1/ cost (f): Where, f(x, w) is the corresponding floorplan of (x, wh), cost (F) is the cost of floorplan, x is a matrix which has the (x, y) location of each module and w is a matrix which has corresponding width and height of each module.

The basic PSO algorithm consists of three steps namely, generating particles '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 and velocities of the initial swarm of particles are randomly generated using upper and lower bounds on the design variables values, x, as

expressed in xi + wi<= xj and yi + hi <= yj. The positions and velocities are given in a vector format with the superscript and subscript denoting the I^{th} min particle at time k. xi + wi <= xj and yi + hi <= yj, 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.

$$X_O^i = X_{min} + Rand\ (X_{max} - Xmin)$$

$$V_o^i = X_{min} \qquad \frac{Xmin + Rand(Xmax - Xmin)}{\Delta t}$$

The second step is to update the velocities of all particles at time k+1 using the particles objective or fitness values that are functions of the particles current positions within the design area at time k. The fitness function value of a particle determines that particle has the most effective global value within the current swarm, and additionally determines the most effective position of every particle over time pi, i.e. in current and every one previous moves. The speed update formula uses these 2 items of data for every particle within the swarm alongside the result of current motion, to provide a search direction, for subsequent iteration. The speed update formula includes some random parameters, represented by the uniformly distributed variables, rand, to confirm sensible coverage of the look area and avoid denial in local optima. The three values that result the new search direction, namely, current motion, particle own memory, and swarm influence, are incorporated via a summation

$$V^t_{k+1} = WV^t_k + C_1\, rand\ (P^t - X^t_k)/\Delta t + C_2\, rand\ (P^t - X^t_k)/\Delta t$$

From the above formula, with three weight factors, namely, inertia factor, w, self-confidence factor c1 and swarm confidence factor, c2 respectively. The research presented during this paper found out that set the three weight factors w=0.4, c1, c2=1.5, respectively provides the best merging rate for all test problems considered. Other combinations of values usually lead to much slower convergence or sometimes non-convergence at all. 2. Location update is the final step in each iteration. The Position of each particle is updated using its velocity vector as shown in below;

$$Xt\ K+1 = Xtk + Vtk+1\ \Delta t$$

In PSO, the design variables will take any values even outside their aspect constraints, supported their current position within the design area and also the calculated speed vector. this implies that the design variables will go outside their lower or higher limits, x min or x max that typically happens once the velocity vector grows very rapidly; this development will cause divergence. To avoid this drawback, during this study, whenever the design variables violate their higher or lower design bounds, they're artificially brought back to their nearest aspect constraint.

This approach of handling aspect constraints is suggested by reference [4] and is believed to avoid velocity explosion. There has been no recommendation within the literature concerning swarm size in PSO. Most researchers use a swarm size of 10 to 50 however there's no well-established guideline.

The swarm size: It is quite a common apply within the PSO literature to limit the quantity of particles to the vary 40–100. There's a small improvement of the optimum value with increasing swarm size, a bigger swarm will increase the quantity of perform evaluations to converge to an error limit.

The Inertia Weight ω: The inertia weight ω controls the momentum of the particle: If $\omega << 1$, solely very little momentum is preserved from the previous time-step; so fast changes of direction area unit potential with this setting. The thought of speed is totally lost if $\omega = 0$, and therefore the particle then moves in every step while not data of the past velocity. On the opposite hand, if ω is high (>1) we tend to observe an equivalent result as once C1 and C area unit low: Particles will hardly amendment their direction and switch around, that after all implies a bigger space of exploration in addition as a reluctance against convergence towards optimum. Setting $\omega >> 1$ should be through with care, since velocities area unit additional biased for an exponential growth.

This setting is never seen in PSO implementation and forever alongside Vmax. In short, high settings close to =1'facilitate international search and lower settings within the vary [0.2, 0.4] facilitate speedy local search. A pair of the decreasing ω-strategy could be a near-optimal setting for several issues, since it permits the swarm to explore the search-space within the starting of the run, and still manages to shift towards a neighborhood search once fine-tuning is required

Velocity Vmax: The maximum velocity Vmax determines the most modification one particle will bear in its point coordinates throughout iteration. typically we have a tendency to set the complete search vary of the particle's position because the Vmax. As an example, in case, a particle has position vector x = (x1, x2, x3) and if $-10 \leq xi \leq$ ten for i = one, 2 and 3, then we have a tendency to set Vmax = twenty.

Originally, Vmax was introduced to avoid explosion and divergence. However, with the utilization of constriction issue χ or ω within the velocity update formula, Vmax to a point has become unnecessary; a minimum of convergence may be assured without it. Thus, some researchers merely don't use Vmax. In spite of this truth, the most rate limitation will still improve the explore for optima in several cases.

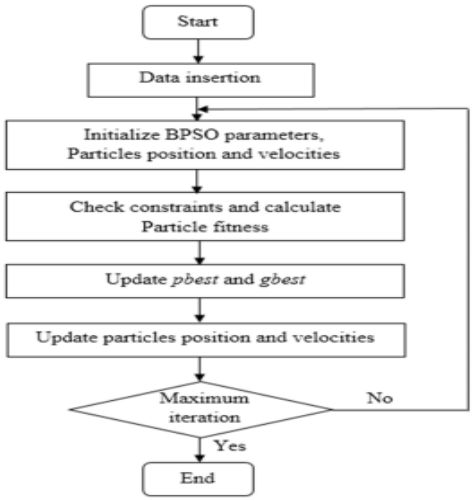

Figure.5. Flow chart of Particle Swarm Optimization algorithm

3. RESULTS AND CONCLUSION

The experiments during this study used GSRC and MCNC benchmarks for the proposed floorplanner and compare with simulated annealing (SA) and fast simulated annealing (FSA). All the cells were set as hard modules. The simulation programs were compiled in MATLAB, and also the results were obtained on a Pentium 4 1.7 GHz with 512MB RAM. The PSO experiments with w, c1 and c2 initializations were 0.4, 1 and 1.5, severally. The particle range is about as five. We have a tendency to run the each floorplanner ten times and calculated their average outcomes of chip area and run time. The experiment results of each floorplanner are shown in Table one. Compare with SA and FSA, our methodology will notice an improved floorplan solution in even less computation time. Beneath an equivalent tree structure, that's to mention, our technique has a lot of efficiency and solution searching ability for floorplan. Though the SA in adopted three an equivalent operations that mentioned higher than, however it'd randomly pick up the operation (somewhat sort of a reasonably trial and error strategy) however not following the previous expertise whereas making an attempt to seek out another higher resolution. This may lead to the floorplanner waste an excessive amount of time on trapping into native minimal and tougher to get a stronger solution. Our technique will overcome these drawbacks. Thus, the appropriate resolution will find out in shorter computational time. Relative to each strategies, our technique possesses a lot of strength to prevent the solution from falling into local minimal. It'd be useful to seek out a stronger solution in shorter time.

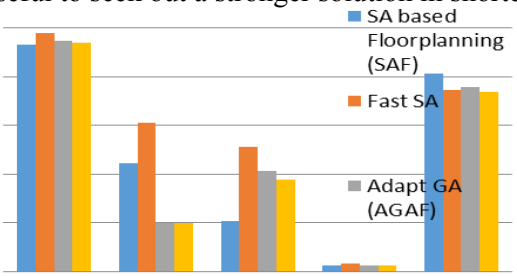

Figure.6. Area Comparison of MCNC Benchmark Circuits

The proposed VLSI floor planning based on Particle Swarm Based Optimization with polish expression on fixing modules with in fixed outline constraint. The PSO ability to identify the feasible solution for soft and hard modules instead of simulated annealing, fast simulated annealing (FSA).

Table.1. Area comparison for MCNC benchmark circuits.

Benchmark circuit	SA based Floor planning (SAF)	Fast SA	Adapt GA (AGAF)	Particle swarm optimization (PSO)
	46.56	48.85	47.3	46.95
	22.22	30.58	10.05	10.02
	10.28	25.59	20.56	18.83
	1.327	1.69	1.2	1.3
	40.66	37.16	37.81	36.89

The graph clearly explains the experimental result of floorplanning delivers global solution with GSRC benchmark circuits compared with SA, FSA algorithm.

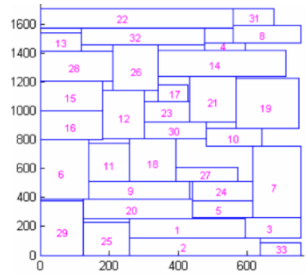

Figure.7. Floorplanning result of ami 33circuit

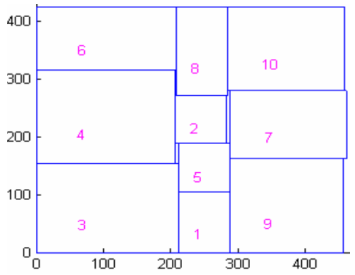

Figure.8. Floorplanning result of Xerox circuit

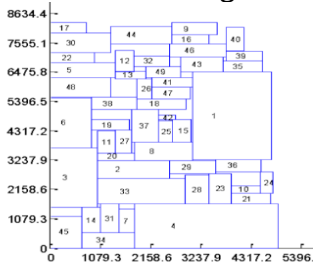

Figure.9. Floor planning of ami49 benchmark circuit

REFERENCES

Alec Banks, Jonathan Vincent AE, Chukwudi Anyakoha, A review of particle swarm optimization, Part II, hybridization, combinatorial, multicriteria and constrained optimization, and indicative applications, Springer Science Business Media B.V, 2007, 330.

Clarc M and Kennedy J, The particle swarm, explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, 2002, 58-73.

Clerc M, The swarm and the queen, towards a deterministic and adaptive particle swarm optimization, Proc 1999 Congress on Evolutionary Computation, Washington, DC, Piscataway, NJ, IEEE Service Center, 1999, 1951-1957.

Dain Palupi Rini, Siti Mariyam Shamsuddin & Siti Sophiyati Yuhaniz, Particle Swarm Optimization, Technique, System and Challenges, International Journal of Computer Applications 14 (1), 2011, 0975-8887.

Hsieh S.T, Lin C.W and Sun T.Y, Particle Swarm Optimization for Macro cell Overlap Removal and Placement, in Proc. of IEEE Swarm Intelligence Symposium (SIS‘05), 2005, 177-180.

Jae-Gon K, and Yeong-Dae K, A linear programming-based algorithm for Floor planning in VLSI design, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 22 (5), 2003, 584-592, 2003.

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

Rebaudengo M and Reorda M, GALLO, A genetic algorithm for floorplan area optimization, IEEE Trans. Computer.-Aided Design Integr. Circuits Syst, 15, 8, 1996, 943–951.

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

Sherwani N, Algorithms for VLSI Physical Design Automation, Kluwer Academic Publishers, Boston, Mass, USA, 1999.

Shi Y and Eberhart R, A modified particle swarm optimizer, in Proceedings of IEEE World Congress on Computational Intelligence, 1998, 69–73.

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

Tung-Chieh Chen, and Yao-Wen Chang, Modern floor planning based on fast simulated annealing, Proceedings of the 2005 international symposium on Physical design, San Francisco, California, USA, 2005.

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