IJCTA
Vol.8, No.1, Jan-June 2015, Pp.221-236
© International Sciences Press, India

Hybrid PS-ACO Algorithm in Achieving Multiobjective Optimization for VLSI Partitioning

Atul Prakash[1] and Dr. R. K. Lal[2]
[1]Dept. of Electronics & Communication Engg, Birla Instiitute of Technology, Mesra, Ranchi, India
[2]Asso. Professor, Dept. of Electronics & Communication Engg, Birla Instiitute of Technology, Mesra, Ranchi, India
Email:-prakashatul.uit@gmail.com; rklal@bitmesra.ac.in;

ABSTRACT

In this paper multiobjective optimization problem simultaneously optimized using hybrid PS-ACO algorithm has been attempted. The methodology used in this paper is based upon the information sharing and movement of swarms or particles in a search space, and further applying ACO on the result obtained by the PSO. Multiobjective optimization problems are present at physical design level at partitioning process of VLSI circuit optimization. Here present the results of multiobjective optimization of cutsize, delay and sleep time simultaneously using hybrid swarm technique (PS-ACO). Results in this paper shows that the NP hard problem effectively solved by PS-ACO algorithm. Here set up the problem as a simultaneously multiobjective optimization and solve it by programming method. Information of the circuit has been given in accordance with circuit netlist files used in ISPD'98 circuit benchmark suite. The proposed approach has a good potential in VLSI circuit partitioning.

Index Terms— PSO, ACO, Cutsize, Sleep time, NP-hard.

1. INTRODUCTION

Modern VLSI chips contain millions of transistors. This has become possible by the development of sophisticated design tools, software's and highly scaled VLSI fabrication techniques. To deal with such a huge chip complexity and achieve a short turnaround time, VLSI design tools must not only be computationally fast but also generate optimal designs. Partitioning is an important step in physical design of circuits. In order to build complex digital logic circuits it is often essential to subdivide multi-million transistor circuit designs into manageable pieces. So, Partitioning, on the one hand, is a design task to break a large system into pieces to be implemented on separate interacting components and on the other hand it serves as an algorithmic method to solve difficult and complex combinatorial optimization problems as in logic or layout synthesis. Partitioning if done in a proper way can solve many design issues and simultaneously can reduce the delay, overall size of the circuit, the number of cutsize (or number of connections between two partitions) and can also lead the design process to design a low power consuming circuits.

NP-hard problem of partitioning cannot be effectively solved by deterministic method. In this paper a heuristic iterative approach to solve the partitioning problem and simultaneously optimization of power, delay and cutsize is presented.

Kernighan and Lin [1] proposed the first heuristic search algorithm for bipartitioning with several times randomly generated initial partitions and to obtained the best solution based on swapping of vertices. Modified version [2] of [1] leading to a fast linear time algorithm for partitioning and improved time complexity but inefficient time complexity was improved. Krishnamurthy [3] by modified [2] introduce the concept of look ahead to choose the cell move. Multiway partitioning problem was effectively solved by recursive bipartitioning and improved time complexity [3].

Optimize the power and delay in VLSI with optimal sizing the transistor in a digital MOS VLSI circuit were proposed [4]. To perform multiway partitioning modified [3] by developing appropriate data structure and proved that the optimal number of gain levels necessary depends on the number of blocks to be partitioned net size and degree of distribution to the circuit network, but not on the size of the network [5]. One new methodology transforms the circuit optimization into multivariable optimization problem was shown to provide an optimum design with circuit analysis accuracy [6].

Kennedy and Eberhart [7] introduced a concept for the optimization of nonlinear function using particle swarm methodology, PSO is a robust stochastic optimization technique based on information sharing and movement of swarms. Another heuristic technique geometric partitioning were proposed [8] for partitioning of a system to maximize exploitable sleep time for low power synthesis with deactivate the memory refresh circuitry ,apply power down or disable clock signal during the inactive periods of operation of circuit elements and thus maximize the power consumption.

Various hypergraph partitioning algorithms were proposed based on successively hypergraph and fixed vertices which reduces the complexity of partitioning problem [10] [11]. A multiobjective h-metis partitioning were proposed [12] for simultaneously cutsize and circuit delay minimization using memetic algorithm for VLSI physical design. Various optimization algorithms were proposed for power and delay minimization [13] [14].

Ghafari et al. [15] focused on minimizing the average power consumption in CMOS circuits. A discrete PSO algorithm was proposed [16] for the optimization of VLSI interconnections (netlist) bipartitioning and giving good result as compared to GA. Gill et al. [17] proposed multiway circuit partitioning using genetic algorithm with objectives of mincut, ratio cut minimization and shows good result. Shanavas et al. [18] discussed the memetic algorithm was used to optimize the various objective functions.

The different objective function that may satisfied by partitioning are:

1. Maximization of sleep time due to partitioning.

2. Minimization of number of cuts.
3. Minimization of delay due to partitioning.
4. The percentage of saving power should be larger than the percentage of consumed power during switching activities.
5. To reduce the fabrication cost with minimum area or as a balance constraints.
6. The number of terminals should not exceed the terminals available on PCB.

From the literature review it is found that the various researchers have applied various optimization techniques for the partitioning optimization problems with mixed results. In the present work excellent optimization method of Particle Swarm Optimization has been applied to partitioning optimization problems.

2. PROBLEM FOMULATION

With the advancements in VLSI technology the chip complexity is increasing, leading to more and more integration and increased design sizes. A huge chip estate is being occupied by interconnects, which leads to increased delay. Improved physical design tools, are necessary to handle these issues. Circuit partitioning plays an important role in physical design automation of very large scale integration (VLSI) chips. In VLSI circuit partitioning, the problem of obtaining a minimum cut is of prime importance. To enhance, other criterion like power, delay and area in addition to minimum cut is included. Circuit net list partitioning is an important step in VLSI physical design and involves the division of a circuit into smaller parts for ease of design and layout. The main objectives of circuit net list partitioning include minimization of number of interconnections between the partitions, minimization of delay due to interconnections between partitions & ratio-cut minimization and minimization of power consumption by maximizing the total sleep time of different partitions. Present work demonstrates the versatility of PSO for bi-partitioning to minimize the Interconnections also called cuts, delay and maximizing sleep time.

Circuit partitioning problem is a non polynomial hard problem cannot be effectively solved by deterministic methods. PSO is a stochastic algorithm can be used effectively for circuit partitioning. In this paper a heuristic approach is presented to optimize the three design issues the cutsize, delay and sleep time. In this paper all the experiments have been done on MATLAB R2010.

1. Mincut minimization

The numbers of interconnections among partitions have to be minimized. Reducing the interconnections not only reduces the delay but also reduces the interface between the partitions making it easier for independent design and fabrication. It is also called the mincut problem or minimization of the number of cuts.

The problem involves dividing the circuit net list into two subsets and some of the connections are also cut. The number of cut belonging in two different partitions is the cost of a partition, and this cost can be defined as follows

$$C = \sum_{i=1}^{k} \sum_{j=1}^{k} C_{ij} \ , (i \neq j) \tag{1}$$

where i , j are the vertices (nodes) of an edge (net)

C = cost of cut

C_{ij} = cost of an edge

The partitioning problem is to partition $'V'$ into $V_1, V_2, \ldots \ldots V_n$

Where $V_i \cap V_j = \emptyset$

$U_{i=1}^{n} V_i = V$

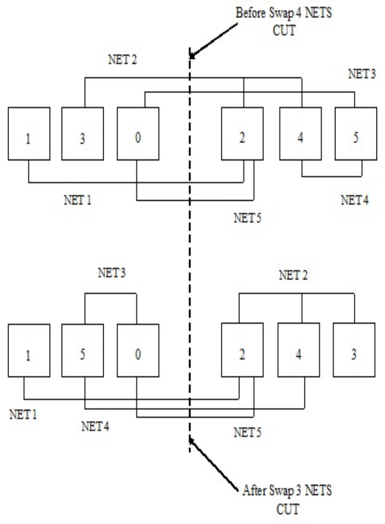

Figure 1 Circuit partitioning overview [Coe et al. (2004)]

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

$$\sum_{i=0}^{k} m_i = \sum_{j=0}^{k} n_j \tag{2}$$

Where, m_i & n_j are the nodes in two partitions [19].

2. *Delay Minimization*

The partitioning of a circuit might cause a critical path to go in between partitions a number of times. As the delay between partitions is significantly larger than the delay within the partition, it is an important consideration in circuit partitioning. Important considerations for partitioning constraints include minimization of delay due to partitioning.

First of all, the critical paths between the input/output ports (pads) are checked. The critical path is defined as the path having maximum delay between the I/O pads.

$$Delay = \max_{p_i \in P} \big(H(p_i)\big) \tag{3}$$

Where $H(p_i)$=No. of times a hyper path, p_i is cut [19].

To calculate this delay we use the well-known Elmore Delay model. Our delay model has two components. The first component is the gate delay. For all gates we consider a typical intrinsic delay that is given for a typical input transition and a typical output net capacitance. The second component is the wire delay, which we approximate using the Elmore delay model. The Elmore delay for an edge *e* (an edge corresponds to the wire connecting the net source to one of its fanout sinks) is given by [12]:

$$Delay(e) = R_e \left(\frac{C_e}{2} + C_t\right) \tag{4}$$

$$R_e = L_{avg} + r_e \tag{5}$$

$$C_e = L_{avg} + c_e \tag{6}$$

where *Re* is the wire lumped resistance, C_e is the wire lumped capacitance, and C_t is the total lumped capacitance of the source node of each net, which is taken as zero [20]. To compute R_e and C_e we need the length of each edge. For that, we use the statistical net-length estimation method, also known as MRST (Minimum Rectilinear Steiner Tree) model. According to this method the average length of a net, connecting *m* cells enclosed in a rectangular area with width *a* and height *b*, is given by:

$$L_{avg} = (\alpha . m^{\gamma} - \beta)\frac{a.b}{a+b} + (a + b) \tag{7}$$

where α, β, and γ are fitting parameters computed as $\alpha \approx 1.1$, $\beta \approx 2.0$, and $\gamma \approx 0.5$, m is the number of nets, a and b are the net bounding area dimensions. During recursive partitioning, when a net is cut, it is assigned a certain wire delay that will be used to re-compute all delays on the paths that include that net. The higher the level in which a net is cut during recursive partitioning, the greater the back-annotated wire delay has to be. In our case, any net that is cut during the first bi-partitioning step is assumed to be bounded by a rectangular area which is the same as the chip area and for simplicity we consider an aspect ratio equal to 1.The delay of each net is set only the first time when it is cut. In our experiments we consider a 0.18μ copper process technology (unit length resistance r_e= 0.115, unit length capacitance c_e= 0.00015) [21].

3. *Sleep Maximization*

The idea about the low power consuming circuit partitioning is that for a given period of time if all the elements in the particular partition is idle then we can send that partition into sleep mode so that the power is save during the time interval. The idea is exploited in the Figure 1.It can be notice that higher discrete overlapping of idle time means greater number of switching and a more complicated control circuitry. Hence the gain function *G(S₁, S₂)* should be an increasing function of t_i and a decreasing function of sw_i. For a bi-partitioning problem the gain function that needs to be maximized is defined as:

$$f = t_1 + t_2 - \beta(sw_1 + sw_2) \qquad (8)$$

$where\ t_{i\,,} is\ the\ sleep\ time\ for\ i^{th}\ partition\ and$
$sw_i\ is\ the\ switching\ activity\ of\ the\ i^{th}\ partition$

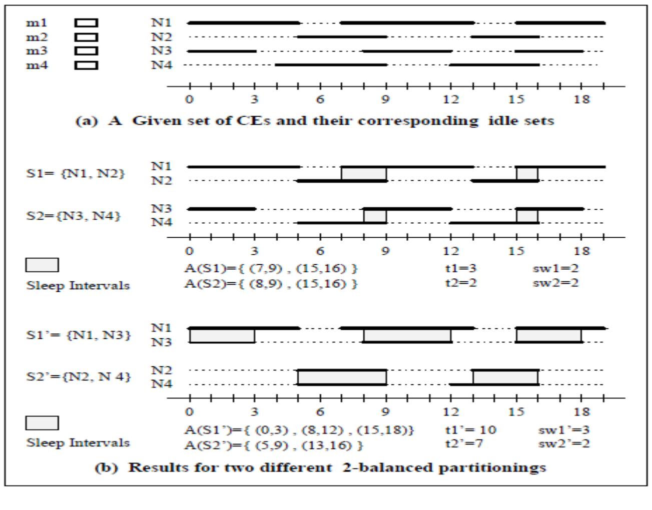
m1
m2
m3
m4
N1
N2
N3
N4
0 3 6 9 12 15 18
(a) A Given set of CEs and their corresponding idle sets
S1= {N1, N2}
S2={N3, N4}
Sleep Intervals
A(S1)={ (7,9) , (15,16) }
A(S2)={ (8,9) , (15,16) }
t1=3
t2=2
sw1=2
sw2=2
S1'= {N1, N3}
S2'={N2, N 4}
A(S1')={ (0,3) , (8,12) , (15,18)}
A(S2')={ (5,9) , (13,16) }
t1'= 10
t2'=7
sw1'=3
sw2'=2
(b) Results for two different 2-balanced partitionings

Figure 2 Partitioning to maximize sleep time [8]

In equation (8), summation of $ti's$ accounts for the savings in power consumption due to sleep mode operation the partitions and β *(sw₁ + sw₂)* accounts for overhead in power consumption due to extra control circuitry. Parameter 'β' controls relative significance of power savings (*ti*) and the overhead terms (*swi*) and depends on the available technology and type of circuitry in modules *m*.

If $P \text{ and } P'$ be the power consumption with and without using sleep mode then

$$P = \frac{[P_0\,(T-t_1)+P_s t_1]+[P_0(T-t_2)+P_s t_2]}{T} = \frac{P_0[2T-(t_1+t_2)]+P_s(t_1+t_2)}{T}$$

and $$P' = \frac{[(P_0 T)+(P_0 T)]}{T} = 2P_0 \tag{9}$$

Where $P_0 \text{ and } P_s$ be the power consumption of each partition in operating and sleep mode and T is the operation time.

Then the percentage of power saving can be given by [8],

$$S = \frac{P' - P}{P'} \times 100$$

For a given memory chip typically we have $\frac{P_0}{P_s} > 25$, [8] therefore the percentage of power consumption would be atleast:

$$S_{min} = 48 \times \frac{t_1+t_2}{T} \tag{10}$$

The combined objective function used to optimization of the above stated quantities is taken as,

$$C_c = max\left[\left(\gamma_s\left(\sum_{i=0}^{p} t_i - \beta \sum_{i=0}^{p} sw_i\right) + \gamma_c\left(\sum_{i=0}^{k}\ \sum_{j=0}^{k} y_{ij}(i \neq j)\right)\right) \times \frac{1}{R_e\left(\frac{C_e}{2}+C_t\right)}\right] \tag{11}$$

Where $\gamma_s \text{and } \gamma_c$ are the weight given to the sleep time and mincut, y_{ij} is the inverse of C_{ij} .

3. SOLUTION METHODOLOGY

In the present work, a hybrid PS-ACO algorithm for optimization of multimodal continuous functions is proposed.

PSO is based on the intelligence, which can be applied into both scientific research and engineering use. It has no overlapping and mutation calculation. The search in this technique is carried out by the speed of the particle. During the development of several generations, only the most optimist particle can transmit information onto the other particles, and the speed of the researching is very fast. The algorithm is comparatively easy than the other intelligence based search algorithm as it adopts the real number code, and it is decided directly by the solution. The number of the dimension is equal to the constant of the solution.

The algorithm easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction. Again it cannot be work out for the problem having scattering optimization, non-coordinate system such as the solution to the energy field and the moving rules of the particles in the energy field

The demerits which have been possessed by the PSO can easily be applying the ACO algorithm as it has Inherent parallelism and positive Feedback accounts for rapid discovery of good solutions. Furthermore it can be used in dynamic applications (adapts to changes such as new distances, etc). Applying ACO can lead to some more difficulties such as it has random decision making ability which is not independent. Again the probability distribution changes by iteration.

So application of PSO and ACO consecutively, leads to faster convergence and it secures better solution in respect to the previously proposed algorithms.

In this, PSO is applied for global optimization by updating positions of particles to attain rapid convergence. PSO simulates the behavior of bird flocking. One of the advantages of PSO is that PSO take real numbers as particles. It is not like GA, which needs to change to binary encoding, or special genetic operators have to be used. The searching is a repeat process, and the stop criteria are that the maximum iteration number is reached or the minimum error condition is satisfied.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In each iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness), it has achieved so far. The fitness value is also stored. This value is called pbest. Another "best" value is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is the global best and called $gbest$. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called $lbest$.

After finding the two best values, the particle updates its velocity and positions with the following equations (12) and (13)

$$v[\,] = w \times v[\,] + c_1 \times rand_1(\,) \times (pbest[\,] - present[\,]) + c_2 \times rand_2(\,) \times (gbest[\,] - present[\,]) \quad (12)$$

$$present[\,] = present[\,] + v[\,] \quad (13)$$

$v[\,]$ is the particle velocity, w is inertia weight, $present[\,]$ is the current particle (solution). $pbest[\,]$ and $gbest[\,]$ are defined as stated before. $rand_1(\,)$ and $rand_2(\,)$ are random numbers between 0 and 1. c_1 *and* c_2 are learning factors. Usually, $c_1 = c_2 = 2$.

Particles' velocities on each dimension are clamped to a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed V_{max} (where Vmax is a parameter specified by user), then the velocity on that dimension is limited to V_{max}. The basic pseudo code for PSO is shown in Figure 3.

After applying PSO to calculate the *pbest* for each particle the ACO is applied taking the *pbest* for each particle as the initial value of the ants. To calculate the *gbest* value the pheromone is initialized.

ACO works in three steps:

1. Construct Ant Solutions(which has been found using PSO)
2. Daemon action and
3. Updating of Daemon

An Ant will move from node i to j with probability

$$P_{i,j} = \frac{\left(\tau_{i,j}^{\alpha}\right)(\eta_{i,j}^{\beta})}{\Sigma\left(\tau_{i,j}^{\alpha}\right)(\eta_{i,j}^{\beta}} \tag{14}$$

Where, $\tau_{i,j}^{\alpha}$ *is the amount of pheromonr on edge* i,j

$\eta_{i,j}^{\beta}$ *is the desirability of edge* i,j $\left(typically\, \frac{1}{d_{i,j}}\right)$

α *is a parameter to control the influence of* $\tau_{i,j}$

β*is a parameter to control the influence of* $\eta_{i,j}$

α *and* β *has been taken eaqul to* 0.5 *for this experiment*

Amount of pheromone is updated according to the equation

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j} \tag{15}$$

Where, ρ *is the rate of pheromone evaporation*

$\Delta\tau_{i,j}$ *is the amount of pheromone deposited, typically given by*

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{1}{L_k}\ if\ ant\ k\ travels\ on\ edge\ i,j \\ \quad 0,\ otherwise \end{cases},$$

Where, L_k*is the cost of the* k^{th} *ant's tour* (*typically length*)

Pheromone values are updated by all the ants that have completed the tour

$$\tau_{i,j} \leftarrow (1-\rho) + \textstyle\sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{16}$$

Where m is the number of ants.

The proposed algorithm is as follows:

STEP 1. Start at the beginning of netlist and convert it into matrix form.

STEP 2. Bipartition the circuit into 0 and 1 partitions as

$$\sum_{i=0}^{L} li = \sum_{i=o}^{k} m_i + \sum_{j=o}^{k} n_j \tag{17}$$

Also, from (2),

```
For each particle
Initialize particle
End
Do
For each particle
Calculate fitness value
If the fitness value is better than the best fitness value ( $pbest$ ) in history
Set current value as the new $pbest$
End
Choose the particle with the best fitness value of all the particles as the $gbest$
For each particle
Calculate particle velocity according to equation (a)
Update particle position according to equation (b)
End
While maximum iterations or minimum error criteria is not attained.
```

Figure 3 The basic pseudo code for PSO

$$\underset{\{0\,\text{partition}\}}{\sum_{i=0}^{k} m_i} = \underset{\{1\,\text{partition}\}}{\sum_{j=o}^{k} n_j}$$

where $l_i = m_i + n_j$

STEP 3. Initialize the particles′ positions using some random value and divide this random value with the length of either partition generated earlier. Use this calculated random value to swap the node between the two previous partitions to create new partitions.

$$Initialized\ Particles = \sum_{i=0}^{L} P_i\ ,$$

P_i Represents the PSO particles where i varies from 0 to L (any user defined value)

STEP 4. Create any number of particles using this concept. Calculate their mincut using

$$Interconnects\ between\ partitions\ (C_{ij}) = \textstyle\sum_{i=0}^{k} m_i \left(\sum_{j=0}^{k} n_j\right) \tag{18}$$

And delay using the equation (4). Evaluate the sleep time S of the two partitions form equation (8). Correspondingly evaluate fitness function, C_{cij} , for all the particles using (11), taking preference vectors $Y_s = Y_c = 0.5$, (for 50% weight to sleep and mincut objectives).

STEP 5. Take the maximum of these C_{cij} values as '*gbest*' and all other values as individual '*pbest*' for all the particles.

STEP 6. Initialize the velocity for all the particles.

Initial velocity $= v_{id}$, where i varies from 0 to k

STEP 7. Assume the initial positions of all the particles as their "*pbest*"
$Initial\ Position = x_{id}\ , where\ i\ varies\ from\ 0\ to\ k$

STEP 8. Using PSO equations, particle velocity, v_{id} and particle position, χ_{id} are evaluated. This updated particle position is used further to create new particle using the same concept as discussed in STEP 3.

STEP 9. Apply PSO equations L number of times.

STEP 10. Calculate number of interconnections, delay, sleep time and combined fitness function for all the particles using (15), (4),(8) and (11) respectively.

STEP 11. Assume that the new combined fitness function = $C_{cij}{}'$

Compare $C_{cij}{}'$ with C_{cij} for each particle.

If $C_{cij}{}' > C_{cij}$, then accept $C_{cij}{}'$

If $C_{cij} > C_{cij}{}'$, then C_{cij} will remain as it is.

STEP 12. Find particle P_i , whose fitness function is maximum and take the position of this particle as "*gbest*".

STEP 13. Calculate new positions for all the particles with their current "*pbest*" and "*gbest*" (the position of the particle having maximum fitness value) using equations 12 & 13..

STEP 15. Apply the above mentioned ACO algorithm for each particle.

STEP 14. If the termination condition is true (i.e. no. of iterations reach a threshold value set by user), then END of the program, otherwise go to STEP 8.

4. RESULTS AND DISCUSSION

As the objective of the paper is to optimize the interconnections, the delay and the sleep time between two partitions (bi-partitioning), all the parameters have been optimized simultaneously in this work. First of all, the interconnections, delay and sleep time were calculated before applying the proposed PS-ACO approach; then the proposed approach was applied to optimize all the parameters simultaneously with 50% weight to mincut and the sleep. The coding was done using MATLAB R2010. A number of netlists consisting of 10-25 nodes were used for this purpose. PS-ACO algorithm has been applied to all the aforesaid netlists; the results of few (showing the best results) are shown in the Table 1. From Table 1, it is seen that the proposed approach has been exhaustively tested and results have been obtained on circuit netlist files of varying size ranges. The proposed approach performs better overall size ranges. The computational time for calculation of the sleep time is taken as 100 clock period.

The interconnection, delay and sleep time when optimized simultaneously as a multi-objective fitness function, with 50% weight to sleep and mincut, the two quantity mincut and delay try to minimize and the sleep time try to be maximize and finally become stagnant as the number of iterations goes on. As shown in Figure 3 the interconnections minimize, delay minimizes and the sleep time maximizes as the iteration goes on, and finally become constant. It means that there is no further scope of optimization and are the ultimate results of interconnections, delay and sleep time.

5. CONCLUSION

VLSI circuit bi-partitioning using Multi-objective PS-ACO Algorithm have been proposed for mincut and circuit delay minimization along with maximization of sleep time. The advantages of the proposed PS-ACO approach are:

(1) It is fast, thus applicable to large-sized circuits.

(2) It performs better suitable partitioning, as it optimizes all the parameters with some cutsize, delay and sleep time trade-off. The proposed approach is tested on various circuit partitioning instances (netlists) given in ISPD'98 Benchmark Suite.

The Particle Swarm Optimization algorithm applied to VLSI partitioning produces a very good result of these three objectives simultaneously .In this paper the sleep time maximization along with minimization of cutsize and delay were explored. This triple objective function was separately formulated and then combined into one objective function. The combined problem is NP hard, hence heuristic approach was successfully introduced. There is an average improvement of 38 percent in cutsize, 65.67 percent in delay and in sleep time 43.36 percentage improved simultaneously for the netlist series used in Table 1.

As compared to GA the better results shown by this proposed algorithm. The comparison of results obtained through the proposed algorithm is better than the [15] as objectives of sleep time and mincut and also one more objective delay find out in this paper and shows a very good improvement. Moreover, results obtained show the versatility of the proposed method in solving non-polynomial hard problem of circuit netlist partitioning. It is proved that PSO approach is an excellent method of global search to achieve better solutions.

6. FUTURE SCOPE

There are many ways to extend the proposed work. The delay optimized is the net based delay. The same approach can be used to optimize path based delay. After finding path delay, combined net and path based delay can be calculated by giving weights to each delay. Then, the proposed PS-ACO approach can be applied for mincut and combined delay minimization with sleep time maximization. This PS-ACO is used to solve two-way circuit partitioning problem. The results can be improved by combining it with other evolutionary algorithms to make hybrid PS-ACO. The algorithm can also be improved by multi-way partitioning techniques and multi-point crossover with different selection methodologies. The efficiency of the proposed algorithm can be compared with other standard algorithms by solving the same problem and a comparative study can be done. It may be possible to enhance the efficiency of the proposed hybrid algorithm by hybridizing it with GA (Genetic Algorithm) or SA (Simulated Annealing). It may be a good idea to hybridize PSO and GA as PSO adopts the real number code and it is decided directly by the solution[22], while GA adopts the binary code to solve the problem.

REFERENCES

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example: [1]. Where appropriate, include the name(s) of editors of referenced books. The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in "[3]"—do not use "Ref. [3]" or "reference [3]". Do not use reference citations as nouns of a sentence (e.g., not: "as the writer explains in [1]").

Unless there are six authors or more give all authors' names and do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] B.W Kerninghan. and S. Lin, "*An efficient heuristic procedure for partitioning graphs*", Bell System Tech. Journal, Vol.49, pp.291-307,Feb,1970.

[2] C.M. Fidduccia and R.M. Mattheyes, "*A linear time heuristic for improving network partitions*", Proceeding of the 19th ACM/IEEE Design Automation Conference, IEEE Press, Piscataway, NJ, USA, 1982, pp.175-182.

[3] B. Krishnamurthy, "An *improved min-cut algorithm for partitioning VLSI circuits*", IEEE Transactions on Computers, Vol. c-33, Issue: 5, pp.438-446, May, 1984.

[4] Lance A. Glasser and Lennox P. J. Hoyte, "Delay and power optimization in VLSI circuits" Proc. of 21st IEEE Design Automation Conference, 1984, pp. 529-535.

[5] L.A. Sanchis, "*Multiple way network partitioning*", IEEE Transactions on Computers, Vol.38, No. 1,1989, pp.62-81.

[6] Y. Hsieh, K. Chin., C. Te Chuang, "Power Partition and Emitter Size Optimization for Bipolar ECL Circuit" in IEEE J. of solid-state circuits, Vol. 28, no. 5, pp. 548-552 ,May 1993.

[7] J. Kennedy and R. Eberhart ,"*Particle Swarm Optimization*", Proceeding of IEEE International Conference on Neural Networks ,perth , Austrailia Vol. 4,pp.1942-1948, 1995.

[8] Farrahi A. H. and Sarrafzadeh M., "*Geo_Part: A system partitioning to maximize sleep time",* Technical Report, Department of Electrical and Computer Engineering, Northwestern Univ., Evanston, 1995.

[9] Farrahi A. H. and Sarrafzadeh M., "*System partitioning to maximize sleep time,*" in *Proc. the 1995 IEEE/ACM Int. Conf. on Computer Aided Design*, San Jose, California, USA, 12-16 Jun. 1995, pp.242-247.

[10] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "*Multilevel hypergraph partitioning: Application in the VLSI domain,*" in *Proc. the 34th Annual Design Automation Conf.*, Anaheim, California, USA, 9-13 Jun. 1997, pp. 526-529.

[11] E. Caldwell, B. Kahng and L. Markov, "Hypergraph partitioning with fixed vertices," in *Proc. the 36th Annual ACM/IEEE Design Automation Conf.*, New Orleans, Louisiana, USA, 21-25 Jun. 1999, Article No. 21.3, pp. 355-359.

[12] C. Ababei, N. Selvakkumaran, K. Bazargan, G. Karypis, "*Multi-objective Circuit Partitioning for Cutsize and Path-Based Delay Minimization*", Proceeding of 2002 IEEE/ACM international conference on computer-aided design,Vol.8 No.6, pp.181-185.

[13] S. Sait, A. Maleh, and R. Abaji, "Enhancing performance of iterative heuristics for VLSI netlist partitioning," in *Proc. the 10th IEEE International Conference on Electronics, Circuits and Systems*, 2003, pp. 507-510.

[14] M. E. Ekpanyapong and S. K. Lim, "Simultaneous delay and power optimization for multi-level partitioning and floorplanning with retiming," in *Proc. the Int. Symp. on Circuits and Systems (ISCAS 2004),* Vancouver, Canada, 23-26 May 2003,pp.1-9.

[15] P. Ghafari, E. Mirhadi, M. Anis, S. Areibi, and M. Elmasry, "A low power partitioning methodology by maximizing sleep time and minimizing cut nets," in Proc. the Fifth Int. Workshop on System-on-Chip for Real-time-Applications, Banf , Alberta, Jul. 2005, pp. 368-371.

[16] S. Peng, G. Chen, W. Guo, "A Discrete PSO for Partitioning in VLSI Circuit" in Proceeding of 2009 , International conference on computational intelligence and software engineering, Wuhan, China, 2009, pp.1-4.

[17] S.S Gill., R. Chandel and A.K. Chandel (2010), "*Swarm Intelligence Based circuit partitioning*", Asia Pacific conference on Postgraduate Research in Microelectronics and Electronics, Shanghai, China.

[18] H. Shanavas, R.K. Gnanamurthy, T.S. Thangaraj, "A Novel Approach to find the best fit for VLSI Partitioning - Physical Design" in proc. International Conference on Advances in Recent Technologies in Communication and Computing. Artcom, 2010, pp. 330-332.

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

[20] J.J Cong. and K.S. Leung (1995), "*Optimal wiresizing under Elmore Delay model*", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol.14, No.3.

[21] P. Zarkesh, J.A Davis. and J.D. Meindl (2000), "*Prediction of Net-Length Distribution for Global Interconnects in a Heterogeneous System-on-a-Chip*", IEEE Transactions on VLSI Systems, Vol. 8, No. 6.

[22] Selvi V. and Umarani R. (2010), "Comparative Analysis of Ant Colonyand Particle Swarm Optimization Techniques" International Journal of Computer Applications (0975 – 8887) Volume 5– No.4, August 2010

Table 1 Mincut, delay and sleep time for different circuits using PS-ACO approach for partitioning

S.N.	File Name	Before optimization			After optimization			Percentage improvement			
		Min-cut	Delay (ps)	Sleep	Min-cut	Delay (ps)	Sleep	Min-cut	Delay	Sleep	Power
1.	Spp_N10_E7_R1_1025	8	5.7680	14	4	0.2367	38	50	95.89	171.42	18.24
2.	Spp_N10_E37_R1_3228	9	7.8305	21	8	5.7680	36	11.11	26.34	71.42	17.28
3.	Spp_N11_E12_R1_3386	7	3.9179	25	4	0.2367	36	42.85	93.95	44	17.28
4.	Spp_N20_E20_R1_1344	11	12.4628	38	6	2.3276	39	45.45	81.32	2.63	18.72
5.	Spp_N20_E20_R2_942	11	12.4628	33	6	2.3276	46	45.45	81.32	46	22.08
6.	Spp_N21_E18_R2_1659	18	32.1605	27	6	2.3276	42	66.67	92.76	55.55	20.16
7.	Spp_N22_E22_R2_1232	12	14.9863	35	9	7.8305	42	25	47.75	20	20.16
8.	Spp_N23_E27_R2_1796	20	38.4595	37	12	14.9863	38	40	61.03	2.7027	18.24
9.	Spp_N24_E25_R3_823	21	41.6913	21	9	7.8305	32	57.14	81.23	52.38	15.36
10.	Spp_N25_E87_R3_812	21	41.69	18	17	29.1026	39	19.05	30.19	116.67	18.72
	AVERAGE	13.8	21.1431	26.9	8.1	7.2974	38.8	40.27	69.17	58.28	18.62