IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

Genetic Algorithm Based Multiobjective Optimization for Very Large-Scale Integration (Vlsi) Circuit Partitioning

Sharadindu Roy

Department of Computer Science, Sonarpur Mahavidyalaya, Affiliated to University of Calcutta, Kolkata-700149, West Bengal, India

Available online at: www.ijcseonline.org

Accepted: 22/Jan/2019, Published: 31/Jan/2019

Abstract- A genetic algorithm based multi objective optimization technique for very large-scale integration (VLSI) circuit partitioning has been proposed. An efficient fitness function that simultaneously optimizes minimum net cut size and delay time and maximum sleep time has been worked out along with minimum power consumption. Use of bipartition has balanced the circuit perfectly. Circuit partitioning is a non-polynomial (NP) hard problem. I have used Genetic algorithm (GA)-based optimization as it shows a global optimum solution. This is a hyper graph - based solution. Since it is a part of a physical design, all the computational part including input-output (IO) pads are converted into a hyper graph. Genetic algorithm is an evolutionary optimization technique based on Darwinian Theory of natural selection. Fitness value has been evaluated and solution with low fitness value has been discarded. The method has been applied on the net list files used in ISPD’98 circuit benchmark suite where each file contained 20-30 nodes. MATLAB18a was used to code all the algorithms. The improvement of net cut size, delay and sleep time was 40.62%, 41.54% and 95.42% respectively compared to initial bipartition of circuit. Thus, the proposed methodology might be promising for current trends in VLSI circuit partitioning.

Keywords- Partitioning, Genetic Algorithm, NP-hard, Net list, Sleep time, Delay Crossover, Mutation, Cut size.

I. INTRODUCTION

Recently, there has been tremendous growth in the field of very large-scale integration (VLSI) design and automation and it is obvious there is no sign of saturation in this field and will be growing continuously. The improvement of technology makes life of human being easy, simple and affordable. But to design a VLSI circuit has become more complex and difficult to design and fabrication. There are so many issues to design complex component such as increased design time, increased delay, more area, more power consumption and infeasible design cost. The goal of partitioning is to break up large complex circuit component efficiently and logically into smaller interacting unit for easy and better handling. Various researchers and industries are continuously working to find better algorithm and methodology to break component (highly dense chip) into competent partition.

The first iterative improvement algorithm for partitioning a graph was proposed by Kernighan and Lin of bell telephone laboratories. They started with random partitions and tried to minimize the cut cost by swapping pairs of nodes [1]. This method suffered from local minima. Another algorithm proposed by Fiducca and Mattheyses was very fast and tended to cover the local minima [2]. In both the model a single point in the solution space was iteratively refined to obtain higher fitness value. But, these methods led to the locations of false peak in multi-modal search space [3]. The advantage of using GA based solution, is that here the search is done not in a single point. It exhibits higher degree of parallelism (large number of parallel points). Inspiring work by Goldberg explained the basics of GA and several other researchers used random crossover points over the chromosome to justify the ideas put forward by Goldberg [4-6].Various methods have already been proposed to improve selection of crossover boundaries. One such method showed comparison between individual chromosomes and the crossover operator so generated were based on the difference between the individual chromosomes [7]. Yuen and Chow emphasized on mutation operator and kept track of the chromosomes to avoid revisiting to reduce total run time [8]. An efficient method for partitioning of hardware and software which deals with improvement of running time and power of the system was proposed by Jigang and Srikanthan [9]. Most of these methods still lack to choose powerful or intelligent chromosome selection which hinders their time savings [10]. A remarkable study by Arato et al., proposed thinking of partitioning using both GA and Integer Linear Programming (ILP) [11]. He showed that GA was better than ILP in case of runtime.

Although, a lot of effort and time has been spent to drive a good quality partition, yet previous approaches have still not fulfilled all the design constraint. Circuit partitioning is a non-polynomial (NP) hard problem. Recent work by Prakash and Lal has proposed an important methodology for multi-objective VLSI circuit partitioning using Particle Swarm Optimization (PSO) [12].

Since GA is initially a discrete technique that is also suitable for combinatorial optimization problems over PSO, which is a continuous technique that is very poorly suited to combinatorial problems, a multi-objective Genetic algorithm (GA)-based solution for the improvement of partition quality of circuit is proposed which might prove to be efficient to fulfil the current trend in the design of VLSI. The first objective of my approach was to minimize the number of net cut size as I have considered the circuit as a hyper graph.

In addition, the number of inter-connections among partitioning must be minimized. Reducing the inter-connection not only reduce the delay but also reduce the interface between the partition making it easier for independent design and fabrication [13]. The second objective was to minimize delay due to partitioning. The partitioning of a circuit might cause a critical path to go in between partitions several times. As the delay between partitions is significantly larger than the delay within the partition, it is an important consideration in circuit partitioning. The third optimization issue was the concept of sleep maximization. Sleep mode refers to the mode in which there is no activity in part(s) of the circuit or the system. My objective was to maximize sleep time. Maximizing sleep time and minimizing power loss in transmission automatically reduces the power consumption of the whole system hence it was logically safe to power down the idle part(s) through specially designated control signal [14].

The algorithm can partition circuit into several sub-circuits. My method calculates the fitness value and discards solution with low fitness value. I have applied a multi objective genetic algorithm-based approach. Especially this approach was focused on to find an optimal solution that was able to optimize three objective functions.

These three functions are used to maximize sleep time, minimize delay and to minimize cut size respectively. Maximizing sleep time means to minimize power consumption of the whole circuit. Crossover boundary would be changed when fitness value was low in previous generation. MATLAB 2018a tool was used to code all algorithms which were not included here.

II. PROBLEM FORMULATION

2.1. Minimize net cut size

A circuit or a system in general consists of a set of modules connected by a set of nets. Each net should connect two or more modules together. We denoted m as number of modules and n as number of nets, hence:

-M Modules (Circuit elements): M= $\{m_1, m_2 \ldots m_m\}$

-N nets (signals): N= $\{n_1, n_2 \ldots n_n\}$

The goal was to partition the system into p block or partitions. The cut net minimization problem then consisted of finding a partition of p blocks ($p \geq 2$) such that the size of the cut set (total number of cut nets) was minimized or number of uncut nets was maximized. x_{ik} was defined as $x_{ik}=1$, if the module i(m_i) was in partition k (p_k) and $y_{jk}=0$ otherwise. y_{jk} was defined as $y_{jk} = 1$, if net j (n_j) was completely absorbed in partition k(p_k) and $y_{jk}=0$ otherwise. The objective was then to:

$$\text{Max}\sum_{j=1}^{n}\sum_{k=1}^{p} y_{jk}$$

Subject to:

Module Placement Constraint: $\sum_{k=1}^{p} x_{jk} = 1$

Net List Constraint: $y_{jk} \leq x_{ik}$, where $l \leq j \leq n$, $l \leq k \leq p$; i, j ∈ N

Constraints: $x_{ik} \in \{0,1\}$, $1 \leq i \leq n$; $1 \leq k \leq p$

$y_{jk} \in \{0,1\}$, $1 \leq j \leq n$; $1 \leq k \leq p$

The above maximization problem was a 0-1 linear integer programming problem which was shown to be NP-hard [14-17] and hence an interactive improvement algorithm was needed to be used.

This was a hyper graph that consisted of 7 module (4 cells and 3 Pads) and 5 nets. In this example net N2 and N3 are cut. So total number of net cuts =2. (See Fig. 1)

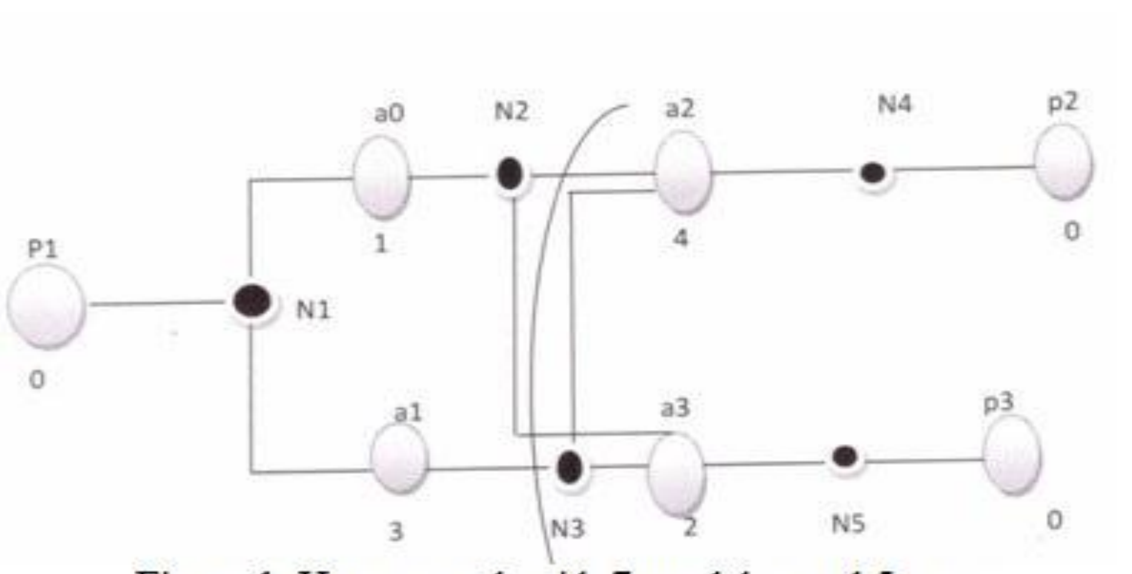

Figure 1: Hyper graph with 7 modules and 5 nets

2.2. Delay minimization

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

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

$$\text{Delay} = \underset{P_i \in P}{\text{Max}} (H(P_i))$$

Where H (P_i) = No. of times a hyper graph, Pi was cut [13].

To calculate this delay, the very well-known Elmore delay model was used. The delay model proposed here had two components. The first component was the gate delay. For all gates, a typical intrinsic delay was considered, that was given for a typical input transition and a typical output net capacitance. The second component was the wire delay, which was approximated 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 fan-out sinks) was given by [5].

$$\text{Delay (e)} = R_e (C_e/2 + C_t{*})$$
$$R_e = L_{avg} {*} r_e$$
$$C_e = L_{avg} {*} c_e$$

Where R_e was the wire lumped resistance; C_e, was the wire lumped capacitance, and C was the total lumped capacitance of the source node of each net which was taken as zero [16].

The length of each edge was needed to compute R_e and C_e. The statistical net length estimation method, also known as MRST (Minimum Rectilinear Steiner Tree) model was used for this purpose. According to this method, the average length of a net, connecting m cells enclosed in a rectangular area with width ‘a’ and height ‘b’, was given by:

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

Where α, β and γ are fitting parameters computed as α=1.1, β=2.0 and γ =0.5, m is the number of nets, a and b are the net bounding area dimension during recursive partitioning, when a net was cut, it was assigned a certain wire delay that was later used to recompute all delay on the paths that included those net. In this work, any net that was cut during the first bi-partitioning step was assumed to be bounded by rectangular area which was the same as the chip area and for simplicity I have considered an aspect ratio equal to 1.The delay of each net was set only for the first time when it was cut. In this experiment, I have considered a 0.18µ copper process technology (unit length resistance r_e=0.115, unit length capacitance C_e= 0.00015) [5, 17].

2.3. Maximization of Sleep time

Some basics definitions are given to formulate the problem for the maximization of sleep time. These definitions were based on the studies conducted in the recent past [14, 15]. The set M contains of m modules, when a module m was idle, then module m can be switched to sleep mode, during the time interval T=(r,e) if e<e. Given two intervals T_1=(r_1,e_1) and T_2=(r_2,e_2) a non-overlapping interval R was defined if $r_1 \geq e_2$ or $r_2 \geq e_1$. A non-overlapping intervals set (NIS) R_i for module m, was a set of interval during which module m could be set to sleep mode R_i ={T_{i1},T_{i2},…,T_{im}}.

S was the idle sets of all modules in M and was given as: S = {R_1, R_2 …R_m}. An empty interval T_1 was denoted by (). It was assumed that T_1covers T_2 if $r_1 \leq r_2 \leq e_2 \leq e_1$ or if T_2=(). The length of interval T, L (T)was defined as the intervals end point subtracted from the intervals starting point (e-r). Intersection of the two intervals T_1 and T_2 was denoted by $T_1 \cap T_2$ which was the longest interval that covered both T_1 and T_2. Intersection of two NIS, $R_1$and R_2 was defined as:

R_1= {T_{11}, T_{12}…T_{1n}} and R_2= {T_{21}, T_{22}……T_{2n}}

$R_1 \cap R_2$= {$T_1 \cap T_2$ |T_1 € R_1, T_2 € R_2, $T_1 \cap T_2 \neq$ ()}

Duration of NIS R = (T_1, T_2,...T_k} was defined as D(R)=∑L(T_1). Given S = {R_1, R_2,…R_m}. A(S) was defined as the intersection of all the NISs in S. {S_1, S_2,…S_p} was a p-partitioning of S if {S_1, S_2,…S_p} <S and $S_i \cap S_j$=□ and S_1υ, S_2υ,…υS_p was b-balanced if |S_1|≥b,|s_2|≥b,|S_P|≥b where |S| was cardinality of set S and equal to size of partition S. To define the objective function for maximizing the sleep time for p-partitioning problem, a gain G (S_1, S_2,…S_p) of a balanced p-partition was defined as follows:

G (S_1, S_2…S_p) =f (t_1, t_2…t_p, sw_1+sw_2+sw_3+…+sw_p) where t_1 and sw_1 were defined as:

t_1= D (A (S_i)) (sleep time of partition).

sw_1=|A (S_1) | (number of switching's of partition S_1).

It was noted that higher discrete overlapping of idle time meant greater number of switching and a more complicated control circuitry. Hence, the gain function G (S_1, S_2…S_p) should be an increasing function of t_1 and a decreasing function of sw_1. For a p-partitioning problem the gain function that needed to be maximized was defined as:

f= t_1+t_2+…+t_p, -β (sw_1+sw_2+sw_3+…+sw_p)

Parameter β controls relative significance of power savings (t_1) and the overhead terms (sw_1) and depended on the available technology and on circuitry in modules m. Fig. 2a, shows the activity profile of the modules discussed.

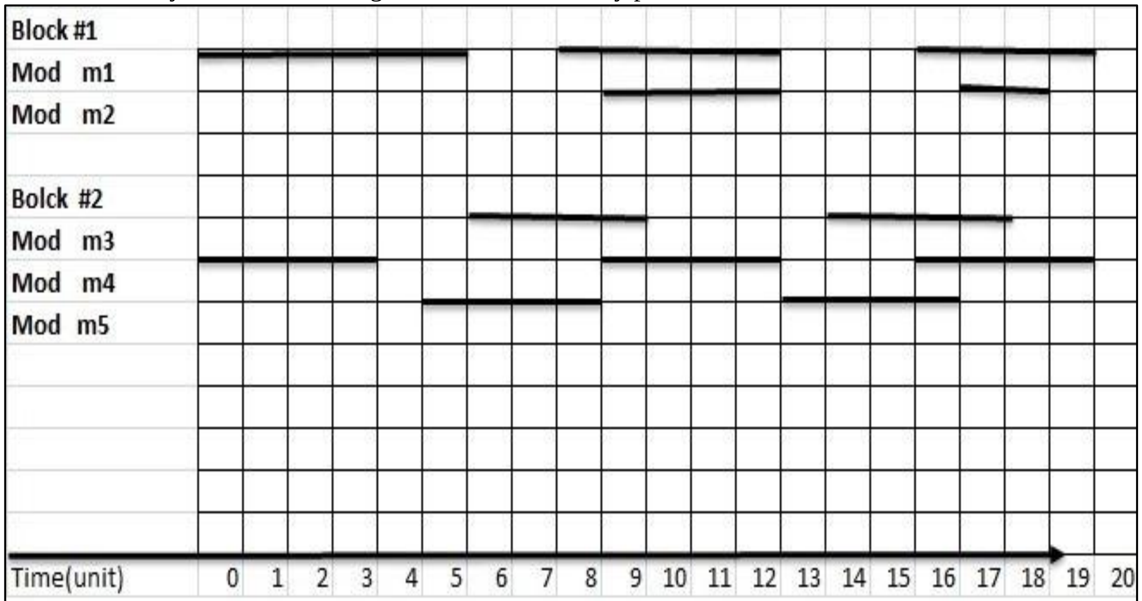

Figure 2.a : Activity profile of modules.

Activity profile of each module was shown with bold line in the Fig. Fig. 2b shows an example of overlap and switching time of two partitions block 1 and block 2. For 1st partition, overlapped time ={(8,12),(16,18)} So, t_1=6 switching time, sw_1=2; For 2nd partition, overlapped time={(8,9),(15,16)}, t_2=2 switching time, sw_2=2. Total sleep time = 8 as shown in the Fig. 2b.

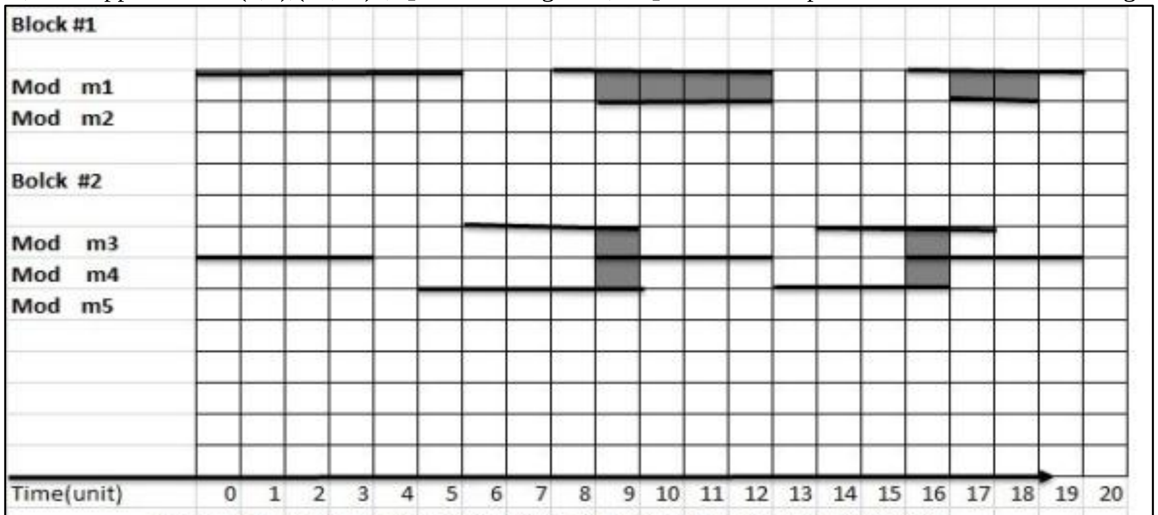

Figure 2.b: overlap and switching time of two partitions block1 and block2

Fig. 2c, shows another example of overlap and switching time of two partitions block 1 and block 2. For 1st partition, overlapped time ={(7,8),(15,16)}, So, t_1=2 switching time, sw_1=2. For 2nd partition, overlapped time ={(8,12),(15,17)},So t_2=6 switching time, sw_2=2; Total sleep time =8 as shown in the Fig. 2c.

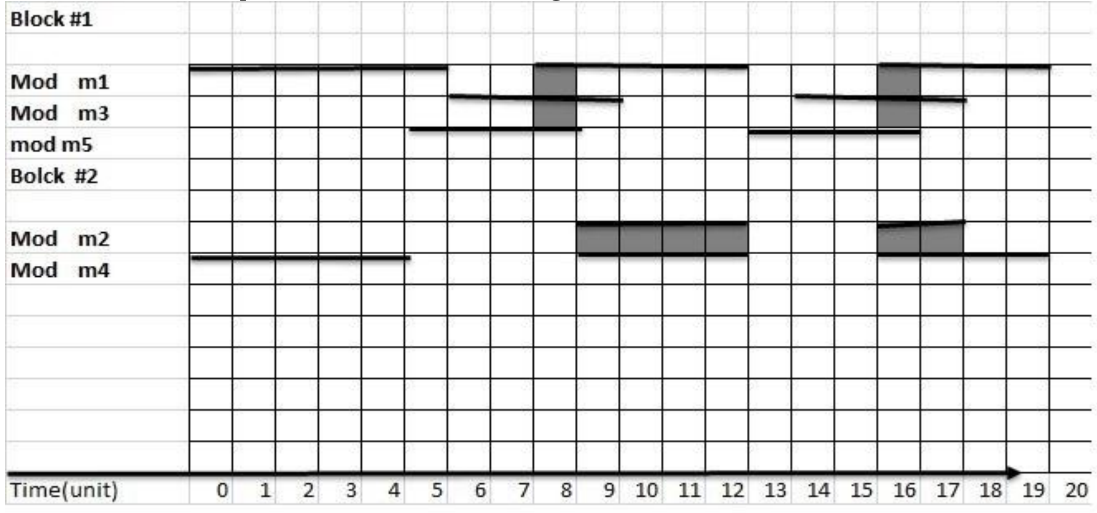

Fig. 2c. Overlap and switching time of two partitions.

If p_s and p_0 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

$$p' = \frac{[(p_0 T) + (p_0 T)]}{T} = 2p_0$$

where p_0 and p_s be the power consumption of each partition in operating and sleep and T was the operation time, then the percentage of power saving was given by [4, 18],

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

For a given memory chip typically, $\frac{Po}{Ps} > 25$, [18] therefore, the percentage of power consumption would be at least:

$S_{min}=48\times \frac{t1+t2}{T}$

2.4. Composite function

It has been formulated to obtain the triple goals and made the combined objective function as follows:

Thus 1st objective function y (1) = Max$\sum_{j=1}^{n}\sum_{k=1}^{p} y_{jk}$ (maximum uncut nets)

Thus 2nd objective function y (2) = R_e (C_e/2+C_t*) (minimizing delay)

Thus 3rd objective function y(3) = max (($t_1+t_2+\ldots+t_{p)}$, -β ($sw_1+sw_2+sw_3+\ldots+sw_p$)) (maximizing sleep time)

The combined function, y = max ((γ_c * y (1) + γ_s *y (2)) *1/y (3))

Where γ_c and γ_s were cut factor and sleep factor respectively and controlled the relative significance of cut nets versus sleep time in the objective function. The sum of cut factor and sleep factor was defined to be unity (γ_c+ γ_s=1). In this present work γ_c= γ_s= 0.5 was considered.

III. SOLUTION METHODOLOGY

In this present work, a multi-objective optimization technique using genetic algorithm was proposed. Solving my problem had taken prohibitive long time but the specific instance to solve was not known long before a decision must be made. In this case, I was able to input all parameters which were known long enough and treat the remaining parameters as objectives. Computing a possible large set of solutions upfront and by fixing all the remaining parameters, would help to search for a best possible solution from that set. The algorithm 1 was proposed to produce a connectivity matrix from circuit description in the ISPD 98 benchmark and then convert the matrix to a graph.

3.1. Algorithm 1

Algorithm 1 had been proposed for generating connectivity matrix from circuit description file (.NetD) and drawing a graph corresponding to the connectivity matrix.

```
Step 1:  Read circuit description file as a text file (.NetD) and convert it into a tabular form.
Step 2: Convert table into cell matrix (2D array)
Step 3: Create a connectivity matrix of module (adjacency matrix)
If a module $V_i$ has connection with module $V_j$ then
Connection matrix [i, j] =1,
Else
Connection matrix [i,j] =0
Step 4: List the source cell and connected cell from the circuit description.
Step5:  Create net matrix with m by n where m is the number of nets and n is the total number of modules.
Step 6: Make a graph with source module and connection module.
Step 7: End
```

3.2. Algorithm 2

Algorithm 2 had been proposed for a multi-objective optimization that would satisfy the triple objectives was as follows:

```
Step1: Start
Step2: Input netlist (.netD and .are file)
Step3: Form the connectivity matrix (adjacency matrix) of module and net matrix as the file was considered to be a hyper graph.
Step4: Bipartition the circuit (one was 0 partition and other was 1 partition)
Step5: Traverse the graph in BFS order and encode the sequence (chromosome) with partition number (0 and 1 only as it is bi partitioned)
Step6: Define selection, crossover, mutation type and their probability.
Step 7: Calculate 1st objective to minimize net cut i.e. to maximize uncut net by the formula as in the following.
  y (1) = Max$\sum_{j=1}^{n}\sum_{k=1}^{p} y_{jk}$.
Step 8: Calculate 2nd objective (y (2) = $R_e$ ($C_e$/2+$C_t$*)) to minimize delay.
8.1: Calculate area for the nets which are on the cut
8.2: Calculate average length of net wire by MRST (Minimum Rectilinear Steiner Tree).
```

$L_{average}$= (α.m^{γ} -β)$\frac{a.b}{a+b}$ + (a+b)

Where a and b were net bounding area dimension. M was the number of cells of the nets, α, and β were constants and taking the value of α=1, β =2.0 and γ= 0.5.

8.3: Calculate Lumped Resistance R_e and lumped capacitance (C_e) by the following formula.
$R_e = L_{average} * r$
$C_e = L_{average} * c$
(0.18 µ copper process technology was considered here, where unit length resistance (r) =0.115 and unit length capacitance (c) =0.00015).
8.4: Calculate delay using Elmore delay models
Delay(e)=R_e (C_e/2+C_t*)
Thus y (2) =R_e (C_e/2+C_t*)
Taking *C_t as the total lumped capacitance of the source model of each net which was take zero here.
Step 9: Calculate 3rd objective to Maximize sleep time:
Step 9.1: Define activity profile of each module with duration.
Step 9.2: Calculate overlapped time (t) and switching time (sw)
Step 9.3: Calculate sleep time as
y (3) =(t_1+t_2+…+$t_{p)}$,-β (sw_1+sw_2+sw_3+…+sw_p))
Where t_1, t_2…,tp were overlapped time of partitions and sw_1, sw_2…,sw_p were switching time .
β =0.1 was taken for this work.
Step10: Evaluate the multi-objective fitness (cost) function
y =max ((γ_c * y (1) + γ_s *y (2)) *1/y (3))
Step 11: Form the new population
Step 12: if the new optimized solution is better than the previous, accept it
Step13: if stopping criteria (optimal solution) is met go to step 14.
Else go to Step7.
Step14: Output the optimal solution
Step 15: End

IV. RESULT AND DISCUSSION

After analysing, the report was prepared on a net list in the ISPD 98 bench mark file format. The application was implemented using MATLAB 18a and was applied to several net list. .netD and .are file were taken as input for each net list. The first algorithm was applied on Spp-N90-E117-R6-433 which consisted of 92 nodes and 117 nets. The O/P was shown in Fig. 3, which shows the graphical representation of the Spp_N90_E117_R6_433netlist (ISPD 98 benchmark format.

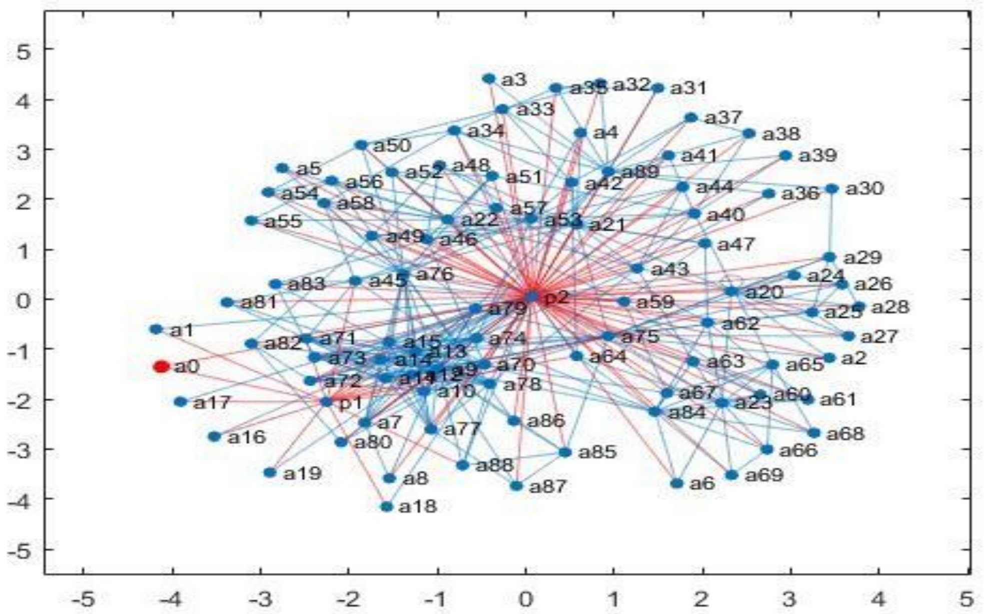

Figure 3: Graphical representation of the Spp_N90_E117_R6_433netlist (ISPD 98 benchmark format)

The second algorithm was used on the net list consisting of 20-30 nodes and 20-24 nets. All the results were reported by running the program on core-i3 (7th generation) machine with 4GB RAM. The procedure O/P starts with taking input file in .netD and .are format and converts it into hyper graph. After that the hyper graph has been partitioned into two equal blocks initially. As a multi-objective cost function I have used net cut, delay and sleep time between partitions that were optimized by this approach. All the values were calculated and reported initially (before the program run). After that the proposed approach was applied to optimize all the objectives simultaneously. The result was shown by giving 50% weight to minimum net cut and 50% weight to sleep time (that has to be maximized) as shown in Table 1. It was observed that the proposed approach was very much efficient as it was able to improve 40.62 % net cut, 41.54% delay and 95.42% sleeping time compared to initial bipartition of circuit.).

Table 1: **Result of objective function obtained before and after optimization along with percentage of improvement**

		Initial result after bipartition			Parameters		Optimized result			Improvement (%)		
Circuit description	No.of Net	Net cut	Delay	Sleep time	Net cut factor	Sleep factor	Net cut	Delay	Sleep time	Net cut	Delay	Sleep time
spp_N20_E20_R1_1344	20	**12**	**29384**	**12.7**	0.5	0.5	5	12477	18.2	58.33	57.54	43.31
spp_N20_E20_R2_942	20	**9**	**28937**	**4.5**	0.5	0.5	7	13215	12.7	22.22	54.33	182.22
spp_N20_E22_R2_659	22	**13**	**20866**	**8.2**	0.5	0.5	9	14315	18.4	30.77	31.40	124.39
spp_N20_E23_R2_857	23	**17**	**51546**	**9**	0.5	0.5	10	34562	16.4	41.18	32.95	82.22
spp_N20_E23_R2_1909	23	**12**	**117810**	**9**	0.5	0.5	8	81315	16.2	33.33	30.98	80.00
spp_N20_E23_R2_1878	23	**12**	**78672**	**8.3**	0.5	0.5	7	47887	13.6	41.67	39.13	63.86
spp_N20_E23_R2_1415	23	**11**	**47313**	**6.3**	0.5	0.5	5	33785	14.5	54.55	28.59	130.16
spp_N20_E22_R3_1063	22	**14**	**69222**	**10**	0.5	0.5	7	22735	16.6	50.00	67.16	66.00
spp_N20_E22_R2_3036	22	**16**	**68838**	**7.2**	0.5	0.5	9	44559	14	43.75	35.27	94.44
spp_N20_E24_R2_1014	24	**15**	**38894**	**9**	0.5	0.5	8	20208	13.6	46.67	48.04	51.11
spp_N20_E24_R2_1414	24	**13**	**57051**	**4.2**	0.5	0.5	9	37013	8.3	30.77	35.12	97.62
spp_N20_E24_R2_2178	24	**13**	**92557**	**6.6**	0.5	0.5	8	61998	13.5	38.46	33.02	104.55
spp_N20_E24_R2_1063	22	**11**	**82167**	**6.3**	0.5	0.5	7	43912	13.9	36.36	46.56	120.63
Average	**22.46**	**12.92**	**60250.54**	**7.79**	**0.50**	**0.50**	**7.62**	**35998.54**	**14.61**	**40.62**	**41.54**	**95.42**

. Conclusion

Multi-objective Genetic algorithm has been proposed that simultaneously optimized minimum net cut size, delay and maximum sleep time. Main philosophy was followed according to Holland [19]; objective functions were separately formulated and then combined into one objective function. The problem was NP hard. Several benchmark circuits have been optimized with the developed algorithm. Since every chromosome was coded in binary, the proposed method was very fast and efficient for those examples. Thus, from this study it can be concluded that the improvement in power consumption was very high i.e. power consumption has been minimized. It should be easily noted while the GA was inherently discrete, i.e. it encodes the design variable into bits of 0's and 1's, and therefore it was able to easily handle discrete design variables. PSO was inherently continuous and must be modified to handle discrete design variable. Further work using multi-objective genetic algorithm for multi-way partitioning of circuit with multipoint crossover is needed.

References

[1] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, *The Bell System Technical Journal*, 1970, 49: 291-372. **DOI:** 10.1002/j.1538-7305.1970.tb01770.x

[2] C.M. Fiducca, R.M. Mattheyses, S. Areibi, Mimetic algorithms for VLSI Physical Design: Implementation Issues. *Genetic and Evolutionary computation*, Proceedings of the 19th Design automation Conference, *IEEE Press*, 1992, 175-181.

[3] L.D.E. Goldberg, Genetic algorithms in search, optimization and machine learning. *Pearson Education*, 2004.

[4] S. Areibi, Mimetic algorithms for VLSI Physical Design: Implementation Issues. *Genetic and Evolutionary computation Conference, San Fransisco, California,* 2001, 140-145.

[5] C. Ababei, S. Navaratnasothie, K. Bazargan, G. Karypis, Multi-objective circuit partitioning for cut-size and path-base delay minimization. *IEEE International Conference on Computer aided Design*, 2002. **DOI:** 10.1109/ICCAD.2002.1167532

[6] M. Palesi, T. Givargis, Multi-objective design space exploration using genetic algorithms, *Proceedings of the 10th international symposium on Hardware/software codesign, ACM Press, Estes Park, Colorado,* 2002, 67-72. **DOI:** 10.1145/774789.774804

[7] Z.Q. Chen, Y.F. Yin, A new crossover operator for real-coded genetic algorithm with selecting breeding based on difference between individuals, *Natural Computation (ICNC), 2012 Eighth International Conference*, 2012, 644-648. **DOI:** 10.1109/ICNC.2012.6234556

[8] S.Y. Yuen, C.K. Chow, A genetic algorithm that adaptively mutates and never revisits, *Evolutionary Computation*, 2009, 13: 454-472. **DOI:** 10.1109/TEVC.2008.2003008

[9] W. Jigang, T. Srikanthan, Efficient algorithms for hardware/software partitioning to minimize hardware area, *Circuits and System*, 2006, IEEE Asia Pacific Conference, 1875-1878. **DOI:** 10.1109/APCCAS.2006.342205

[10] S.S. Gill, R. Chandel, A. Chandel, Genetic algorithm-based approach to circuit partitioning, *International Journal of Computer and Electrical Engineering*, 2010, 2: 1793-1863. **DOI**: 10.7763/IJCEE.2010.V2.136

[11] P. Arato, S. Juhasz, Z.A. Mann, D. Papp, Hardware-Software partitioning in embedded system design, International Conference on Complex, Intelligent and Software Intensive Systems, 2003, 197-202. DOI:10.1109/ISP.2003.1275838

[12] A. Prakkash, R.K. Lal, PSO: An approach to multi-objective VLSI Partitioning. *Proceedings of the 2nd International Conference on Innovations in Information, Embedded and Communication systems. IEEE*, 2015. **DOI:** 10.1109/ICIIECS.2015.7192971

[13] N. Sherwani, Algorithms for VLSI physical design automation. 3rd edition, Springer (India) Private limited, New Delhi, 2005.

[14] A.H. Farrahi, M. Sarrafzadeh, System partitioning to maximize sleep time, Proceedings of the 1995 IEEE/ACM International Conference on computer Aided Design, San Jose, California, USA, 1995, 452-455. **DOI:** 10.1109/ICCAD.1995.480155

[15] P. Ghafari, E. Mirhard, M. Anis, S. Areibi, M. Elmary, A low power partitioning methodology by maximizing sleep time and minimizing cut nets. *IWSOC, Bauf, Alberta, Canada*, 2005, 368-371. **DOI:** 10.1109/IWSOC.2005.15

[16] J.J. Cong, K.S. Leung, Optimal wire sizing under Elmore delay model, *IEEE Transactions on Computer Aided Design of Integrated Circuits and System*, 1995, 14: 3. DOI: 10.1109/DAC.1996.545625.

[17] P. Zarkesh, J.A. Davis, J.D. Meindail, Prediction of Net-Length Distribution for global interconnects, In a Heterogeneous system-on-a-chip. *IEEE Transaction on VlSA Systems*, 2000, 8: 6. **DOI:** 10.1109/92.902259

[18] K.P. Subbaraj, P. Sivasundari, S. Kumar, An effective mimetic algorithm for VLSI partitioning problem, *International conference on ICTES Chennai India*, 2007, 667-670.

[19] J.H. Holland, Adaption in natural and artificial system: An introductory analysis with applications to biology, control and artificial intelligence, *MIT press*, 1992.
ISBN:02620821

Authors Profile

Sharadindu Roy is working as an Assistant Professor and Head, in the Department of Computer science, Sonarpur Mahavidyalaya, Rajpur, South 24 Parganas, Kolkata, West Bengal, India. He received Post graduate degree in computer science from the University of Calcutta in the year of 2005.He is currently pursuing Ph.D degree in computer Science at the University of Calcutta, Kolkata,India. His area of research is to study the mechanism in the VLSI physical design and design some algorithms that resolve the problem of integrated circuit(IC) partitioning.He is also interested in the soft computing and optimization. He wants to study various multiobjective problem and implement using soft computing approach .