International Journal of Electrical & Electronics Research (IJEER)
Volume 4, Issue 3, Pages 98-101, September 2016, ISSN: 2347-470X

Comparison of HNN and Ga Based Hybrid Algorithm for Standard Cell Placement in VLSI Design

Dr. Aaquil Bunglowala[1], Dr. Nidhi Asthana[2]
[1]Asso. Dean and Professor,
Department of Electronics and Telecommunication,
MPSTME, NMIMS, Shirpur, Maharashtra, India; Mobile No.: +91942579258,
[2]Asst. Professor, Department of Engineering Mathematics,
SAIT, Indore, M.P., India; Mobile No.: +919691238294
[1]aaquilbun@gmail.com [2]drnidhi80@gmail.com

ABSTRACT- This research contracts with the notion of hybridization and reports use of hybridizing on GA and HNN. GA and HNN were individually applied to resolve the SCP [3, 4, 5]. In first section we used GA and HNN independently to solve Standard Cell Problem. In the second section we present a new hybrid of GA and HNN. In the last section of the paper we compare the results of hybrid system of GA and HNN with independent result of GA and HNN in respect of wire length and CPU time.

Keywords- NP Hard, Standard Cell Problem, Genetic Algorithm, Hopfield Neural Network.

1. INTRODUCTION

Placement is the most important problem during physical design stage. It is accountable for decreasing the area of the chip and interconnection wire-length. Therefore, placement is the basic step in decreasing the fabrication cost per chip and increasing its performance. The placement problem arises after the circuit is partitioned into cells. The assignment of standard cell placement is to organize the cells on rows without overlap with minimum interconnecting wire-length and area of chip. Besides, the placement is routable. Sometimes the length and height of the chip are given, or sometimes only the aspect ratio (ratio of height and width of the chip) or, number of feed-through cells is specified. **The standard cell problem is stated as**: Given an electrical circuit consisting of fixed rectangular shaped cells and a net-list stating interconnections among terminals on the periphery of the cells and on the periphery of the circuit itself, it is required to construct a layout indicating the position of each cell such that all the nets can be routed and the total area is minimized. The purpose for high performance systems is to reduce the complete delay of the system by reducing the length of the critical paths [12]. The value of placement is based on layout area, completion of routing and circuit performance. It has been seen that standard cell placement problem is computationally hard [11], a NP-Hard problem. These problems cannot be resolved in polynomial time [5].

1.1 HOPFIELD NEURAL NETWORK

The early neural network models were feed-forward networks with three layers of neurons. Input layer excited the neurons. Its output multiplied by the weighing factors loaded in the second layer, activates the third layer. Later Hopfield proposed a recurrence based network with high degree of connectivity-NN [7]. It uses the McCulloch-Pitts neuron model [9]. Hopfield demonstrated that a large number of densely connected simple units provide a massive and powerful parallel system which can solve complicated problems. His network was a one layer of neurons with feedback. The neurons of the output layer receive the weighted inputs and send response to other „remaining" neurons. He also established similarity between the physical systems and the recurrent NN model. Hopfield, in his earlier work, showed that these systems are capable for solving Content Addressable Memory problems [1]. In 1984, he proposed yet another neuron model that behaved in a continuous manner as does a real neuron in the brain. In 1985, Hopfield and Tank [8] demonstrated the capability of this recurrent network model extended in solving optimization problems.

1.2 GENETIC ALGORITHM

Genetic Algorithms (GA) are again derivative-free stochastic optimization methods, based on the concepts of natural selection and evolutionary processes. It was initially proposed by John Holland [6] in 1975. The technique is finding several significant applications in diversified avenues and in this context their applications are extending beyond academia. Their popularity is attributed to their freedom from dependence of functional derivatives and ability to incorporate the characteristics such as:

- GAs are parallel-search procedures and are implemental on parallel processing machines for speeding up their operations.
- GAs are applicable to both continuous and discrete optimization problems.
- These are stochastic and least likely to be trapped in local optimization.
- GA"s flexibility facilitates both structure and parameter

identification in complex models such as neural networks and fuzzy inference systems.

GA encodes each point of the solution space into an ordered triplet of symbols in a form of fixed length string called chromosome. This chromosome is an „individual" in the space called 9population. The size of the population is maintained fixed. Each cell of the chromosome, called gene, is associated with a fitness value which is derived from the objective function evaluated at that point. The algorithm is iterated within the population space to get new generation with better fitness values. In every iteration, GA constructs a new population using the set of genetic operators like crossover, mutation and inversion. Members with higher fitness values stay in population. They then, participate in future cross-over operations to produce off-springs of even better fitness value. Chromosomes with inferior most values are discarded to maintain the population size and thus improve the quality of the population [10]. Therefore GAs and their variants are also referred to as methods of fixed population-based group optimization that improve performance by upgrading entire populations through its individual members.

2. STANDARD CELL PLACEMENT TECHNIQUES

2.1 Standard Cell Placement by Hopfield Neural Network

Hopfield and Tank showed that an objective function of a dynamic system and its energy are simultaneously minimal**;** which, according to the system theory, is observed only in steady state. Thus optimization solution of a dynamical system also lies in its stable state. The task is to place the given number of cells on the given layout such that the total wire-length is shortest possible. The placement would be a valid if:

1. A given matrix element should be filled with one cell only, this constraint is mathematically represented as:

$$E_1 = (-1/2)\, A \sum_K \sum_{L=K} \sum_x (out)_{Kx} (out)_{Ly} = O_2$$

Here 20 describes the order of the square matrix and 'O', that all its cells are vacant. A is a constant, K, L are the indices of cells and x, y are indices of positions on the layout.

2. A cell should not be placed on more than one positions and With B as a constant, this constraint is mathematically represented as:

$$E_2 = (-1/2)\, B \sum_K \sum_x \sum_{y=x} (out)_{Kx} (out)_{Ly} = O_{20}$$

3. Every cell should be placed on the layout, where C is a constant again. This constraint is mathematically represented as:

$$E_3 = (-1/2)\, C \{ \sum_K \sum_x (out_{Kx}) \}^2 = O_{20}$$

4. The goal is to minimize the wire-length of total interconnection. Assuming the energy function as the objective function and minimizing this should lead to the goal. Therefore we select an m- dimensional energy function. As explained in the previous section, the energy function should be bilinear. Hence, the objective function for Hopfield network that minimizes the pair-wise wire-length is used. Thus this energy function can be written as:

$$E_4 = (-1/2)\, D \sum_K \sum_{x_i} \sum_{L=K} \sum_{y=x} D_{xy}\, N_{kl}\, (out)_{Kx}\, out)_{Ly} = O_{20}$$

The sum of all these energy terms is the total energy function:

$$E = E_1 + E_2 + E_3 + E_4$$

Figure 1 describes the algorithm to implement as sequential machine.

Read the input data
Develop Distance matrix Dxy and the connection- matrix NKL based on the input data
Calculate the weight matrix $W_{Kx\,Ly}$
Initialize the network by giving some input to each neuron
LOOP
For each neuron update its activation level according to equation
$\frac{du_i}{dt} = \frac{u_i}{dn_i} + \sum W_{ij} g_j(u_j) + I_i$
Update the input values for all neurons according to output given by neurons
Calculate the energy function E
UNTIL termination criterion [E1= E2=E3=0] is reached

Figure 1: HNN Algorithm for Standard Cell Placement

2.2 Standard Cell Placement by Genetic Algorithm

To solve any optimization problem by GA, an appropriate solution representation (an encoding) is needed. Then to suit the representation, appropriate cross-over, mutation, and inversion operators are constructed. All these must generate consistent solutions.

For SCP, the representation of SA can be used. Thus macro blocks are not explicitly considered for simplification but the order of cells and their positions on the chip, indicated by co-ordinates in a given unit of measurement, are also mapped.

Up to this point, the representation chosen for the SCP problem and the GA: cross-over, mutation and inversion operators have been discussed. In the following paragraphs, the complete algorithm to implement is described.

Read input parameters: number of cells N_c, population size N_p, cross-over rate R_c, mutation rate R_m, inversion rate R_i and number of generations MAXGEN.
Create initial population by generating N_p feasible solutions, i.e. sequences of cells, randomly. Calculate x and y coordinates of the sequences of cells generated randomly.
Evaluate and store the wire-length computed for the sequences. Calculate the number of offspring to be generated as $N_o = N_p * R_c$. REPEAT
REPEAT
Iteration = Iteration + 1
Apply inversion operator to individuals with probability R_i.
Calculate x and y coordinates and fitness values of the individuals which were inverted. Select two parents according to their fitness values.
Apply cycle cross-over to the parents in order to create children with probability R_c. Apply mutation operator to the children with probability R_m.
Calculate x and y coordinates of the children.
Evaluate children by computing the total bounding rectangle wire-length.
UNTIL Iteration = $N_o/2$
Create new population by selecting fitter individuals.
generation = generation + 1
UNTIL generation = number of generations given.

Figure 2: Genetic Algorithm for Standard Cell Placement

2.3 Hybrid of Ga & HNN for standard cell placement

Performance of the HNN was shown inferior to SA and GA. The computing time taken by the HNN had also been prohibitively high. The reason has been the use of bilinear energy function that minimizes the pair-wise wire-length and not the required total bonding rectangle wire-length. In this section, we propose ways to overcome these limitations.

3. COMPARISON AND RESULTS

A hybrid system of a HNN and GA is introduced. Here, we make use of the observation that the HNN takes very long to converge and in that 90% of the time is spent for the placement of about 20% of the last cells, in our strategy to reduce convergence time in hybrid. **This above observation is researched again** with the help of energy versus iteration plots in following figures. Here we take 20 cells with parameters sets: A=B=C=1 and D=0.0001.
We run the HNN for 20%, placing 4 cells out of 20. The placement is shown in table 1.

Table 1: Intermediate solution: placement of 20% cells

Cell Placed			1	17						9
Position	1	2	3	4	5	6	7	8	9	10
Cell Placed								15		
Position	11	12	13	14	15	16	17	18	19	20

The HNN is run again to place 40%; 8 of the 20 cells, as in table 2.

Table 2: Intermediate solution: placement of 40% cells

Cell Placed	11		**1**	19			8	7		**9**
Position	1	2	3	4	5	6	7	8	9	10
Cell Placed						2		**15**		
Position	11	12	13	14	15	16	17	18	19	20

We observe that 3 of these 8 placed cells are placed exactly in same location as in earlier case with an exception of cell no. 17 displaced by cell no. 19 in position 4, shown grey.

The HNN is again run this time to place 12 cells. Positions of all the 8 cells (marked bold) of earlier case are intact. This can be seen in table 3.

Table 3: Intermediate solution after placement of 60% cells

Cell Placed	**11**		**1**	**19**	18		**8**	**7**		**9**
Position	1	2	3	4	5	6	7	8	9	10
Cell Placed	12	13				**2**		**15**		3
Position	11	12	13	14	15	16	17	18	19	20

We again run the HNN to place 16 cells. We observe that all the 12 cells are intact as in the location of table 4.

Table 4: Intermediate solution after placement of 80% cells

Cell Placed	**11**		**1**	**19**	**18**	16	**8**	**7**		**9**
Position	1	2	3	4	5	6	7	8	9	10
Cell Placed	**12**	**13**	6	5		**2**	20	**15**		**3**
Position	11	12	13	14	15	16	17	18	19	20

In the next and final step, the Hopfield network is run without any constraint. It places all the cells as shown in table 5.

Table 5: Intermediate solution after placement of 100% cells

Cell Placed	11	17	1	19	18	16	8	7	14	9
Position	1	2	3	4	5	6	7	8	9	10
Cell Placed	12	13	6	5	4	2	20	15	10	3
Position	11	12	13	14	15	16	17	18	19	20

It is apparent in the above example that placement by HNN is almost rigorous. The cells which are once placed are least disturbed while placing the remaining cells. Then why not to run the HNN to place say, 75% of the cells and explore other method to place the remaining cells. This instigates us to propose a method using hybridizing. Thus, we use HNN for the placement of 50% to 80% of the cells. Remaining cells are proposed to be placed by GA. GA is designed to put the unplaced cells at the empty positions on the layout while minimizing the wire length. The placements so obtained are sent back to HNN for validation.

Two placement problems consisting of 20 and 40 cells [test cases A to F] are taken for illustration. In table 4.1, the results of the six placement problems, when solved by HNN alone are shown. Later the same problem is solved by hybrid approach treating 20%, 40%, 60% and 80% of cells placed by HNN and remaining by GA are shown in table 6.

Table 6: Results of HNN alone

Test Case	Average		CPU Time	Wire-length	
	Cells	Nets	Cell width	(sec)	(µm)
A	20	20	30	6800	3120
B	20	18	30	5875	3222
C	20	16	30	5474	2648
D	40	15	25	42458	4119
E	40	14	25	41522	3986
F	40	11	25	38667	3824

It is perceived that:

(a) To place 20 cells, the Hopfie1d network needs approximately 90 minutes to 115 minutes as against 20-25 minutes in hybrid system (A, B, C). To place 40 cells, the network takes about 10 hours 45 minutes to 11 hours 45 minutes while with hybrid system, it takes about two hours (D, E, F), a fivefold improvement. It is graphically expressed in figure 3.

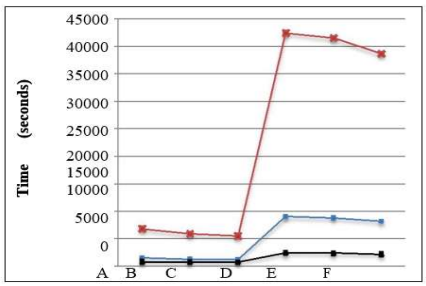

Figure: 3: Comparison of CPU time of GA, HNN and hybrid HNN-GA (80% cells placed by HNN)

(b) Wire Length: It is within 1% variation in both techniques.

4. CONCLUSION

It is specious that hybrid of HNN and GA compromises an important enhancement in the performance of algorithm decreasing the CPU time many fold. Note that when linked to average CPU time with pure GA**,** the time taken by the hybrid rises with the size of problem. So, for placement of more number of cells, this methodology is not suggested. Hybrid of HNN and GA has been tried first and it somewhat considerably contributed in speeding up the performance of HNN but it is quiet very slow.

REFERENCES

[1] Anderson, J. E. and Rosenfeld, E. (Eds.): "Neurocomputing: Foundations of Research; Cambridge", MA (1988).

[2] Bunglowala, A., Singhi, B. M., "Memetic Algorithms as a Solution to combinatorial Optimization Problem", Proceedings of 2nd PIMR International Conference, 2008.

[3] Bunglowala, A., Singhi, B.M., "Performance Evaluation and Comparison and Improvement of Standard Cell Placement in VLSI Design", International Conference on Emerging Trends in Engineering and Technology, July 2008

[4] Bunglowala, A., Singhi, B.M., "A Solution to combinatorial Optimization Problem using Memetic Algorithms", International Journal of Computer System Applications [IJCSA], December 2008.

[5] Donath, W. E.: "Complexity theory and design automation"; Proceedings of the 17th Design Automation Conference (1980).

[6] Holland, J. H.: Adaptation in Natural and Artificial Systems; the University of Michigan Press, Ann Arbor (1975).

[7] Hopfield, J. J.:"Neural networks and physical systems with emergent collective computational abilities" Proceedings of the National Academy of Sciences U.S.A. 79 (1982) April.

[8] S. Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic algorithms for solving job-shop scheduling problems" Memetic Computation, vol. 1, no. 1, pp. 69–83, Mar. 2009.

[9] McCulloch, W. S. and Pitts, W.: A Logical Calculus of the Ideas in Nervous Activity; in Bull. Math. Biophys. 115 (1943).

[10] Shahookar, K. and Mazumder, P.: "A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization" IEEE Transactions on Computer-Aided Design 9 :(1990).

[11] Shahookar, K. and Mazumder, P."VLSI Placement Techniques" ACM Computing Surveys 23(1991).

[12] Sherwani, N. A.: "Algorithms for VLSI Physical Design Automation" Boston (1993).