International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:9, No:6, 2015

Music-Inspired Harmony Search Algorithm for Fixed Outline Non-Slicing VLSI Floorplanning

K. Sivasubramanian, K. B. Jayanthi

***Abstract*—**Floorplanning plays a vital role in the physical design process of Very Large Scale Integrated (VLSI) chips. It is an essential design step to estimate the chip area prior to the optimized placement of digital blocks and their interconnections. Since VLSI floorplanning is an NP-hard problem, many optimization techniques were adopted in the literature. In this work, a music-inspired Harmony Search (HS) algorithm is used for the fixed die outline constrained floorplanning, with the aim of reducing the total chip area. HS draws inspiration from the musical improvisation process of searching for a perfect state of harmony. Initially, B*-tree is used to generate the primary floorplan for the given rectangular hard modules and then HS algorithm is applied to obtain an optimal solution for the efficient floorplan. The experimental results of the HS algorithm are obtained for the MCNC benchmark circuits.

***Keywords*—**Floor planning, harmony search, non-slicing floorplan, very large scale integrated circuits.

I. Introduction

FLOORPLANNING is the first step of the physical design in the VLSI design flow. It provides early feedback that evaluates architectural decisions and estimates delay and congestion caused by wiring. It is an essential design step to estimate the chip area by considering the optimal placement of digital blocks and their interconnections. Each block consists of several hundreds or thousands of cells that perform a specific operation. The blocks are of rectangular shape with different aspect ratios. The blocks can be classified into two types based on their shape flexibility. They are hard blocks and soft blocks. Hard block has fixed width and height whereas soft block's width, and height can be varied as long as its aspect ratio is within the given range, and its area is fixed. The aspect ratio of a block is defined as the ratio between the height and the width of a block. To optimize the area of the chip, hard blocks are rotated then the width and height of soft blocks are modified without affecting the total area of the block. The classical floorplanning methods normally handles only block packing to minimize the total chip area, but modern floorplanning methods could be devised as a fixed outline floorplanning.

There are two types of floorplanning methods used in electronic design automation in which the first one is slicing floorplan and the second one is non-slicing floorplan. In slicing floorplan, the whole block area is first partitioned into two slices of equal or unequal sizes using either a horizontal or vertical line then the individual blocks are again partitioned into by using either horizontal or vertical lines. This process continues until all the blocks are separated. This process of partitioning the block area is called slicing floorplan, as depicted in Fig. 1. The slicing floorplan is represented by a binary tree structure known as a slicing tree, as shown in Fig. 2. The leaf nodes of the slicing tree are the blocks of the design. The other nodes are either a vertical partition represented by V or a horizontal partition represented by H.

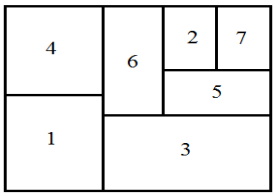

Fig. 1 Slicing Floorplan

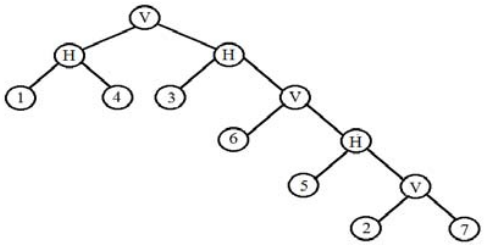

Fig. 2 Slicing Tree Structure

If a floorplan is obtained with no recursive through cuts, then it is called as non-slicing floorplan. Fig. 3 represents the non-slicing floorplan structure. Different techniques are used to represent the non-slicing structure of a floorplan. They are sequence pair, Bounded Slicing Grid (BSG) O-tree and B*-tree representations [1]-[4].

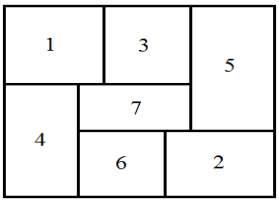

Fig. 3 Non-Slicing Floorplan

A B*-tree is an ordered binary tree for modeling non-slicing floorplans. In a compacted floorplan, no blocks can be moved toward left or bottom in the floorplan. Accordingly, an area-optimal floorplan always corresponds to some B*-tree.

K.Sivasubramanian and K. B. Jayanthi are with the K. S. Rangasamy College of Technology, Tiruchengode, Tamilnadu, India (e-mail: sivameae@gmail.com, jayanthikb@gmail.com).

Its root corresponding to the block at the bottom left corner. The construction of B*-tree is similar to the Depth-First Search (DFS) algorithm. In B*-tree, starting from the root node, construction of left subtree and then the construction of right subtree is done in a recursive fashion. B*-trees are very easy to implement and can perform the three primitive tree operations such as search, insertion, and deletion. Fig. 4 denotes the admissible placement of the modules and Fig. 5 gives the corresponding B*-tree representation.

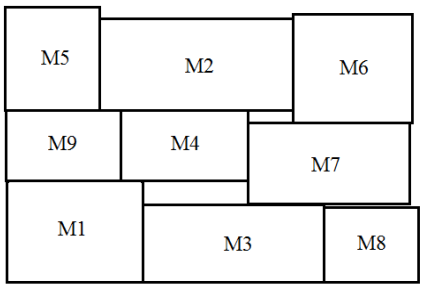

Fig. 4 Admissible Non-slicing Floorplan

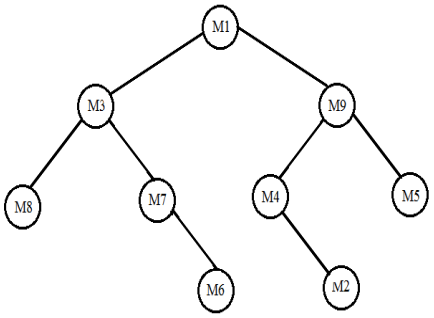

Fig. 5 B*-Tree Representation

Let R_i represents the set of modules located on the right-hand side and adjacent to module m_i. The left child of the node n_i corresponds to the lowest unvisited module in R_i. The right child of the node n_i represents the module located above and adjacent to m_i, with its x-coordinate equal to that of m_i and its y-coordinate less than that of the top boundary of the module on the left-hand side and adjacent to m_i. Let T be the root node with the coordinates $(x_{root}, y_{root}) = (0, 0)$ and n_j *be* the node either on the left or right side. If the node n_j is on the left of root node n_i, then the module m_j is placed on the right-hand side of module i (i.e., $x_j = x_i + w_i$), where w_i is the width of module i. Similarly, if the node n_j is on the right of root node n_i, then the module m_j is placed above the corresponding root node n_i, i.e., $x_j = x_i$.

Every B*-tree represents a possible module placement solution. To find the next possible solution, the B*-tree is perturbed by any one of the operations like the movement of the node, swapping the nodes and node rotation to get another B*-tree. Rotate the node (block) by 90◦ such that the height and width of the corresponding block are swapped. If the height and width of the blocks are not equal, then after rotation the x and y-coordinates of each block, should be exchanged, yielding a new B*-tree.

Optimization is the act of achieving the best possible result under given conditions. The objective of any optimization algorithm is to minimize or maximize the objective function. An optimization algorithm is a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found. Optimization algorithms are mostly used in all engineering problems. Since VLSI floorplanning is an NP-hard problem which is to be optimized, many optimization techniques were used in the literature.

Harmony Search algorithm is one of the optimization technique. It has been successfully applied in the fields of function optimization, mechanical structure design and pipe network optimization [5]-[7]. It does not need any earlier domain knowledge, such as the gradient information of the objective function. It requires fewer mathematical requirements and does not entail initial value settings of the decision variables. As it uses stochastic random searches, derivative information is also not necessary. HS algorithm generates a new solution, after considering all the existing solutions. These features increase the flexibility of the HS algorithm and produce better solutions [8]. HM stores the past search experiences and plays an important role in its optimization performance. HS has the attractive advantage of algorithm simplicity. In this work music-inspired Harmony Search algorithm is used for VLSI floorplanning.

II. Related Work

Many researchers have focused on optimization algorithms for floorplanning in both slicing and non-slicing structures [9]-[13]. O-tree representation was proposed in [3] based on a non-slicing floorplan. A boundary constraint algorithm for general floorplan by extending the Corner Block List (CBL) algorithm, which was an efficient topology representation for non-slicing floorplan, was proposed in [14]. Their contribution was to find the necessary and sufficient characterization of the modules along the boundary represented by Corner Block List.

A widely used global search method for VLSI floorplanning problems is Genetic Algorithm (GA). GA has been successfully applied to solve slicing VLSI floorplanning problems [15]-[17]. For non-slicing VLSI floorplanning, a GA has also been presented by [18]. Since the encoding scheme does not capture any topological information of the floorplans, the performance of the GA was not satisfactory. Tabu search algorithm was used to solve module placement problem in [19]. Initially, all the modules are merged into some clusters according to the ratio-connectivity of circuit modules. The placement of the large modules is represented by sequence-pairs. The searching of the optimal solution for the placement was performed by the Tabu search algorithm.

A GA to tackle the VLSI floorplanning problem using O-tree representation was proposed in [20]. Particle Swarm Optimization (PSO) was introduced into the floorplanning problem to find the potential optimal placement solution [21]. The implementation details of the algorithm were not stated in

this work; only the area optimization was considered. This structure was unable to solve problems that optimize the area and wirelength simultaneously. A Memetic Algorithm (MA) for a non-slicing and hard-module VLSI floorplanning problem was presented in [22]. A hybrid genetic algorithm that uses an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region was used in this work.

A multi-objective genetic algorithm for floorplanning that simultaneously minimizes area and total wirelength was proposed in [23]. The proposed genetic floorplanner was the first to use non-domination concepts to rank solutions. In this work, two novel crossover operators were presented that build floorplans using good sub-floorplans. A novel floorplanning algorithm based on Discrete PSO (DPSO) algorithm, in which integer coding based on module number was adopted in [24]. The principles of mutation and crossover operator in the GA were also incorporated into the proposed PSO algorithm to achieve better diversity and escape from local optima. This algorithm can avoid the solution from falling into local minimum and have good convergence performance.

A PSO algorithm was used for floorplanning where area and wirelength were considered as the fitness function [25]. A new heuristic method was proposed that applied Hybrid Simulated Annealing (HSA) to represent a non-slicing floorplan with an objective function by restricting the area and wirelength [26]. A sequence pair approach was used to represent non-slicing floorplans with a smaller search space [27]. Shanavas et al. proposed a method that combined a hierarchical design technique like genetic algorithm and constructive technique like Simulated Annealing for local search to solve VLSI partitioning and floor-planning problem [28]. A Co-evolutionary Multi-Objective Particle Swarm Optimization (CMOPSO) algorithm was proposed to solve a VLSI floorplanning problem which was a multi-objective combinatorial optimization and has been proved to be an NP-hard problem [29]. The algorithm imported the concept of the co-evolutionary algorithm and elitist strategy into basic PSO algorithm. It took both the layout area and total interconnection wire lengths into consideration simultaneously.

An approach based on iterative Prototypes Optimization with Evolved improvement (POEMS) algorithm was proposed by [30]. It used a GA for local search and adopted a non-slicing structure called B*-tree for the placement of rectangular modules. A VOAS (Variable – Order Ant System) for area optimization based on the ant algorithms was proposed in [13]. This method used the models that are derived from the observation of real ants' behavior, and used as a source of inspiration for the design of novel algorithms for the solution of optimization and distributed control problems. A smart decision-making PSO-GA based hybrid method for thermal-aware non-slicing VLSI floorplanning was used in [31]. B*-tree representation has been used in this method. VLSI floorplanning is an NP-hard problem. The solution space will increase exponentially with the growth of circuits scale. Thus, it is difficult to find the optimal solution by exploring the global solution space.

III. Proposed Methodology

A. Harmony Search Algorithm

The Harmony Search (HS) method is a meta-heuristic optimization algorithm proposed in [32]. It mimics a musical improvisation process in which the musicians in an orchestra/band try to find a perfect state of harmony through musical improvisations. When musicians compose harmonies, they usually try various possible combinations of the music pitches stored in their memory. This algorithm was designed to mimic the way a musician uses short-term memory and the past experiences to lead his/her to the note that results in the most pleasing harmony when played together with the other musicians. HS is easy to implement and can easily be applied to solve almost any problem that can be designed as the minimization or maximization of an objective function. This kind of efficient search for a perfect state of harmonies is related to the procedure of finding the optimal or near-optimal solutions for a problem. When solving a particular problem, each musician is considered as a decision variable. So, the perfect harmony means the global or near-global solution.

In HS, each musician corresponds to a decision variable in the solution vector of the problem and also represents a dimension in the search space. Each musician (decision variable) has a different instrument whose pitch range corresponds to a decision variable's value range. A solution vector, also called an improvisation, at certain iteration corresponds to the musical harmony at a particular period, and the objective function corresponds to the audience's aesthetics. New improvisations are based on earlier remembered good ones which are stored in the data structure called the Harmony Memory (HM). A new solution is improvised by using three rules. They are (a) Play what he/she exactly knows (memory consideration) (b) Play by slightly adjusts the pitch (pitch adjustment) and (c) Play a new composition (random selection).

The main control parameters of HS algorithm are harmony Memory (HM), Harmony Memory Size (HMS), Harmony Memory Considering Rate (HMCR), Pitch Adjusting Rate (PAR), and Bandwidth (BW). Here, HM is a memory location where all the solution vectors are stored; HMCR and PAR are parameters that are used to improve the solution vector.

1. Harmony Memory and Improvisation Process

The core data structure of HS is a matrix of the best solution vectors called the HM. The number of vectors that are concurrently processed is known as the Harmony Memory Size (HMS). It is one of the algorithm's parameters that have to be set manually. Memory is structured as a matrix with each row represents a solution vector, and the final column represents the vector's fitness value. In the HS algorithm, X represents the harmony and $f(\mathrm{X})$ denotes the melody of harmony X. In a N-dimensional problem, the HM would be represented as:

$$\begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^N & | & w_1 \\ x_2^1 & x_2^2 & \dots & x_2^N & | & w_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{HMS}^1 & x_{HMS}^2 & \cdots & x_{HMS}^N & | & w_{HMS} \end{bmatrix}$$

Before optimization starts, the HM is initialized with HMS randomly generated solution vectors. Based on the problem, these vectors can also be randomly chosen around a seed point that may represent an area in the search space where the optimum is most likely to be found [33].

Each decision variable is improvised individually, and any one of the three rules can be applied for any variable. The HMCR is one of the HS parameters that must be manually chosen. It controls how often the HM is taken into consideration during improvisation. For standard HS, memory consideration means that the decisions variable's value is chosen directly from one of the solution vectors in the HM. A random number is generated for each decision variable. If it is less than the HMCR, the memory is taken into consideration; else, a value is randomly chosen from the range of possible values for that dimension. The Pitch Adjustment Rate (PAR) is set during initialization, and it controls the amount of pitch adjustment done when memory consideration is used. Another random number is generated. If it is smaller than the PAR, the improvised value is pitch adjusted using (1):

$$x'_{new} = x_{new} + rand().BW \quad (1)$$

wherex'_{new} is the new pitch-adjusted value, x_{new} is the old value chosen using memory consideration, rand() is a random value between −1 and 1, and BW is the Bandwidth parameter. It is the maximum variation in pitch adjustment and is one of the parameters that must be manually set. Once a new value has been improvised, the memory is updated by comparing the new improvisation with the vector in the memory with the lowest fitness. If the new improvisation has a higher fitness, it replaces the vector with the lowest fitness. This process of improvisation and update continues iteratively until some stopping criterion is fulfilled, or the maximum number of iterations is reached.

2. Steps in HS

Step1. Initialize the problem and algorithm parameters

Step2. Initialize the Harmony Memory (HM)

Step3. Improvisation

In this step, New Harmony vector is generated based on three rules, namely, memory consideration, pitch adjustment, and random selection. The value of a design variable can be selected from the values stored in HM with a probability HMCR. It can be further adjusted by moving to a neighbor value of a selected value from the HM with a probability of PAR, or, it can be selected randomly from the set of all candidate values without considering the stored values in HM, with the probability of (1 - HMCR).

Step4. Update HM

If the new harmony vector is better than the worst vector, based on the objective value and/or constraint violation, the new vector will replace the worst one.

Step5. Termination criterion

HS algorithm is terminated if the stopping criterion (maximum number of improvisations) has been met; else steps 3 and 4 are repeated.

3. Pseudocode of Harmony Search Algorithm

```
begin
  Set the objective function of the problem as f(x)
  Generate initial Harmony memory and initial harmonics
  Define pitch adjusting rate (r_pa), pitch limits, and bandwidth
  Define harmony memory accepting rate (r_accept)
  while ( t<Maximum number of iterations )
    Generate new harmonics by accepting the best harmonics
    Adjust pitch value to get new harmonics (solutions)
    if (rand>r_accept), choose a harmonic randomly from HM
    else
    if (rand>r_pa), adjust the pitch randomly within the limits
    else
    generate new harmonics via randomization
    end if
    Accept the new harmonics (solutions) if they are better
  end while
  Find the current best solutions
end
```

4. Fitness function Evaluation

For the HS algorithm in floorplanning, each musician corresponds to a possible solution. The main objective of the floorplanning is to minimize the total chip area and wire length. The formula for calculating the total area of the chip that contains 'i' modules is given in (2):

$$Area = \sum_{i=1}^{n} (length(i) * width(i)) \quad (2)$$

The wirelength of the i^{th} module with respect to the other modules in the given floorplan can be calculated by using Half-Perimeter Wirelength (HPWL). It is defined as half the perimeter length of the smallest bounding box that encloses all pins. Let us consider net i, connected with different terminals, a small rectangle box that encompasses all the terminals chosen. The HPWL of the net i can be calculated according to (3):

$$W_L = (X_{max} - X_{min}) + (Y_{max} - Y_{min}) \quad (3)$$

Here X_{max} and X_{min} are the maximum and minimum x-coordinates of the HPWL bounding box of the net. Y_{max} and Y_{min} are the maximum and minimum y-coordinates of the HPWL bounding box of the net. The total wirelength can be calculated using the formula denoted in (4):

$$Wirelength = \sum_{i=1}^{m} W_L. \quad (4)$$

A fitness function must be devised for each problem to be optimized. It is a designed function that measures the goodness of a solution. It is essential to estimate how good a possible solution is relative to other potential solutions. The fitness function is responsible for performing this evaluation and returns a fitness value. In each iteration, the priority of the solution vector is ranked according to the fitness value

calculated using the fitness function. By maximizing or minimizing the fitness values in each generation, the global optimum value could be found. The fitness function for the proposed method is given in (5).

$$f(x) = \alpha * Area + \beta * Wirelength \quad (5)$$

Here, α and β values are treated as the weighting factors. In this work α value is taken as 0.8 and β value is taken as 0.2.

IV. Results and Discussion

The proposed method utilizes fixed die outline with hard rectangular modules for floor-planning. The experiments in this study make use of MCNC (Microelectronics Center of North Carolina MCNC 2004) benchmark circuits for the proposed floor-planners. Simulations have been carried out for the MCNC benchmark circuits namely apte, ami33, hp, and xerox. Table I shows the characteristics of MCNC benchmark circuits. The second column of Table I shows the number of cells/modules within the circuit. The third column presents the number of nets connecting the cells within the circuits. The fourth column indicates the pads that connect the circuit to the outside world. The total number of pins within the circuit is summarized in column five. All the blocks were considered as hard IP blocks. The coding for the proposed floorplanning has been written and simulated in MATLAB version R2013a. Table II gives the Parameters and their values of the optimization algorithms used in this method.

TABLE I
Characteristics of MCNC Benchmark Circuits

Name of the Benchmark Circuit	Number of Modules	Number of Nets	Number of I/O Pad	Number of Pins
apte	9	97	73	287
ami33	33	123	42	522
hp	11	83	45	309
xerox	10	203	2	698

TABLE III
Parameters and Their Values of HS Algorithm

Parameter	Value
HMCR	0.9
HMS	30,50,100
Maximum no. of Iterations	100,500,1000
PAR	0.3
BW	0.01
α	0.8
β	0.2

The performance of the HS algorithm is evaluated for VLSI floorplanning. In this work, the HMS value is taken as 30, 50 and 100. The proposed algorithm is run for 100, 500 and 1000 iterations. Table III shows the results of the proposed algorithm. Table IV compares the performance of the proposed work with other existing works.

TABLE IIIII
Performance of the Proposed Method

Circuit Name	HMS	HS Area in mm^2(%)		
		Number of Iterations		
		100	500	1000
apte	30	48.95	48.47	48.43
	50	48.47	48.43	48.21
	100	48.47	48.47	48.43
ami33	30	1.52	1.38	1.35
	50	1.835	1.65	1.35
	100	1.75	1.65	1.30
hp	30	10.50	10.16	10.05
	50	10.98	10.79	10.54
	100	10.63	10.26	10.03
xerox	30	20.97	20.64	20.64
	50	21.86	21.86	20.64
	100	21.86	20.97	20.64

TABLE IVV
Comparison of Results with Other Work

Method	Published results (Area in mm^2)			
	apte	ami33	hp	xerox
[4]	46.92	1.27	8.95	19.83
[34]	47.01	1.19	9.13	20.14
[12]	48.47	1.23	9.48	20.42
[26]	48.12	1.25	9.43	21.86
[31]	47.44	1.24	-	20.2
HS	48.21	1.30	10.03	20.64

The optimum result for the apte circuit is obtained when HMS value is 50 and 100. When HMS value is 100 the minimum area is achieved for ami33 circuit. For hp circuit, the minimum area is obtained in 1000^{th} iteration with HMS value as 50. The minimum area for xerox circuit is attained when HMS value is 30 and 100. Figs. 6-9 show the performance of the HS algorithm in floorplanning with different HMS values and different iterations.

V. Conclusion

VLSI floorplanning is an NP-hard combinatorial optimization problem. To solve this problem in an efficient way, music–inspired Harmony Search algorithm is proposed in this method. The simulation results of the MCNC benchmark circuits have good and reasonable solution for the non-slicing placement of hard modules within fixed outline constraints. From the results it is inferred that the performance of the proposed method is comparable to existing algorithms. When a number of iteration increases, the total area of the floorplan gets reduced. The HMS value does not influence the performance of the algorithm, so the value of HMS should be identified through the empirical analysis.

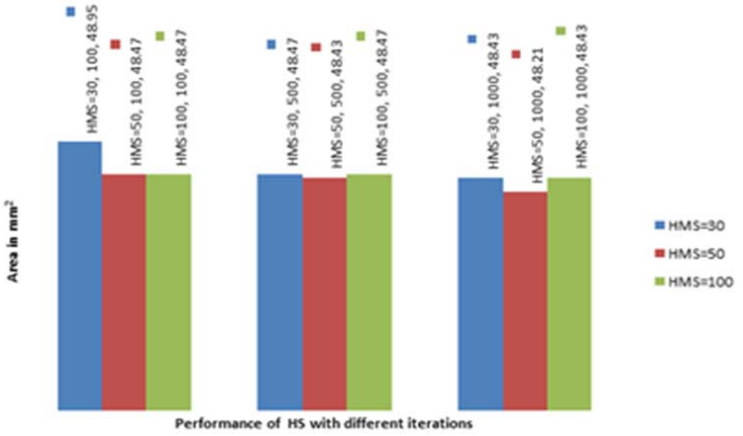

Fig. 6 Result of apte circuit

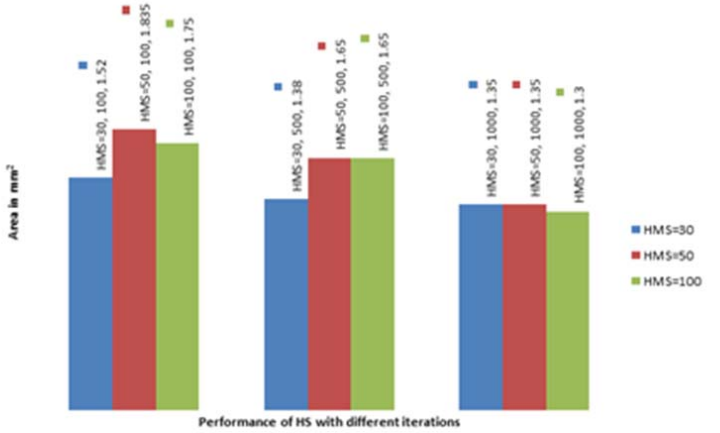

Fig. 7 Result of ami33 circuit

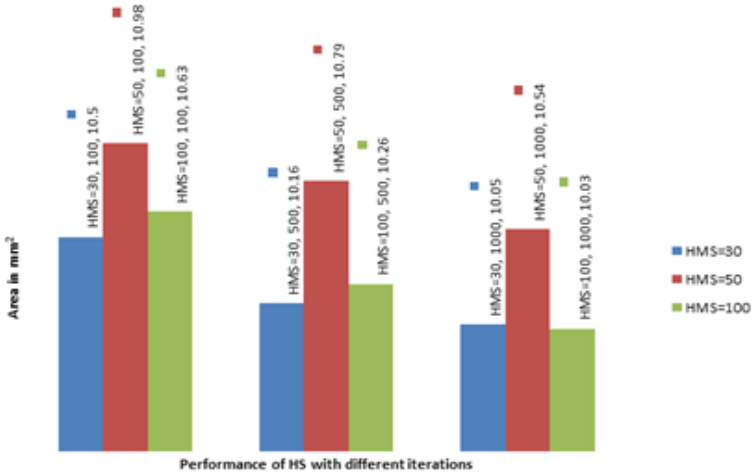

Fig. 8 Result of hp circuit

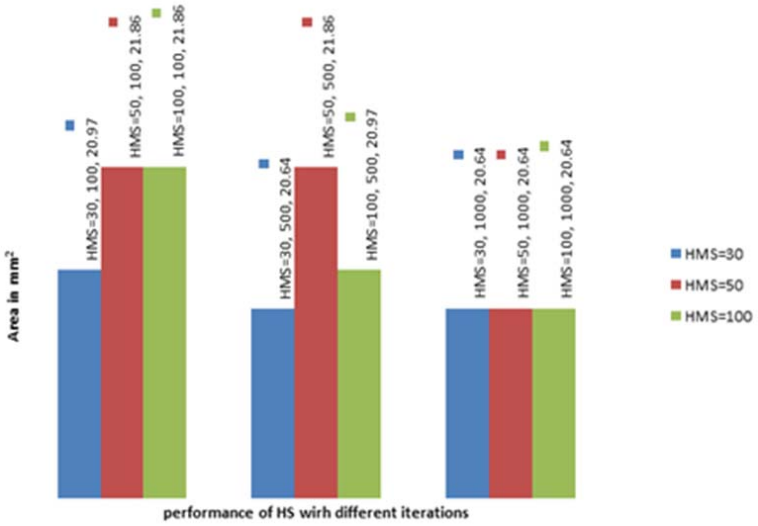

Fig. 9 Result of xerox circuit

REFERENCES

[1] H. Murata, K. Fujiyoshi and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518-1524, 1996.

[2] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, "Module placement on BSG-structure and IC layout applications", in *Proceedings of IEEE/ACM International Conf. on Computer Aided Design*, pp. 484-491, 1996.

[3] P.-N. Guo, C.-K. Chengand T. Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and Its Applications," *in Proceedings of the 36th Annual ACM/IEEE Design Automation Conference (ACM, 1999*, pp. 268–273, 1999.

[4] Y-C Chang, Y-W Chang, G-M Wu and S-W Wu, "B*-Trees: A New Representation for Non-Slicing Floorplans" *ACM*, 2000.

[5] K. S. Leeand Z.W. Geem "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp.3902–3922, 2005.

[6] S. L. Kangand Z.W. Geem,"A new structural optimization method based on the harmony search algorithm",*Computers and Structures*, vol. 82,no. 9–10,pp.781–798, 2004.

[7] Z.W. Geem, J.H. Kimand G.V. Loganathan, "Harmony search optimization: application to pipe network design", *International Journal of Modeling and Simulation*, vol. 22, no. 2, pp.125–133, 2002.

[8] Z. W. Geem, "Harmony search algorithm for solving Sudoku", in *Proceedings of the 11th international conference, KES 2007 and XVII Italian workshop on neural networks conference on Knowledge-based intelligent information and engineering systems*: Part I. 2007, Springer-Verlag: Vietrisul Mare, Italy.

[9] W. Huang, D. Chen and R. Xu, "A new heuristic algorithm for rectangle packing" *Comput. Oper.Res*., vol. 34, no. 11, pp. 3270–3280, 2007.

[10] Y. Pang, C.-K. Cheng and T. Yoshimura, "An enhanced perturbing algorithm for floorplan design using the O-tree representation", in *Proceedings of the International Symposium on Physical Design, ACM,* pp. 168–173, 2000.

[11] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on B*-tree and fast simulated annealing" *IEEE Trans. Comput. Aided Des.Integr.Circuits Syst.,* vol. 25, no. 4, pp. 637–650, 2006.

[12] S. Anand, S. Saravanasankar and P. Subbaraj, "Customized simulated annealing based decision algorithms for combinatorial optimization in VLSI floorplanning problem",*Comput. Optim.Appl*., vol. 52, no. 3, pp. 667–689, 2012.

[13] C.-S Hoo, K. Jeevan, V. Ganapathy and H. Ramiah."Variable-Order Ant System for VLSI multiobjective floorplanning" *Elsevier Journal on Applied Soft-Computing*, vol. 13, pp. 3285–3297, 2013.

[14] Y. Ma, S. Dong, X. Hong, Y. Cai, C.-K Cheng and J. Gu, "VLSI Floorplanning with Boundary Constraints Based on Corner Block List", *IEEE*, pp 509-514,2001.

[15] J. Cohoon, S. Hegde, W. Martin, and D. Richards, "Distributed genetic algorithms for the floorplan design problem," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.,* vol. 10, no. 4, pp. 483–492, 1991.

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

[17] C. Valenzuela and P.Wang, "VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions," *IEEE Trans.Evol. Comput*., vol. 6, no. 4, pp. 390–401, 2002.

[18] B. Gwee and M. Lim, "A GA with heuristic based decode for IC floorplanning," *Integr., VLSI J*., vol. 28, no. 2, pp. 157–172, 1999.

[19] N. Xu, X.-L Hon, S.-QDong and H.-BYu, "TSCSP: Tabu Search Algorithm for VLSI Module Placement Based on the Clustering Sequence-Pair", *IEEE*, 2003.

[20] M. Tang and A. Sebastian,*A genetic algorithm for VLSI floorplanning using O-tree representation*. In Applications of Evolutionary Computing, Springer, Berlin,pp. 215–224, 2005.

[21] T.-Y. Sun, S.-T.Hsieh, H.-M.Wang and C.-W. Lin, "Floorplanning based on particle swarm optimization", in *IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, 2006.

[22] M. Tang and X. Yao, "A Memetic Algorithm for VLSI Floorplanning", *IEEE transactions on systems, man, and cybernetics—part b: cybernetics*, vol. 37, no. 1, pp. 62-69, 2007.

[23] P. Fernando and S. Katkoori, "An Elitist Non-Dominated Sorting based Genetic Algorithm for Simultaneous Area and Wirelength Minimization in VLSI Floorplanning" In *21st International Conference on VLSI Design*, India, IEEE Computer Society, pp. 337-342, 2008.

[24] G. Chen, W. Guo, H. Cheng, X. Fen and X. Fang, "VLSI Floorplanning Based on Particle Swarm Optimization" in *Proceedings of 3rd International Conference on Intelligent System and Knowledge Engineering*, IEEE, pp. 1020-1025, 2008.

[25] G. Chen, W. Guo and Y. Chen, "A PSO-based intelligent decision algorithm for VLSI floorplanning", *Springer, Soft Computing*, pp. 1329–1337, 2010.

[26] J. Chen, W. Zhu, and M. M. Ali, "A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning", *IEEE transactions on systems, man and cybernetics—part c: applications and reviews*, vol. 41, no. 4, pp. 544-553, 2011.

[27] D. Sengupta, A. Veneris, S. Wilton, A. Ivanov and R. Saleh, "Sequence Pair Based Voltage Island Floorplanning", in *Proceedings of the 2011 International Green Computing Conference and Workshops*, IEEE computer society Washington, pp. 1-6, 2011.

[28] I.H. Shanavas and R.K. Gnanamurthy, "Wirelength Minimization in Partitioning and Floorplanning Using Evolutionary Algorithms" *Hindawi Publishing Corporation VLSI Design*, vol. 10, Article ID 896241, 9 pages, 2011.
[29] Z. Chen, J. Chen, W. Guo and G. Chen, "A Coevolutionary Multi-Objective PSO algorithm for VLSI Floorplanning" *8th International Conference on Natural Computation (ICNC),* IEEE, pp. 712-728, 2012.
[30] T. Singha, H. S. Dutta and M. De, "Optimization of Floor-planning using Genetic Algorithm" *Procedia Technology,* vol. 4, pp. 825 – 829, 2012.
[31] P.Sivaranjani and A.S. Kumar,"Thermal-Aware Non-slicing VLSI Floorplanning Using a Smart Decision-Making PSO-GA Based Hybrid Algorithm" *Circuits Syst Signal Process,* DOI 10.1007/s00034-015-0020-x, 2015.
[32] Z.W. Geem, J.H. Kimand G.V. Loganathan,"A new heuristic optimization algorithm: harmony search", *Simulation,*vol. 76, no. 2, pp.60–68, 2001.
[33] J. Fourie, S. Mills, and R. Green, "Harmony filter: a robust visual tracking system using the improved harmony search algorithm," *Image and Vision Computing*, vol. 28, no. 12, pp. 1702–1716, 2010.
[34] J. Chen and W. Zhu, "A Hybrid Genetic Algorithm for VLSI Floorplanning", in *International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, IEEE, 2010, pp. 128-132.