A Memetic Algorithm Based Novel Area Optimization Technique Using O-Tree Representation For 3D ICs

Rohin Gupta,
Research Scholar,
Department of Electronics and Communication Engineering,
I.K. Gujral Punjab Technical University,
Kapurthala–144603, Punjab, India
engg.rohin@gmail.com

Sandeep Singh Gill
Department of Electronics and Communication Engineering,
National Institute of Technical Teachers Training and Research,
Chandigarh 160019, India
ssg@nitttrchd.ac.in

***Abstract*—**

This paper introduces a novel technique for designing a 3D VLSI floorplan using O-Tree representation. In this research paper, an algorithm (tool) is developed for optimizing the area of the chip, which is a key requirement in fabricating compact and miniaturized ICs. The developed algorithm can optimize any number of tiers (or layers). This paper has worked and compared results with 2, 3, and 4 tiers. An improved memetic optimization technique is used in which the genetic algorithm is used for global exploration and an improved local technique is used to improve the result in that search region. The representation for the floorplan used is O-Tree (Ordered Tree) and the CBLSP (Code Based Location Search and Position) methodology is used to implement the representation. Results are checked and compared with MCNC (Microelectronics Center of North Carolina) benchmark circuits. Results show an improvement with the best results in all the benchmark circuits in different tiers with a range of 2% to 48.8% (or reduction) in area.

***Keywords*—3D ICs, O-tree representation, MCNC Benchmarks, Memetic algorithm, VLSI Floorplan.**

DOI Number: 10.14704/nq.2022.20.9.NQ44710 **Neuro Quantology 2022; 20(9):6086-6094**

I. Introduction

VLSI technology-based gadgets have been etched in today's generation as an essential part of life. Floorplanning is an essential step of the physical design step of the VLSI Design cycle as it determines the performance of the manufactured chip. The process of managing, placing, and arranging the blocks or cuboids and their netlist on the die is called floorplanning. Much research has commenced in the 2D floorplan techniques, now researchers are aggressively looking in the third dimension for improved performance as the size, interconnect length and consequently speed depend a lot on these parameters. When the third dimension (Z) is added to the existing 2D (X and Y plane) plane, area and interconnect lengths get shortened to a great extent enhancing the performance of the chip. Another perspective of making a 3D IC is to divide the 2D plane into several layers (or tiers). Most of the research papers have utilized 4 layers for the floorplan. In the implemented research, the 2D floorplan is divided in an efficient way into layers and the goal is to minimize the area or size of the designed floorplan. Overlapping of the modules is strictly prohibited in designing the floorplan. The designed floorplan in the presented paper is rectangular in shape. A general description of the 3D IC with an application chip (a simple microcontroller) is shown in figure 1. The various layers of 3D IC are separated by interconnects or vias. The layer may include some application chips such as processors, SRAMs, DRAMs, ADC, and some control circuitry with some sensors. The proposed work can be boon to a variety of modern-day applications including health-related applications like Brain-computer interaction, Genetic and Circuit mechanisms, neurosciences, etc.

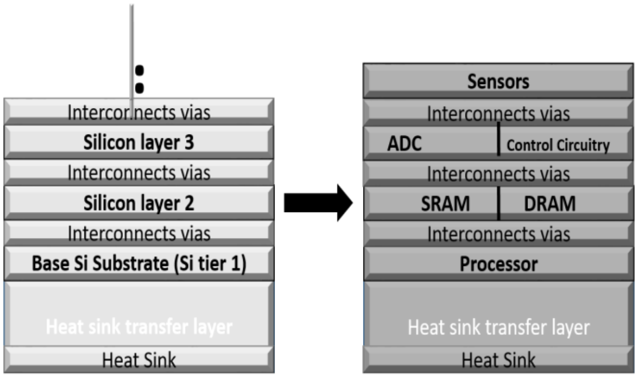


Fig. 1. A simple description of the 3D representation of blocks embedded in various layers with a possible microcontroller application as a 3D IC.

There are many representations proposed by researchers that include models based on polish expression in 1986 [1], the Branch and bound method in 1991 [2], the Sequence Pair representation in 1995 [3], the Bounded Slicing Grid (BSG) structure in 1997 [4], O-Tree representation in 1999[5], Corner Block List (CBL) in 2000 [6], one of the widely used representations: Binary tree (B*-Tree) in 2000 [7], Skewed B*-Tree in 2022 [8]. Though the binary tree is the most widely used representation, the presented paper has used the Ordered Tree or O-Tree representation [5] due to its advantages and flexibility of connecting any number of nodes to any other node which is not possible in the B*-Tree representation where a maximum of two nodes can be connected to a particular node. The representation is implemented by the CBLSP methodology [9] with non-slicing floorplan type. Also, the computational complexity to transform the O-Tree representation to its corresponding floorplan is just O(n). For n modules, the search space of the O-Tree representation is just $n!C_n$ [10].

Where,

$$Cn = \frac{1}{2n+1}\binom{2n+1}{n} \tag{1}$$

The floorplan problem is NP-hard, and the solution space is quite huge, it is imperative to use some kind of optimization technique, to optimize the desired parameter (area in this paper) efficiently. Many types of optimization techniques have been used by researchers without or with shape flexibility (hard modules [10] and soft modules [11]) like Genetic Algorithms mentioned in [12]-[14], Memetic algorithms [10][15], most widely used Simulated Annealing (SA) method [16][17], Particle Swarm Optimization (PSO) [18], PSO with chaos strategy [19], Clonal selection algorithm [20], hybrid forms with Genetic and Particle Swarm Optimization [21], hybrid forms with genetic algorithms and an evolutionary algorithm [22] and many more. Research on soft modules benchmarks has also been commenced like 3D IC design using hybrid Multi-verse optimizer [23]. In the presented paper, the memetic algorithm has been used on hard modules in which a genetic algorithm has been used for global exploration and an efficient local search method for an optimized solution out of the searched solution. The results are checked on MCNC Benchmark circuits. In this paper, the terms operator and function are used interchangeably. Also, the terms tier and layer are used interchangeably.

II. Problem Statement

In the floorplanning problem, let P = {n_1, n_2, n_3, ….., n_n} be the n number of rectangular modules or nodes or blocks, where w_i and h_i be the width and height of n_i^{th} module. The objective of the floorplan problem in this paper is to optimize the floorplan area (A), where a_i is the area of the i^{th} module. All the modules in this paper are considered to be hard modules where the dimensions of the module cannot be changed but the module is free to rotate or move.

Another objective of the floorplan problem is to divide the floorplan into a number of tiers 't' or layers and the floorplan is divided uniformly between these layers in such a way that the minimum area can be achieved, where a minimum fixed outline (minimum bounding rectangle) can be drawn out of the maximum x and y dimension from all the layers. Overlapping between modules is strictly prohibited.

The ideal area (I.A.) of the floorplan problem is given by equation (2):

$$I.A. = (\textstyle\sum_0^n A(i))/t \tag{2}$$

Where,

$$A(i) = w(i) * h(i) \tag{3}$$

Hence,

$$I.A. = (\textstyle\sum_0^n w(i) * h(i))/t \tag{4}$$

Where, t: number of tiers.

The achieved floorplan area of the floorplan problem is given by the equation

$$F = maxx * maxy \tag{5}$$

Where,

maxx: maximum x-axis value from all the layers

maxy: maximum y-axis value from all the layers.

The fitness value of the floorplan defines how close the floorplan is to the ideal value and it is defined as:

$$Fit = I.A./F.A. \tag{6}$$

Where, $0 \le Fit \le 1$

Putting equations (4) and (5) in (6) gives:

$$Fit = (\textstyle\sum_0^n w(i) * h(i))/(t * maxx * maxy) \tag{7}$$

The dead space (D.A.) is the unoccupied area and may be considered and used as white space (useful space) in 3D ICs. Its value is given by equation (8):

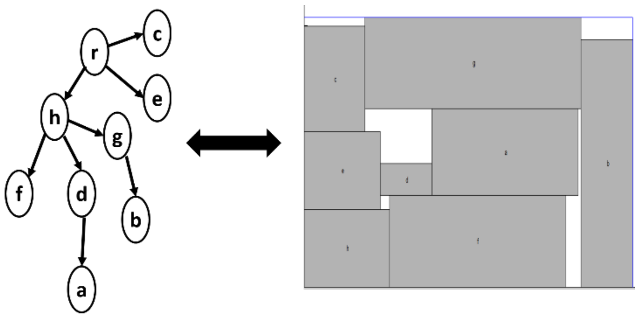
$$D.A. = 1 - Fit \tag{8}$$

Putting equation (7) in (8) gives:

$$D.A. = \frac{t*maxx*maxy - \sum_0^n w(i)*h(i)}{t*maxx*maxy} \tag{9}$$

III. Proposed Memetic Algorithm With O-Tree Representation

The presented paper has utilized one of the most important and efficient representations: O-Tree representation used for floorplan problems [5]. This representation is implemented by using the CBLSP methodology [9]. The 3D floorplan can be generated by dividing the floorplan into various tiers and applying the representation and optimization techniques on these layers as shown in figure 2.



(a) Tier 1

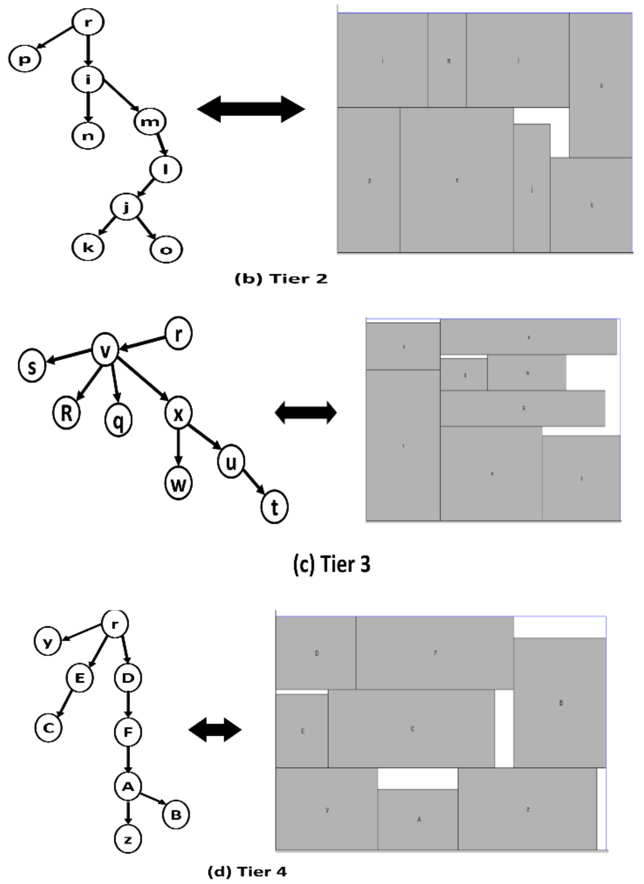


Fig. 2. The O-Tree representation in each tier and the equivalent placement of modules in a four-tier system.

A. Selecting Modules and Samples

In this subsection, the number (and identity) of modules are selected for a particular tier system in the 3D floorplan. In the presented algorithm, the number of modules is selected by calculating the summation of individual areas of the modules. If the summation comes near the acceptable limit or range of the ideal area, then that sample is selected for further operations. This is important since ideally, an area equal to the ideal area is desirable. The function continues to execute until the near-range summation area in each tier is achieved. During the execution of the algorithm, it has been found that in maximum cases for MCNC benchmarks, a tolerance of 5% area near the ideal area is enough. After this, the operations performed on these layers will remain the same in each tier system. A simple flowchart of the algorithm for selecting nodes in various tier systems is shown in figure 3.

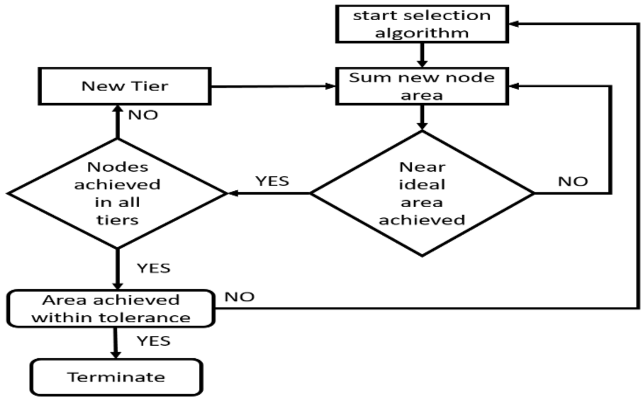


Fig. 3. A flow chart for selecting nodes in various tiers.

B. Crossover (Genetic Algorithm)

Crossover is the heart of the genetic optimization technique. In this operator, two parents are used to generate a child by sharing some attributes or characteristics from both parents. A particular branch with some nodes from the O-Tree representation of parent P1 is selected (some percentage attributes are taken). The selected nodes from parent P1 are deleted in Parent P2. The remaining nodes in both parents are combined to form a new child C1. The attribute sharing in the presented algorithm is kept between 25%-55%. It is not a surety that the crossover operator will always generate a better solution. If a worse child is generated than its parents then the child is rejected. A better child (better fitness) than its parents is kept. It has been found that for parents with weak fitness, the probability of generating a better child is high. For parents with strong fitness, the probability of generating a better solution is quite low. That's why, in the developed algorithm, the crossover step is strictly followed for weak parents, while for high fit parents, the crossover step is followed with some probability. This probability enhances the speed of the algorithm. The concept of crossover operator is shown in figure 4(a) where a particular sub-tree from Parent 'P1' with nodes n6, n7, n8, and n9 are selected and the selected nodes are deleted in parent P2. After the nodes are deleted, the selected sub-tree from parent P1 is connected to the leftover Parent 'P2' to generate a new tree, child 'C1'. It is clearly visible from figure 4 (a) that the generated child C1 holds the characteristic of both the parents 'P1' and 'P2'.

1) Parents Swapping: The swapping operator is implemented in the algorithm, to give equal opportunities to both the parents to act as parents 'P1' and 'P2'. In this operator both the parents are interchanged giving equal opportunities and searching for more relevant solution space. Results have shown the usefulness of this operator, giving optimized results. Out of the obtained results, the best result is kept.

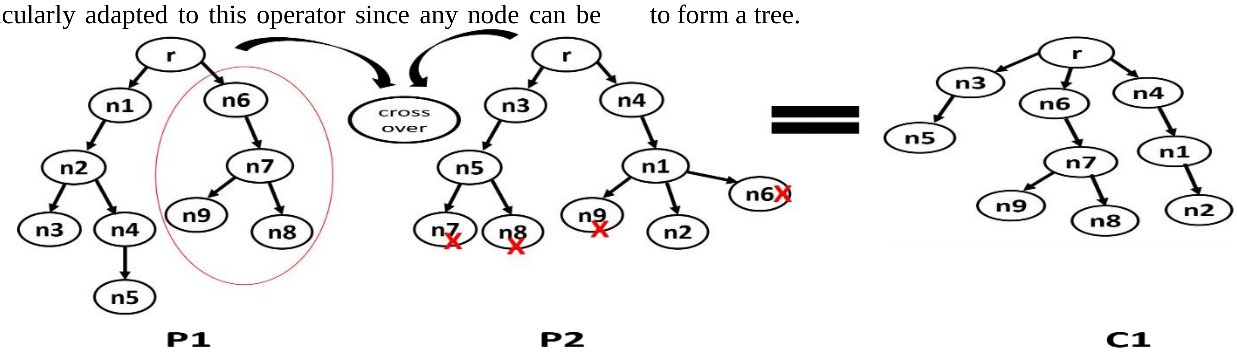
C. Mutation

In the developed tool, different types of mutation functions are implemented rather than simple mutation. It aids in optimizing local solutions from the global search space. Results have shown an immense improvement in the results by utilizing this operator. In the multi-variable problem, there can be many types of mutation that can be implemented as mentioned in the following sub-sections.

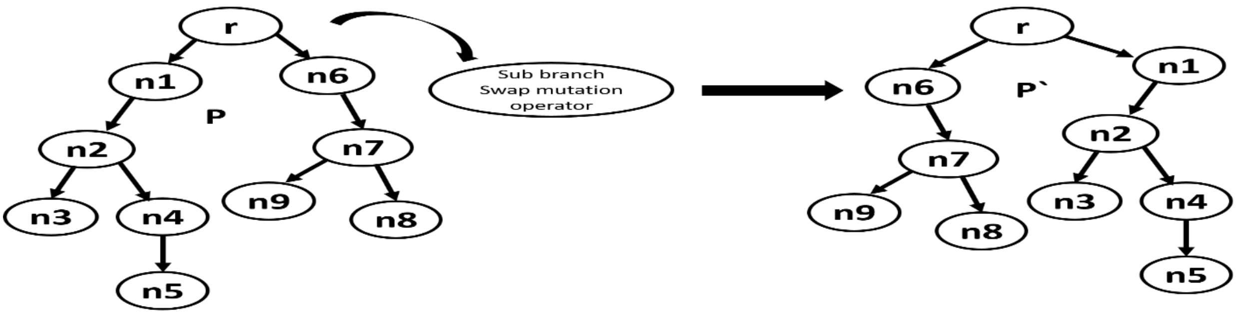
1) Sub Branch Swapping: In this type of operator, a branch of the O-Tree representation is swapped with another branch of the same representation. This changes the position of the blocks in the floorplan and may result in an optimized floorplan. Hence this operator helps in better searching of the particular solution space. If a better result is obtained then that solution is kept. The concept of the sub-branch swapping function is shown in figure 4(b). In this two sub-branches ({n1, n2, n3, n4, and n5} and {n6, n7, n8, and n9}) are selected randomly and exchanged with each other. The sub-branch may or may not be connected to the root branch.

2) Node Mutation: In this type of operator (or function), a particular node or a series of nodes are moved from its existing position to some other node position. To implement this, two nodes are selected randomly, and then the first selected node is connected to another selected node. This type of function may help in placing appropriate modules at appropriate positions resulting in a better floorplan. In the implemented algorithm, a maximum of two node mutations are implemented. This helped in a better solution in less amount of time. O-Tree representation is particularly adapted to this operator since any node can be connected to any other node, which is not possible in B*-Tree representation, where a maximum of two nodes can be connected to another node. The concept of node mutation is shown in figure 4(c). In figure 4(c), single and double node mutation is shown. In single node mutation, two nodes 'n1' and 'n8' are selected randomly. Where the first selected node 'n1' is connected to the other selected node 'n8'. In double node mutation, two connected nodes 'n4' and 'n5' are joined to another randomly selected node 'n9'.

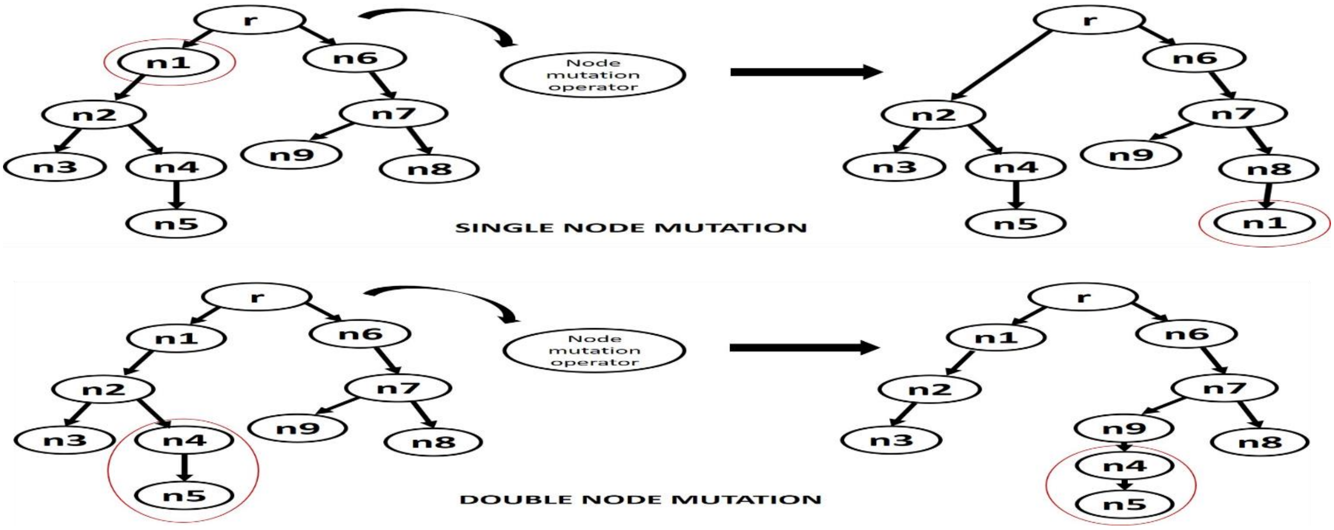
3) Swap Node Mutation: In this type of node mutation, two nodes exchange their position. In this function, two nodes are selected randomly and then these nodes are exchanged with each other. As in node mutation, this function may help in placing appropriate modules at an appropriate position, which helps in optimizing the floorplan problem locally for a given solution space. Swap node mutation can be a single or series of mutations. The concept of swap node mutation is shown in figure 4(d). In this two nodes (node 'n1' and node 'n7') are selected randomly and then swapped with each other to form a tree.



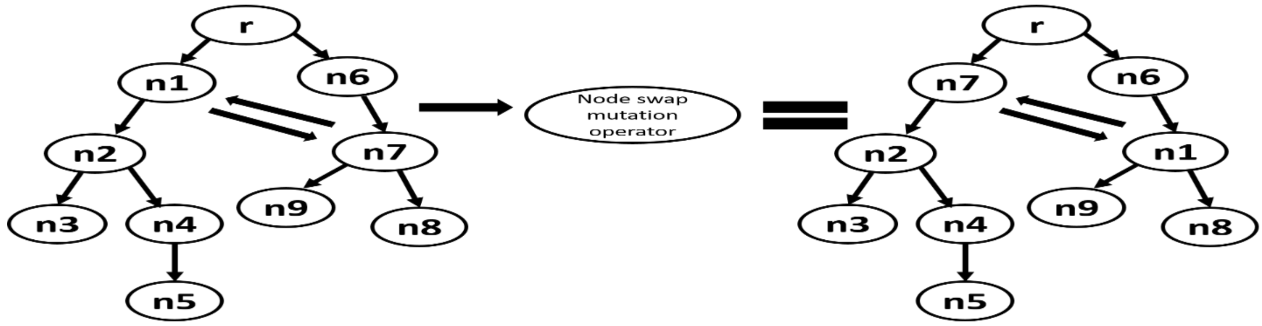
(a)



(b)



(c)



(d)

Fig. 4. Summary of the proposed optimization technique with steps of (a) crossover, (b) Sub branch mutation operator, (c) Single and double node mutation function, (d) Node swap mutation function.

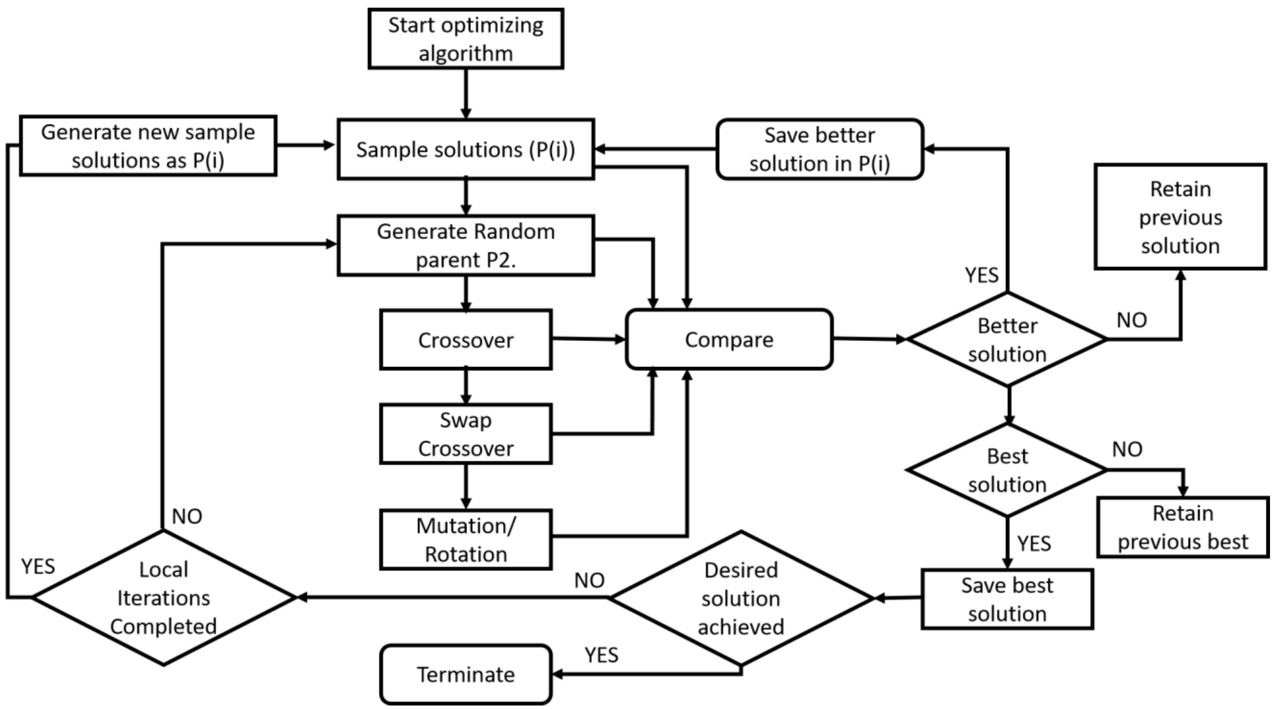


Fig. 5. Flow chart of the outline of the algorithm.

4) Code Mutation: In this type of mutation, perturbation of code is done. For a given solution, (code and sequence of modules), a change of code is implemented. The position with code '0' is chosen randomly, similarly another position with code '1' is chosen. Then these codes are inverted (the code '1' is changed to code '0' and code '0' is changed to code '1'). While choosing the different positions of the code, CBLSP methodology [9] is taken into consideration, where at any index of the code, the number of zeros should always be greater than or equal to the number of ones in the right direction. Similar to the general mutation function, this function helps too in generating an optimized floorplan and it can be a single mutation or series of mutations. After this operation, always the best solution is kept, rejecting others.

5) Rotation: This is another mutation in which, a module is rotated by 90º or 270º. Rotation can be done simply by changing the widths and height of the module. Similar to all other mutation types, the rotation function can be performed on a single module or a series of modules. If a better solution is obtained by performing this function, then that solution is kept, rejecting others.

D. Generation of New Samples

If the desired solution is not achieved within a number of iterations. Then a new solution space is generated (as new Parents 'P1'). In the developed tool, a percentage of the previous best solution space is kept in the new space while the worse solutions are removed. The removed solutions are replaced by newly generated solutions, mixed (crossover) solutions out of two solutions, mutated and rotated versions of solutions. This helps in exploring new solution space which increases the probability of generating the best fit (optimized) solution. The concise outline of the mentioned steps of the algorithm is shown in figure 5. The steps described in the flowchart are already summarised in the previous subsections.

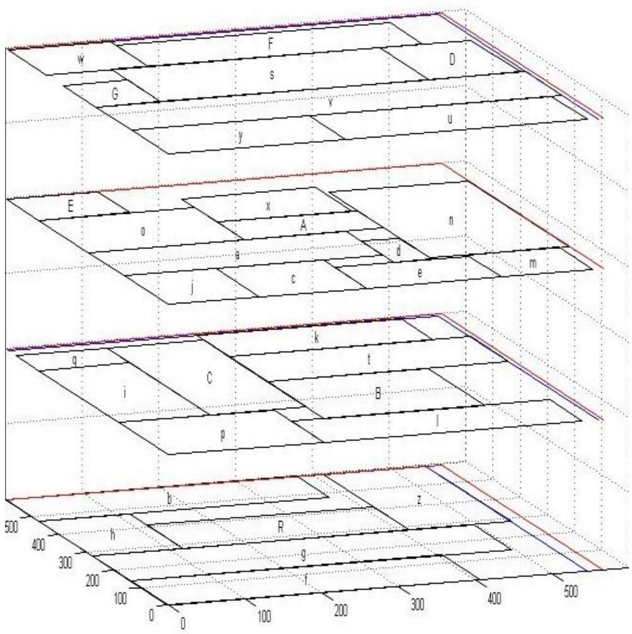
IV. Results and Discussion

The code for the generated algorithm is implemented in the C++ language. The attribute sharing in the crossover part for Parent 'P1' is kept between 25%-55%. The algorithm is divided into various steps in such a way that if a certain condition is met then the further redundant steps are removed, making the algorithm faster to execute.

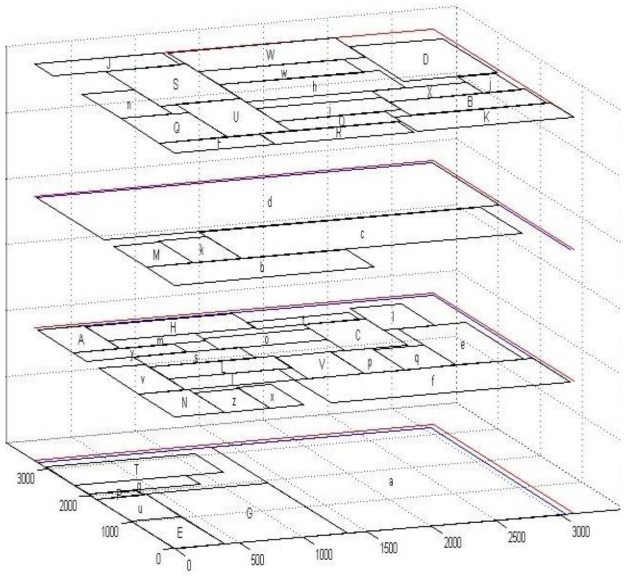
TABLE I. MCNC Benchmarks Characteristics.

Benchmarks	APTE	XEROX	HP	AMI33	AMI49
Number of modules	9	10	11	33	49
Nets	97	203	83	123	408
Pins	287	698	309	522	953
I/O pads	73	107	43	42	24
Ideal area for single tier (mm^2)	46.56	19.35	8.83	1.16	35.45
Ideal area for a two-tier system	23.28	9.675	4.415	0.58	17.725
Ideal area for a three-tier system	15.52	6.45	2.94	0.386	11.81
Ideal area for four-tier system	11.64	4.83	2.20	0.29	8.86

The results were checked and compared with MCNC benchmark circuits. MCNC stands for Microelectronics Center of North Carolina. In 1987, Bryan Preas and Ken Roberts at Xerox PARC (Palo Alto Research Center) developed this language. This benchmark is used by many researchers as a standard to check their results on the developed algorithm or representation. The files of these benchmark circuits are in YAL (Yet Another Language) format. MCNC Benchmark characteristics are shown in table I.



(a) AMI33



(b) AMI49

Fig. 6. Simulated graphical result of the AMI33 and AMI49 benchmark circuits in 3D IC format with a four-tier system.

The simulated graphical results of AMI33 and AMI49 for tier 4 are shown in figure 6. The blue line beside the axis line in the graphical representation shows the local minimum bounding rectangle of the particular tier in the circuit (for that particular tier) and the red line beside the axis line in the figure shows the overall minimum bounding rectangle of the floorplan area. The best result for the AMI33 benchmark circuit in figure 6 has used 6, 8, 11, and 8 different modules for tiers 1, 2, 3, and 4 respectively. The best result for the AMI49 benchmark circuit has used 7, 20, 5, and 17 different modules with different areas respectively for tiers 1, 2, 3, and 4.

Table II shows the comparison of the results of various research articles with our proposed algorithm result. Mentioned area in table II is in μm^2. The best results in the area from previous literature have been taken for comparison. Results are compared with previous proposed algorithmic tools like TCG-based 2D array [24], CBA [24], MILP formulation [25], B*-Tree with SA [26], and Genetic based algorithm [27]. Results show that the proposed algorithm has generated the best results in each benchmark and each tier of the floorplan. For AMI33, the proposed algorithm has shown 40.3%, 48.8%, and 2% improvement (reduction) for tiers 2, 3, and 4 respectively in the floorplan area. For the AMI49 benchmark circuit, the proposed algorithm has shown an 8.7% improvement in tier 4. APTE benchmark has shown 25.5%, 16.8%, and 21.8% reduction in area for tiers 2, 3, and 4 respectively. XEROX benchmark circuit has shown 22.9%, 25.7%, and 33.7% improvement for tiers 2, 3, and 4 respectively. Hence the proposed algorithm is very effective in generating an optimized floorplan. The proposed algorithm has generated results in very less time. Though the time to implement the algorithm depends heavily on the machine on which the algorithm is executed.

V. Conclusion and Future Work

With the increasing miniaturization of electronic circuits and better fabrication techniques, researchers are looking toward the third dimension for designing circuits in 3D ICs. The developed algorithm aims at minimizing the size of the chip. For this, the floorplan has been divided into various layers (tiers) in 3D technology. It has used O-Tree representation with CBLSP methodology at each layer. Since the VLSI floorplan is an NP-Hard problem, it has huge solution space and requires some optimization techniques. The presented algorithm has used the Memetic Algorithm in which the Genetic Algorithm is used for global exploration and local optimization technique for the desired result out of the solution search space. The obtained results through this optimization technique are checked and compared with MCNC Benchmark circuits. It has been found that the proposed algorithm has obtained the best results with a range of 2% to 48.8% reduction in area in all the tiers and benchmark circuits. The proposed result can be an asset for the manufacturing and designing VLSI industry with a wide range of applications including aviation, neurosciences, automobiles, space, genetics and circuit mechanism, etc.

There are many representations with numerous optimization techniques that have been implemented in the 2D Floorplan problem as mentioned in the introduction section. There is an enormous possibility of implementing and checking the 2D techniques to 3D since the implemented research in 3D is quite limited. The proposed work (in 3D in this paper) can be further enhanced for the optimization of the variables: wirelength, temperature, and a combination of any two or more variables. The dead space in this work can be used effectively as white spaces with the introduction of wire and thermal vias. The work can also be enhanced for efficient clock gating and power supply of the circuit in 3D. The proposed work may also be tested for better performance by changing the parameters sharing, mutation, and rotation of the modules.

Acknowledgment

The authors would like to extend gratitude to Dean RIC, I. K. Gujral Punjab Technical University, Kapurthala for support in the completion of this research work.

TABLE II. COMPARISON OF THE PROPOSED ALGORITHM RESULTS (AREA IN UM^2) WITH PREVIOUS LITERATURE (NR- NOT REPORTED).

Reference	Number of tiers	AMI33	AMI49	APTE	XEROX	HP
TCG-based 2D array	2	NR	NR	NR	NR	NR
	3	NR	NR	NR	NR	NR
	4	3.52E+05	1.49E+07	NR	NR	NR
CBA	2	NR	NR	NR	NR	NR
	3	NR	NR	NR	NR	NR
	4	3.44E+05	1.27E+07	NR	NR	NR
MILP formulation	2	NR	NR	NR	NR	NR
	3	NR	NR	NR	NR	NR
	4	3.58E+05	1.22E+07	NR	NR	NR
B*-Tree with SA	2	10.9E+06	NR	32.07E+06	13.62E+06	NR
	3	8.44E+05	NR	21.60E+06	9.48E+06	NR
	4	6.36E+05	NR	15.5E+06	8.04E+06	NR
GA based algorithm	2	NR	NR	NR	NR	NR
	3	NR	NR	NR	NR	NR
	4	3.64E+05	1.148E+07	NR	NR	NR
	3	NR	NR	NR	NR	NR
	4	3.53E+05	1.490E+07	NR	NR	NR
Proposed algorithm (Improvement)	2	6.50E+05 **(40.3%)**	1.96E+07	23.87E+06 **(25.5%)**	10.49E+06 **(22.9%)**	4.85E+06
	3	4.32E+05 **(48.8%)**	1.36E+07	17.96E+06 **(16.8%)**	7.04E+06 **(25.7%)**	3.43E+06
	4	3.37E+05 **(2%)**	1.04E+07 **(8.7%)**	12.12E+06 **(21.8)**	5.33E+06/ **(33.7%)**	2.90E+06

REFERENCES

[1] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design," in *Proceedings of 23rd ACM/IEEE Design Automation Conference*, pp. 101-107, 1986. doi: 10.1109/DAC.1986.1586075.

[2] H. Onodera, Yo Taniguchi and K. Tamaru, "Branch-and-bound placement for building block layout," in *Proceedings of 28th ACM/IEEE Design Automation Conference*, pp. 433-439, 1991. doi: 10.1145/127601.127708.

[3] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, " Rectangle-packing-based module placement," in *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, pp. 472-479, 1995. doi: 10.1109/ICCAD.1995.480159.

[4] M. Kang and W. W. . -M. Dai, "General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure," in *Proceedings of ASP-DAC '97: Asia and South Pacific Design Automation Conference*, pp. 265-270, 1997. doi: 10.1109/ASPDAC.1997.600145.

[5] K Pei-Ning Guo, Chung-Kuan Cheng and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," in *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, pp. 268-273, 1999. doi: 10.1109/DAC.1999.781324.

[6] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.K. Cheng and J. Gu, "Corner block list: an effective and efficient topological representation of non-slicing floorplan," in *Proceedings IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140)*, pp. 8-12, 2000. doi: 10.1109/ICCAD.2000.896442.

[7] Yun-Chih Chang, Yao-Wen Chang, Guang-Ming Wu and Shu-Wei Wu, "B*-trees: a new representation for non-slicing floorplans," in *Proceedings 37th Design Automation Conference*, pp. 458-463, 2000. doi: 10.1145/337292.337541.

[8] J. Shanthi, D.G. Rani and S. Rajaram, "An Enhanced Memetic Algorithm using SKB tree representation for fixed-outline and temperature driven non-slicing floorplanning," *Integration*. Vol. 86, pp. 84-97. Apr. 2022. https://doi.org/10.1016/j.vlsi.2022.04.001.

[9] R. Gupta, S.S. Gill, "Adaptive Memetic Algorithm on novel CBLSP algorithm for O-Tree implementation," In *Advances in VLSI and Embedded Systems. Lecture Notes in Electrical Engineering*, 2021.

[10] M. Tang and X. Yao, "A Memetic Algorithm for VLSI Floorplanning," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 37, No. 1, pp. 62-69, Feb. 2007, doi: 10.1109/TSMCB.2006.883268.

[11] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," in *Proceedings of International Conference on Computer Aided Design*, pp. 484-491, 1996.doi: 10.1109/ICCAD.1996.569870.

[12] C. L. Valenzuela and P. Y. Wang, "VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions," in *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 4, pp. 390-401, Aug. 2002, doi: 10.1109/TEVC.2002.802872.

[13] H. A. Rahim, A. A. H. A. Rahman, R. B. Ahmad, W. N. S. F. W. Ariffin and M. I. Ahmad, "The Performance Study of Two Genetic Algorithm Approaches for VLSI Macro-Cell Layout Area Optimization," in *Proceedings 2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pp. 207-212, 2008. doi: 10.1109/AMS.2008.117.

[14] J. Chen and W. Zhu, "A hybrid genetic algorithm for VLSI floorplanning," in *Proceedings 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, pp. 128-132, 2010. doi: 10.1109/ICICISYS.2010.5658785.

[15] J. Chen, Y. Liu, Z. Zhu and W. Zhu, "An adaptive hybrid memetic algorithm for thermal-aware non-slicing VLSI floorplanning," *Integration*. Vol. 58, pp. 245-252, 2017. https://doi.org/10.1016/j.vlsi.2017.03.006.

[16] J. Chen, W. Zhu and M. M. Ali, "A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 41, No. 4, pp. 544-553, July 2011, doi: 10.1109/TSMCC.2010.2066560.

[17] S. Anand, S. Saravanasankar and P. Subbaraj, "Customized simulated annealing based decision algorithms for combinatorial optimization in VLSI floorplanning problem," *Computational Optimization and Applications*, Vol. 52, No. 3, pp. 245-252, 2012.

[18] G. Chen, W. Guo and Y. Chen, "A PSO-based intelligent decision algorithm for VLSI floorplanning," *Soft Computing*, Vol. 14, No. 12, pp. 1329-1337, 2009. doi: 10.1109/TVLSI.2004.840760.

[19] Y. Ye, X. Yin, Z. Chen, Z. Hong, X. Fan and C. Dong, "A novel method on discrete particle swarm optimization for fixed-outline floorplanning," *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*, pp. 591-595, 2020. doi: 10.1109/ICAIIS49377.2020.9194920.

[20] D. M. Abdullah, W. M. Abdullah, N. M. Babu, M. M. I. Bhuiyan, K. M. Nabi and M. S. Rahman, "VLSI floorplanning design using clonal selection algorithm," in *Proceedings 2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, pp. 1-6, 2013. doi: 10.1109/ICIEV.2013.6572664.

[21] S.J. Basha, B.A. Raheem, "VLSI Floorplanning Using Nature-Inspired Hybrid Optimization Algorithm," In ICDSMLA 2019, pp. 1959-1967, Springer, Singapore, 2020. https://doi.org/10.1007/978-981-15-1420-3_200.

[22] V.V. Ignatyev, A.V. Kovalev, O.B. Spiridonov, V.M. Kureychik, A.S. Ignatyeva and I.B. Safronenkova, "Hybrid genetic-paired-permutation algorithm for improved VLSI placement," *ETRI Journal. 2021*. Apr; Vol. 43, No. 2, pp. 260-271 Apr. 2021. https://doi.org/10.4218/etrij.2019-0412.

[23] S. Pothiraj, J. Kadambarajan and P. Kadarkarai, "Floor Planning of 3D IC Design Using Hybrid Multi-verse Optimizer," *Wireless Pers Commun* Vol 118, pp. 3007–3023, 2021. https://doi.org/10.1007/s11277-021-08166-z.

[24] J. Cong, Jie Wei and Yan Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *Proceedings IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pp. 306-313, 2004. doi: 10.1109/ICCAD.2004.1382591.

[25] X. Li, Y. Ma, X. Hong and S. Dong, "Thermal-aware incremental floorplanning for 3D ICs," in *Proceedings 7th International Conference on ASIC*, pp. 1092-1095, 2007. doi: 10.1109/ICASIC.2007.4415823.

[26] P. Falkenstern, Y. Xie, Yao-Wen Chang and Y. Wang, "Three-dimensional integrated circuits (3D IC) Floorplan and Power/Ground Network Co-synthesis," in *Proceedings 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 169-174, 2010. doi: 10.1109/ASPDAC.2010.5419899.

[27] F. Frantz, L. Labrak and I. O'Connor, "3D-IC floorplanning: Applying meta-optimization to improve performance," in *Proceedings 2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, pp. 404-409, 2011. doi: 10.1109/VLSISoC.2011.6081618.