International J.Math. Combin. Vol.3 (2010), 69-84

Combinatorial Optimization in VLSI Hypergraph Partitioning using Taguchi Methods

P.Subbaraj and S.Saravanasankar

Kalasalingam University, India

S.Anand

National Centre for Advanced Research in Discrete Mathematics (n-CARDMATH)

Kalasalingam University, India

Email: subbaraj_potti@yahoo.com, ssaravanasankar@yahoo.co.in, anandmay22@gmail.com

Abstract: This work addresses the methods to solve Very Large Scale Integration (VLSI) circuit partitioning problem with dual objectives, viz., 1. Minimizing the number of interconnection between partitions, that is, the cut size of the circuit and 2. Balancing the area occupied by the partitions. In this work an efficient hybrid Genetic Algorithm (GA) incorporating the Taguchi method as a local search mechanism has been developed to solve both bipartitioning and recursive partitioning problems in VLSI design process. The systematic reasoning ability of the Taguchi method incorporated after the crossover operation of GA, has improved the searching ability of GA. The proposed Hybrid Taguchi Genetic Algorithm (HTGA) has been tested with fifteen popular bench mark circuits of ISCAS 89 (International Symposium on Circuit and Systems-89). The results of experiments conducted, have proved that HTGA is able to converge faster in reaching the nearer-to-optimal solutions. The performance of the proposed HTGA is compared with that of the standard GA and Tabu Search method reported in the literature. It is found that the proposed HTGA is superior and consistent both in terms of number of iterations required to reach nearer-to-optimal solution and also the solution quality.

Key Words: VLSI, partitioning, genetic algorithm, Taguchi method, cut size, multi-partitioning.

AMS(2000): 49J35

§1. Introduction

During the Very Large Scale Integration (VLSI) design process, the complex circuit comprising of elements like gates, buffers, Input/Output ports which are inter connected by wires is divided into subsets, that is, modules [10,16] as the first step. This partitioning of the circuit into smaller modules is essential to reduce the problem complexity of the VLSI physical design

[1]Received July 21, 2010. Accepted September 14, 2010.

problem. Proper partitioning of a VLSI circuit will result in minimum total area occupied by all the elements of the circuit, and reduction in the total length of interconnecting wires between the elements, which will in turn minimize the power dissipation and time delay during its operation. To achieve these objectives of VLSI design problem, the complex VLSI circuit should be partitioned into smaller sub modules such that the number of wires passing between the elements of different modules is kept minimum. For a particular partition, the sum total of number of wires passing between the modules is known as cutsize of the partition. A partition with modules occupying equal area will largely help in the later part of the VLSI design process namely floorplanning, placement and routing. Hence, partitioning of VLSI circuit should be done in such a way that, all the modules occupy more or less equal area or in other words the uneven distribution of area among the modules, that is, imbalance in area should be kept minimum. Hence in this work, both these objectives (i) minimizing the cutsize and (ii) minimizing the area imbalance among the modules are considered for solving the VLSI partitioning problem.

VLSI circuit partitioning is proved to be an intractable problem [14] and only satisfactory solutions to the different problem instances are being generated by designing suitable metaheuristic algorithms. In this research work, an attempt is made to design a suitable metaheuristic algorithm capable of producing consistent solution with lesser number of iterations for a wider range of VLSI circuit problem.

§2. Literature survey

B.W.Kernighan and S.Lin proposed the group migration algorithm (KL algorithm) [12] for graph partitioning problem which through the years of use has been proved to be very efficient. However KL algorithm is designed only for bipartitioning the given circuit. C.M.Fiduccia and R.M.Mattheyses (FM) improved the KL algorithm by introducing an elegant bucket sorting technique [7]. However, FM algorithm was able to provide satisfactory solutions only for smaller to medium size problems and also only for bipartitioning the circuit. Later Cong.J (1994) developed k-way net based multi way partitioning algorithm to produce better quality solutions than the FM algorithm but only for smaller size problems. Mean time hMetis [24] and other Multilevel Clustering algorithms (MLC) were developed [8] based on the flat partitioning methodology with an aim of further minimizing the cutsize. Later, the Multilevel Partitioning algorithm (MLP) that is also based on the flat partitioning methodology, was developed by Jong-Sheng (2003) and its performance surpassed the result produced by hMetis and MLC in terms of minimal cutsize. However it is proved that flat multiway partitioning approach could produce better quality results for smaller size integrated circuits [17,18], and due to the space complexity ($O(N.K\ (K\text{-}1))$ where N denotes the number of cells) and poor flexibility, the approach is less efficient with larger size integrated circuits. The method of recursive partitioning evolved by Aeribi.S [3] is found to be performing better than the flat partitioning methodology interms of solution quality but at the cost of additional computational load. Sadiq.M.Sait developed metaheuristic algorithms [16] based on Genetic Algorithm (GA) and Tabu search (TS) to address relatively larger size problems and with multiple objectives. In his work he has proved that though GA is able to produce quality solutions for smaller size circuits and Tabu search outperforms GA in terms of both quality of the solution and execution time even for the larger circuits.

In this work, with an emphasis on solution quality, research focus is retained to improve upon the recursive partitioning methodology, inspite of its heavy computational requirement compared to the flat partitioning methodology. Also to address the problem complexity of VLSI multi partitioning problem, which is NP-hard, an attempt is made to develop a metaheuristic algorithm based on the robust and versatile tool, GA. To overcome the inherent scalability issue with the GA, the Taguchi method, a robust design approach is incorporated in the genetic search process.

§3. Problem formulation

Any VLSI circuit consisting of more than one component or element (that is either a gate or flip flop or buffer) can be represented in the form of a hyper graph $H(V, E)$. $V = \{v_1, v_2, v_3 \cdots v_n\}$ is the set of nodes representing the elements used in the circuit and $E = \{e_1, e_2, e_3 \cdots e_n\}$ is the set of edges representing all the required connections between the elements. The aim of the work is to split the given hyper graph into required number of partitions with minimum number of inter connections between the partitions (namely the cutsize) and also with minimal area imbalance between the modules, that is, the uneven distribution of area among the partitions. An attempt to minimize the number of interconnecting wires between two modules by placing the elements associated in the interconnectivity, together in one module will result in increase in area imbalance between the two modules, and vice versa. Hence in order to achieve the above said two contradicting objectives concurrently, the following combined objective function is constructed.

The Combined Objective Function (COF):

$$COF = Minimize\ [(\alpha_1 * F_1) + (\alpha_2 * F_2)] \tag{1}$$

where,
F_1 = Cutsize (given in (2))
F_2 = Area imbalance between the circuits (given in (3))
α_1 = Weightage factor assigned to the cutsize
α_2 = Weightage factor assigned to the area imbalance
The function [23] for cutsize (F_1) is:

$$F_1 = \sum_{\forall r \in E} \left(\sum_{i=1}^{(|Q_r|-1)} (-1)^{i+1} c_i^{Q_r} - 2F \prod_{j=1}^{|Q_r|} x_j \right) \tag{2}$$

where,
Q_r= Set of assignment variables for all non Input/Output components on net (edges) r

$$F = \begin{cases} 1\ if\ |Q_r|\ is\ even \\ 0\ otherwise \end{cases}$$

E = Set of edges
${C_i}^{Q_r}$ = Combinations of the set Q_r taken i at a time
x_j = Set of nodes
The function for area imbalance (F_2) is:

$$F_2 = \beta_1 - \beta_2 \tag{3}$$

where,
$\beta_1 = max \{ |P| : P \text{ is a partition } \}$
$\beta_2 = min \{ |P| : P \text{ is a partition } \}$
$|P|$ = Number of elements in a partition

§4. Proposed methodology

A GA based heuristic namely Hybrid Taguchi Genetic Algorithm (HTGA) is proposed in this work, to solve the VLSI circuit partitioning problem with dual objectives of minimizing the cutsize and minimizing the area imbalance among the partitions. The proposed algorithm is tested with fifteen popular bench mark circuits of ISCAS89, and its performance is compared with that of the other metaheuristics reported in the literature.

4.1 Genetic Algorithm

Genetic algorithm operates on the principle of *survival-of-the-fittest*, where weak individuals die, while stronger ones survive and bear many offspring and breed children, which often inherit qualities that are, in many cases superior to their parent's qualities [14]. GA begins with a population offspring (individuals- representing the design/decision variables) created randomly. Thereafter, each string in the population is evaluated to find its fitness value (that is, the objective function value of the given optimization problem). The operators *Selection, Crossover and Mutation* are used to create a new and better population. The new population is further evaluated for the fitness values and tested for termination. If the termination criteria are not met, the population is interactively operated by the above genetic operators and evaluated. One cycle of these genetic operations and the evaluation procedure is known as a *generation* in GA terminology. The generation cycle is continued until the termination criterion is met.

4.2 Taguchi Method

Taguchi method is a robust design approach, which uses many ideas from statistical experimental design for evaluating and implementing improvements in products, processes and equipment [21,9]. The fundamental principle of Taguchi method is to improve the quality of a product by minimizing the effect of the causes of variation without eliminating the inevitable causes.

The two major tools used in the Taguchi method are:

1. *Orthogonal arrays (OA) which are used to study many design parameters simultaneously,*
2. *Signal-to-Noise Ratio (SNR) which measures quality.*

For instance, let there be an optimization problem whose solution is influenced by, say seven factors and each of these factors can be at any of the two levels. If the objective is to find a suitable level for each factor to find an optimal solution, then the total number of possible experiments is 2^7 to find the optimal solution. An orthogonal array (OA), an example shown in Table 1, represents a set of recommended limited number of experiments, (eight for the example shown in Table 1, needed to find a suitable level for each factor to achieve an optimal solution at a faster rate. Thus, with the help of only these 8 experiments out of a total 2^7 possible experiments, the best solution can be found with each factor being at a suitable level. The orthogonal arrays are represented as $L_n(x^{n-1})$, where $n = 2^k$ is the number of experimental runs, k is a positive integer, x is the number of levels for each factor and $n - 1$ is the number of columns in an orthogonal array. The example OA is shown in the Table 1, is of $L_8(2^7)$ type.

The second tool of Taguchi method, the SNR, is used to find which level is suitable for each factor; SNR calculation is discussed with an example in Section **??**. In communication engineering parlance, the Signal to Noise Ratio means the measure of signal quality, which corresponds to the solution quality in Taguchi method. While conducting each experiment as per the orthogonal array, the objective function value is computed, and the effect of each of the two levels on each factor in contributing to the objective function value is computed. A level to a particular factor, which gives the maximum effect in contribution to the objective function value, is optimal for the concerned factor. As the effect is maximum for this level, it is said to have maximum influence or the maximum *Signal to Noise Ratio* (SNR) and so considered as optimal level for the factor. With the conduct of all the experiments as per the orthogonal array, the solution obtained with optimal level for each factor, is the optimum solution for the given optimization problem.

4.3 Hybrid Taguchi Genetic Algorithm (HTGA)

In the proposed Hybrid Taguchi Genetic Algorithm (HTGA) to solve the VLSI partitioning problem, the Taguchi method is embedded within GA, between the crossover and mutation operations, to improve all the solutions of the intermediate population obtained after the crossover operation and before subjected to the subsequent mutation operation.

The proposed HTGA is designed to generate multi-partitioning solutions for larger size VLSI problems through the recursive approach, recomented by Areibi.S [3]. The adapted recursive approach applies bipartitioning recursively until the desired number of partition is obtained, which is illustrated in the example shown in Fig. 1, where a single VLSI circuit is recursively partitioned into eight partitions.

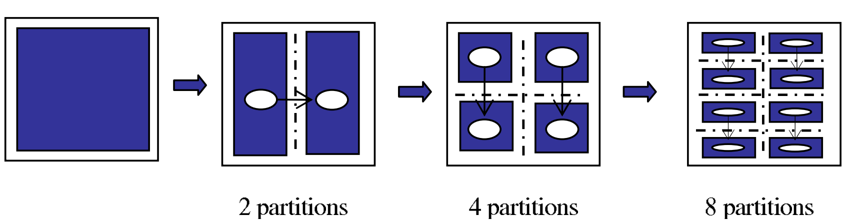

Figure 1: Recursive partitioning of a VLSI circuit

In HTGA, genotype representation is used to code a feasible solution as a chromosome [4,14]. The zeros and ones in a chromosome represents either of the two partitions they belong to. In case of multiple partitions through recursive partitioning, each of the divided chromosomes representing each partition will have zeros and ones representing either of the two sub partitions.

A bipartition solution of a VLSI circuit having components v_1, v_2, v_3, v_4, v_5 and v_6 shown in the Fig. 2 is encoded as a solution chromosome as shown in Fig. 3. The digit one represents that the element is present in the partition P_1 otherwise in P_2.

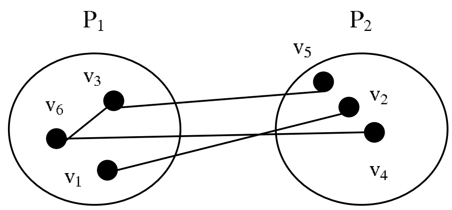

Figure 2: A bipartitioning solution of the example VLSI circuit

v_1	v_2	v_3	v_4	v_5	v_6
1	0	1	0	0	1

Figure 3: Chromosome representation of bipartition solution

When the bipartition solution shown in Fig.3 is further partitioned through recursive method, that is, when P_1 is partitioned into $P_{1(a)}$ and $P_{1(b)}$ and P_2 is partitioned into $P_{2(a)}$ and $P_{2(b)}$, a sample solution shown in Fig.4 is encoded as a solution chromosome as shown in Fig.5.

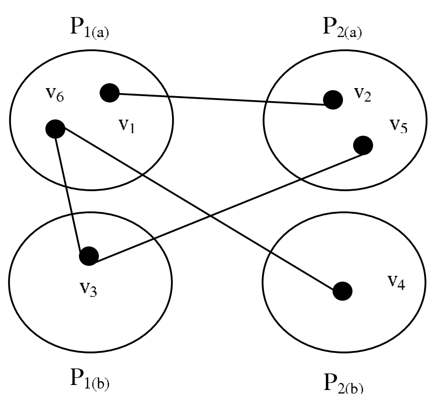

Figure 4: A recursive partitioning solution of the example VLSI circuit

In the proposed HTGA, the random initial population of partitioning solutions is subjected to selection and crossover operations. The resultant intermediate population obtained through the cross over operations is fed to the local search mechanism, Taguchi method module of the

v_1	v_3	v_6	v_2	v_4	v_5
1	0	1	1	0	1

Figure 5: Chromosome representation of the solution with four partitions

HTGA. This phase of the HTGA creates a new improved intermediate population of same size with each solution entirely different from the initial solutions of the intermediate population resulted out of crossover operation of GA.

The algorithm shows the Taguchi phase in HTGA.

Algorithm

Encode the random initial population of solution
Do while the termination criteria is not met
  Step 1: Perform Reproduction
  Step 2: Perform Crossover
  Step 3: Taguchi Method
    a: Select a suitable orthogonal array
    Do while the size of the population is reached
      Do while an improved solution is found
        Step b: Random selection of pair of chromosome.
        Step c: Calculate SNRs.
          Compute Effect of Factors.
          Select the optimal bit
        Step d: Construct new chromosome
      End Do
    End Do
  Step 4: Perform Mutation
End Do
Decode the best solution in the final population to get the optimal partition.

In each iteration of this phase, a pair of chromosomes, say X and Y are selected at random from the intermediate population and a better chromosome Z is evolved by choosing each gene either from chromosome X (level 1) or from chromosome Y (level 2). The Taguchi method of producing a better chromosome Z from a randomly chosen two chromosomes X and Y is illustrated in Table 2. Selection of suitable level is done by conducting eight experiments as per the example orthogonal array, shown in Table 1. For each experiment the functional value which is *COF* of experimental chromosome is computed. As the problem is minimization problem, the signal to noise ratio, SNR (η_i) for each experiment i is computed as a reciprocal of *COF* value of the experimental chromosome. Having calculated the SNR value for all the experiments, for each gene, the effect of choosing from level 1 (chromosome X) or level 2 (chromosome Y) chromosome is computed as equations 4 and 5.

$$Ef_1 = \sum_{i=1}^{n} SNR(\eta_i),\ when\ gene\ i\ is\ belongs\ to\ level\ 1 \tag{4}$$

$$Ef_2 = \sum_{i=1}^{n} SNR(\eta_i),\ \textit{when gene i is belongs to level}\ 2 \quad (5)$$

The gene is selected from the level for which the effect of factor Ef_i is maximum and the improved chromosome Z is thus constructed with all such selected genes in their respective positions.

The above said iteration is repeated by selecting another pair of chromosomes from the intermediate population and a new chromosome is created. The procedure is repeated till the new intermediate population of required size is created. This improved intermediate population is fed to the subsequent mutation operator of generation cycle of GA. The generation cycle of HTGA is repeated till the termination criterion is met.

§5. Results and discussions

The proposed algorithm, HTGA was coded in C++ and experiments were conducted in an IBM Pentium D PC with 3.20 GHz Processor. The HTGA was tested with fifteen number of ISCAS89 (International Symposium of Circuit And Systems) benchmark circuits. The details of the benchmarks are shown in Table 3. To measure the effect of Taguchi method in the proposed HTGA, the performance of HTGA is compared with that of the standard template of GA, that is, a genetic algorithm without the hybridization of Taguchi method. To make the comparison on a common platform the standard GA is also coded in C++, run on the same machine and tested with the same benchmark circuits.

In the proposed HTGA tournament selection is used for reproduction operation, Single cut point crossover is used in the crossover operation and Flap bit mutation is used for mutation operation. The parameters used in HTGA are as below.

1. Population Size = 20
2. Crossover probability (P_c) = 0.6
3. Mutation probability (P_m) = 0.01
4. Termination Criterion = A predefined number of iterations for a given circuit or a predefined satisfactory COF value, whichever occurs first.
5. Orthogonal array used in the Taguchi experimentation is $L_8(2^7)$.

The best values for the individual parameters are fixed by conducting trials and on satisfactory performance. The crossover probability P_c was varied from 0.4 to 0.9, and the GA is found able to converge faster with a crossover probability P_c of value 0.6. Similarly the mutation probability P_m was varied between 0.001 to 0.1 and the GA with the mutation probability P_m of value 0.01 is found able to retain more number of better solution than worse solution at the end of GA cycle.

For all the bench mark circuits taken in this work, the proposed algorithm HTGA is able to outperform the standard Genetic Algorithm both in bipartitioning application and so in recursive partitioning application, again both in terms of number of iterations required to reach a nearer-to-optimal solution and also in terms of the quality of the solution, that is the absolute value of COF. The results of this comparative study between GA and HTGA in bipartitioning and in recursive partitioning (four partitions) are shown in Tables 4 and 5 respectively.

It can be seen from both Tables 4 and 5, that the CPU time taken by HTGA is higher compared to the standard GA for smaller circuit, which may be attributed to the additional computational load required because of the Taguchi method of HTGA. However it can be also seen from these tables that, for larger circuits, the CPU time taken by HTGA is substantially lower than standard GA, which can be attributed to the efficiency of HTGA in reaching the solutions with lesser number of generation cycles.

It is observed that because of the Taguchi method after the crossover operation, HTGA is able to converge at a faster rate than that of the standard GA, which is explained with a sample benchmark problem S832 in Fig.6.

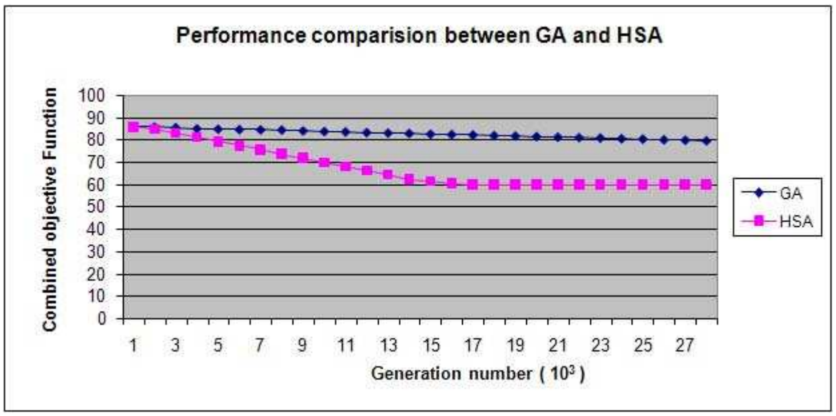

Figure 6: Convergence comparison between GA and HTGA for the benchmark problem S832

For each of the fifteen ISCAS89 benchmark circuits the experiment is conducted with 25 sets of different initial random populations, again with each initial random population the experiment is repeated 100 times to access the consistency rate of the solution produced by the proposed HTGA. The percentage consistency rate is computed as {(number of trials getting COF value within five percent of the best found COF value /total number of trials)*100}. The summary of the findings are shown in Table 6, which exhibit that the consistency rate of proposed HTGA is considerably higher than the normal GA.

The performance of the HTGA is also compared with that of two meta heuristics, reported in the literature [16] viz (i). GA based heuristic, (ii). Tabu Search based heuristic. The cutsize obtained by these heuristic and the proposed HTGA is shown in Table 7.

It can be seen from the Table 7, that though the GA based heuristic proposed in the literature [16] is effective in minimizing the cutsize for smaller benchmark circuits, the Tabu Search based heuristic given in the literature is able to outperform the GA for larger benchmark circuits. The proposed HTGA overcomes this issue and produces lesser cutsize for all the benchmark circuits except S386 and S5378. For these two circuits cutsize produced by HTGA is marginally higher than the Tabu Search based meta heuristics but lower than GA based heuristics. The effectiveness of HTGA in producing better quality solutions could be attributed to the systematic reasoning ability of the Taguchi method, which is built in the proposed HTGA.

Again the proposed HTGA may be made to surpass the performance of TS for the circuits S386 and S5378 by designing an improved OA even with more than 2 levels, (if required), which is a part of the scope for future work.

As the hMetis [24] algorithm, and other algorithms such as MLP, MLC mentioned in the literature in section 2 are suited for only flat partitioning [3] and are capable of producing solutions even for very large size problems with appreciably lesser time with the objective of producing solution with satisfactory quality level, the run time of hMetis, MLP, MLC cannot be compared with that of the proposed HTGA, which uses recursive partitioning methodology and whose solution quality is expected to be much higher than that of the flat partitioning methodology [3,17-18].

Due to the recursive nature and a larger number of computations involved in OA, HTGA needs more computational time for larger scale benchmarks. However this issue could be addressed by constructing dedicated OA with more number of factors. And grouping of higher cardinality edges in a particular partition (P_i) instead of doing random initial population generation, which is again the scope for future work.

§6. Conclusion

In this work, an attempt is made to solve the VLSI circuit partitioning problem with an objective of minimizing the cutsize, that is, the number of wires passing between the partitions and also balancing the area between the partitions. An efficient hybrid Genetic Algorithm incorporating Taguchi method as a local search mechanism, named as, Hybrid Taguchi Genetic Algorithm (HTGA) has been developed to solve both the bipartitioning and recursive partitioning problem in the VLSI design process. The proposed HTGA is tested with a wide range of ISCAS89 benchmark circuits and its performance is compared with that of a standard GA (without the use of Taguchi as a local search tool) and it is found that HTGA out performs the standard GA both in terms of solution quality and the number of iterations required for reaching the nearer-to-optimal solution, due to the systematic reasoning ability of the Taguchi method. The experimentation with proposed HTGA was also repeated with the same and different input data sets and it was found that the proposed HTGA is consistent in producing quality solutions. The performance of HTGA is also compared with that of the GA and Tabu Search based meta heuristics reported in the literature. And it is found that the proposed HTGA is able to give better solutions than the GA based heuristics for all the benchmark circuits considered in this work. Compared to the Tabu Search based heuristic, the proposed HTGA is able to produce better solution for all the benchmark circuits except S386 and S5378. Again HTGA may be made to surpass the performance of TS for the circuits S386 and S5378 by designing an improved orthogonal array (OA) even with more than 2 levels (if required) which is a part of the scope for the future work.

Acknowledgement

The authors thank the Department of Science and Technology, New Delhi, Government of India, for the support under the project no: SR/S4/MS: 326/06.

References

[1] C.J. Alpert, J.H. Huang and A.B. Kahng, Multilevel circuit partitioning, in *Proceedings of the 34th annual Design Automation Conference* (1997),pp. 530-533

[2] Cristinel Ababei and Kia Bazargan, Timing minimization by statistical timing hMetis-Based partitioning, in *Proceedings of 16th International Conference on VLSI Design* (2003), pp. 58-63

[3] Areibi.S, Recursive and flat partitioning for VLSI circuit design, in *Proceedings of the 13th International Conference on Microelectronics* (2001), pp. 237- 240

[4] Byung-Ro Moon and Chun-Kyung Kim, Genetic VLSI circuit partitioning with dynamic embedding, in *Proceedings of the First International Conference on Knowledge-Based Intelligent Electronic Systems* (1997) Vol.2, pp. 461-469

[5] Coe. S, Areibi. S and Moussa. M, A genetic local search hybrid architecture for VLSI circuit partitioning, in *Proceedings of the 16th International Conference on Microelectronics* (2004), pp. 253-256

[6] Cong.J, W.labio and N.Sivakumar, Multi-way VLSI circuit partitioning based on dual net representation, in *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design* (San Jose, California, United States ,1994), pp.56-62

[7] C.M.Fiduccia and R.M.Mattheyses, A linear Time heuristic for improving network partitions, in *Proceedings of the 19th Design Automation Conference* (1982), pp. 291-307

[8] Jianhua Li and Behjat.L, A connectivity based clustering algorithm with application to VLSI circuit partitioning, *IEEE Trans. Circuits and Systems* (2006), Vol.53, pp. 384 - 388

[9] Jin-Tong Tsai, Tung-Kuan Liu, Jyh-Horng Chou, Hybrid taguchi genetic algorithm for global numerical optimization, *IEEE transaction on evolutionary computation* (2004), Vol.8, pp. 365-377

[10] Johannes.F.M, Partitioning of VLSI circuits and systems, in *Proceedings of the 33rd annual Design Automation Conference* (Las Vegas, Nevada, United States, 1996), pp.83-87

[11] Jong-Sheng Cherng and Sao-Jie Chen , An efficient multi level partitioning algorithm for VLSI Circuits, in *Proceedings of the 16th International Conference on VLSI Design* (2003), pp. 70-75

[12] B.W.Kernighan and S.lin, An efficient heuristic procedure for partitioning graphs, *The Bell System Technical Journal* (1970), Vol.49, pp. 291-307

[13] B.Krishnamurthy, An improved min-cut algorithm for partitioning VLSI networks, *IEEE Trans. Computers.,* (1984) Vol.33, 438-446

[14] Pinaki Mazumder and Elizabeth.M, *Genetic algorithms for VLSI design, layout and test automation* (Prentice Hall PTR, 1999)

[15] Sabiah.H.Gerez, *Algorithms for VLSI Design Automation*, Wiley, 1999.

[16] Sait.S.M, El-Maleh.A.H, Al-Abaji.R.H, General iterative heuristics for VLSI multiobjective partitioning, in *Proceedings of the International Symposium on Circuits and Systems* (ISCAS 2003), pp. 497-500

[17] L.A.Sanchis, Multiple -Way network partitioning, *IEEE Trans. Computers.,* (1989),C38, pp.62-81

[18] L.A.Sanchis. Multiple-Way network partitioning with different cost functions, *IEEE Trans. Computers.,* (1993), Vol.42, 1500-1514

[19] C.Sechen and D.Chen, An improved objective function for mincut circuit partitioning, in *Proceedings of the IEEE International Conference of Computer Aided Design* (1994), pp. 502-505.

[20] D.G.Schweikert and B.W.Kernighan, A proper model for the partitioning of electric circuits, in *Proceedings of the 9th Design Automation Workshop* (1979), pp. 57-62

[21] Tapan. P. Bagchi, *Taguchi methods explained*, Prentice-Hall, 1993.

[22] X.Tan, J.Tong and P.Tan, An efficient multi way algorithm for balanced partitioning of VLSI circuits, in *Proceedings of the IEEE International Conference on Computer Design* (1997), pp.608

[23] Tumbush.G and Bhatia.D, Partitioning under timing and area constraints, *Proceedings of the IEEE international conference* VLSI in Computers and Processors (1997), pp.614 - 620

[24] Vipin Kumar, George Karypis, Rajat Aggarwal, and Shashi Shekhar, Multilevel Hypergraph Partitioning: Applications in VLSI Domain, *IEEE Transactions on VLSI Systems* (1999), Vol. 7, No. 1, pp. 69-79.

Appendix:

Table 1: An example Orthogonal Array, $L_8(2^7)$

	Factors						
	1	2	3	4	5	6	7
Experiment	A	B	C	D	E	F	G
number	Levels assigned						
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

Table 2: An example calculation of Taguchi method.

Step a: Select a suitable two level orthogonal array, say $L_8(2^7)$ shown in Table 1

Step b: Randomly select two chromosomes from the intermediate crossover population

Chromosome X : 1 0 1 1 1 1 1 (level 1)
Chromosome Y : 0 1 1 1 0 1 0 (level 2)

Step c: Taguchi Experiment

	Factors								
	1	2	3	4	5	6	7		
Experiment	A	B	C	D	E	F	G	Function value COF_i	$SNR(\eta_i)$
1	1	0	1	1	1	1	1	3.5	0.28
2	1	0	1	1	0	1	0	2.0	0.50
3	1	1	1	1	1	1	1	4.0	0.25
4	1	1	1	1	0	1	1	5.0	0.20
5	0	0	1	1	0	1	0	3.0	0.33
6	0	0	1	1	1	1	1	3.0	0.33
7	0	1	1	1	0	1	1	3.0	0.33
8	0	1	1	1	1	1	0	5.0	0.20
Ef_1	1.23	1.44	1.31	1.19	1.06	1.14	1.14		
Ef_2	1.19	0.98	1.10	1.23	1.36	1.41	1.28		
Optimal Level	1	1	1	2	2	2	2		

Step d: Construct a new chromosome

Optimal Chromosome Z	1	0	1	1	0	1	0

Table 3: Details of ISCAS89 benchmark problems tested with HTGA

S.NO	Benchmark Circuit Code	Number of Elements	Number of Interconnections
1	S27	18	13
2	S208	117	108
3	S298	136	130
4	S386	172	165
5	S641	433	410
6	S832	310	291
7	S953	440	417
8	S1196	561	547
9	S1238	540	526
10	S1488	667	648
11	S1494	661	642
12	S5378	2994	2944
13	S9234	5845	5822
14	S13207	8652	8530
15	S15850	10384	10296

Table 4: Performance comparison between GA and HTGA in bipartitioning

Benchmark Circuit	Standard Genetic Algorithm				
	Cut size (F_1)	Area (F_2)	COF	No. of Generations	CPU time (s)
S27	3	2	2.5	2	2
S208	30	20	25	25641	552
S298	15	26	20.5	4872	95
S832	40	84	62	28436	278
S386	38	101	69.5	7985	165
S641	47	128	87.5	33700	1506
S953	95	139	117	27741	600
S1196	110	13	61.5	6654	396
S1238	98	65	81.5	4385	380
S1488	104	10	57	9359	1058
S1494	104	18	61	8659	1102
S5378	541	30	285.5	12658	1956
S9234	1082	42	562	28958	4558
S13207	1602	80	841	30258	6582
S15850	2186	24	1105	38598	8965
HTGA					
S27	3	1	2	2	2
S208	27	18	22.5	9189	659
S298	13	25	19	2346	112
S832	39	74	56.5	18849	290
S386	32	95	63.5	3339	170
S641	44	117	80.5	29221	1600
S953	84	141	112.5	21080	556
S1196	102	13	57.5	4159	398
S1238	73	74	73.5	2958	302
S1488	92	18	55	8158	650
S1494	101	19	60	6858	520
S5378	463	36	249.5	9958	952
S9234	915	46	480.5	12554	2858
S13207	1328	91	709.5	20587	4965
S15850	1665	30	847.5	25987	4895

Table 5: Performance comparison between GA and HTGA in Multi-Partitioning(4-Partitions)

Benchmark Circuit	Standard Genetic Algorithm				
	Cut size (F_1)	Area (F_2)	COF	No. of Generations	CPU time (s)
S27	6	3	4.5	11	15
S208	45	19	32	37580	705
S298	55	19	37	10144	192
S832	97	27	62	48325	596
S386	72	105	88.5	16470	421
S641	99	83	91	49435	3254
S953	102	115	108.5	45434	1000
S1196	123	8	65.5	12065	821
S1238	118	49	83.5	8658	859
S1488	112	6	59	15285	3548
S1494	123	11	67	16258	2658
S5378	552	25	288.5	24585	4586
S9234	1125	33	579	45866	5486
S13207	1658	45	851.5	60258	8456
S15850	2103	18	1060.5	66558	12455
		HTGA			
S27	5	2	3.5	10	13
S208	34	20	27	17125	802
S298	48	22	35	4913	185
S832	85	21	53	26218	630
S386	69	98	83.5	15264	513
S641	80	52	66	34934	3951
S953	123	68	95.5	31849	916
S1196	112	10	61	4586	795
S1238	98	40	69	4589	698
S1488	102	6	54	10258	2854
S1494	119	11	65	12859	1425
S5378	545	22	283.5	18548	1922
S9234	1123	30	576.5	25866	3596
S13207	1659	42	850.5	40287	4987
S15850	2102	18	1060	39854	7584

Table 6: Comparison on consistency rate between GA and HTGA

Benchmark	Consistency rate	
Circuit	Genetic Algorithm	HTGA
S27	40	60
S208	46	63
S298	52	68
S832	58	66.25
S386	62.5	71
S641	48	62
S953	46	63
S1196	48	69.65
S1238	40.5	70.6
S1488	45.26	69.24
S1494	49.65	65
S5378	55	70.65
S9234	48.4	67.25
S13207	59.65	69
S15850	51	68.6

Table 7: Cutsize Comparison of HTGA with GA and TS (S.MSait)

Benchmark	Cutsize of the Benchmark Circuits		
Circuit	Genetic Algorithm	Tabu Search	HTGA
S298	19	24	13
S832	45	50	39
S386	36	30	32
S641	45	59	44
S953	96	99	84
S1196	123	106	102
S1238	127	79	73
S1488	104	98	92
S1494	102	101	101
S5378	573	430	463
S9234	1090	918	915
S13207	1683	1332	1328
S15850	2183	1671	1665