GATS: A Novel Hybrid Algorithm for Multiobjective Cell Placement in VLSI Circuit Design

Sadiq M. Sait
Department of Computer Engineering

Mahmood R. Minhas
Department of Information & Computer Science

King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
e-mail: {sadiq,minhas}@ccse.kfupm.edu.sa

Abstract

This paper addresses the optimization of cell placement step in VLSI circuit design [1]. A novel hybrid algorithm is proposed for performance and low power driven VLSI standard cell placement. The above problem is of multiobjective nature since three possibly conflicting objectives are considered to be optimized subject to the constraint of layout width. These objectives are power dissipation, timing performance, and interconnect wire length. It is well known that optimizing cell placement for even a single objective namely total wire length is a hard problem to solve. Due to imprecise nature of objective values, fuzzy logic is incorporated in the design of aggregating function. The above technique is applied to the placement of ISCAS-89 benchmark circuits and the results are compared with those obtained from individual application of GA and TS on this problem.

1 Introduction

Hybrid algorithms combine features from various heuristics as an effort to develop efficient techniques for solving hard optimization problems [2]. In the scope of the present work, hybridization refers to mixing of good characteristics from several iterative algorithms. The choice of iterative algorithms to be hybridized needs some careful analysis of the underlying problem and the characteristics of individual candidate algorithms. The application of an individual algorithm to the problem may give an insight of its suitability to the problem. If the results of application of the individual algorithms are encouraging, then their hybridization is expected to produce even better performance in many cases.

Genetic Algorithm (GA) and Tabu search (TS) are two well-known iterative heuristics that have been applied for solving a range of combinatorial optimization problems in various disciplines of science and engineering [1, 2]. Both of these algorithms have exhibited good performance for the present problem [3, 4]. Therefore, it seems reasonable to hybridize GA and TS with the view of developing an efficient hybrid technique for timing and low power driven placement. A good property of GA is its implicit parallel nature that helps in exploring the search space efficiently. This parallelism is due to the fact that GA processes a population of solutions instead of a single solution. On the other hand, TS showed better results than GA, and this better performance can be attributed to TS searching mechanism. These facts lead us to a proposition that an efficient hybrid technique can be developed by combining the features from these two iterative algorithms. The experimental results support our proposition as the proposed GATS was able to obtain the results that are better than those obtained by using TS as well as GA.

This paper is organized as follows: In the next section, we formulate our problem and cost functions. Section 3 presents the details of our proposed approach, and the experimental results are presented and discussed in section 4.

2 Problem Formulation and Cost Functions

In this section, we formulate our problem and the cost function used in our optimization process.

VLSI design is a complex process and is carried out at certain abstraction levels [1]. We are addressing the problem at cell placement level with the objectives of optimizing power consumption, timing performance (delay), and wire length while considering layout width as a constraint. Formally, the problem can be stated as follows: *A set of cells or modules* $M = \{m_1, m_2, ..., m_n\}$ *and a set of signals* $S = \{s_1, s_2, ..., s_k\}$ *is given. Moreover, a set of signals* S_{m_i}*, where* $S_{m_i} \subseteq S$*, is associated with each mod-*

ule $m_i \in M$. Similarly, a set of modules M_{s_j}, where $M_{s_j} = \{m_i | s_j \in S_{m_i}\}$ is called a signal net, is associated with each signal $s_j \in S$. Also, a set of locations $L = \{L_1, L_2, ..., L_p\}$, where $p \geq n$ is given. The problem is to assign each $m_i \in M$ to a unique location L_j, such that all of our objectives are optimized subject to our constraint.

2.1 Cost Functions

Now we formulate cost functions for our three said objectives and for the width constraint.

Wire length Cost: Interconnect Wire length of each net in the circuit is estimated and then total wire length is computed by adding all these individual estimates:

$$Cost_{wire} = \sum_{i \in M} l_i \tag{1}$$

where l_i is the wire length estimation for net i and M denotes total number of nets in circuit (which is the same as number of modules for single output cells).

Power Cost: Power consumption p_i of a net i in a circuit can be given as:

$$p_i \simeq \frac{1}{2} \cdot C_i \cdot V_{DD}^2 \cdot f \cdot S_i \cdot \beta \tag{2}$$

where C_i is total capacitance of net i, V_{DD} is the supply voltage, f is the clock frequency, S_i is the switching probability of net i, and β is a technology dependent constant.

Assuming a fix supply voltage and clock frequency, the above equation reduces to the following:

$$p_i \simeq C_i \cdot S_i \tag{3}$$

The capacitance C_i of cell i is given as:

$$C_i = C_i^r + \sum_{j \in M_i} C_j^g \tag{4}$$

where C_j^g is the input capacitance of gate j and C_i^r is the interconnect capacitance at the output node of cell i.

At the placement phase, only the interconnect capacitance C_i^r can be manipulated while C_j^g comes from the properties of the cell library used and is thus independent of placement. Moreover, C_i^r depends on wire length of net i, so equation 3 can be written as:

$$p_i \simeq l_i \cdot S_i \tag{5}$$

The cost function for total power consumption in the circuit can be given as:

$$Cost_{power} = \sum_{i \in M} p_i = \sum_{i \in M} (l_i \cdot S_i) \tag{6}$$

Delay Cost: Delay cost is determined by the delay along the longest path in a circuit. The delay T_π of a path π consisting of nets $\{v_1, v_2, ..., v_k\}$, is expressed as:

$$T_\pi = \sum_{i=1}^{k-1} (CD_i + ID_i) \tag{7}$$

where CD_i is the switching delay of the cell driving net v_i and ID_i is the interconnect delay of net v_i. The placement phase affects ID_i because CD_i is technology dependent parameter and is independent of placement.

The delay cost function can be written as:

$$Cost_{delay} = max\{T_\pi\} \tag{8}$$

Width Cost: Width cost is given by the maximum of all the row widths in the layout. We have constrained layout width not to exceed a certain positive ratio α to the average row width w_{avg}, where w_{avg} is the minimum possible layout width obtained by dividing the total width of all the cells in the layout by the number of rows in the layout. Formally, we can express width constraint as below:

$$Width - w_{avg} \leq \alpha \times w_{avg} \tag{9}$$

Overall Fuzzy Cost Function: Since, we are optimizing three objectives simultaneously, we need to have a cost function that represents the effect of all three objectives in form of a single quantity. We propose the use of fuzzy logic to integrate these multiple, possibly conflicting objectives into a scalar cost function. Fuzzy logic allows us to describe the objectives in terms of linguistic variables. Then, fuzzy rules are used to find the overall cost of a placement solution. In this work, we have used following fuzzy rule:

IF a solution has
SMALL wire length **AND**
LOW power consumption **AND**
SHORT delay
THEN it is an *GOOD* solution.

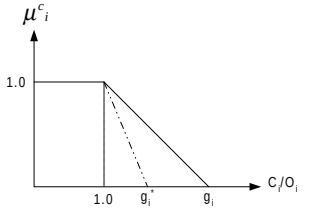

Figure 1: Membership functions

The above rule is translated to *and-like* OWA fuzzy operator [5] and the membership $\mu(x)$ of a solution x

in fuzzy set *GOOD solution* is given as:

$$\mu(x) = \begin{cases} \beta \cdot \min\limits_{j=p,d,l} \{\mu_j(x)\} + (1-\beta) \cdot \frac{1}{3} \sum\limits_{j=p,d,l} \mu_j(x); \\ \qquad \text{if} \quad Width - w_{avg} \leq \alpha \cdot w_{avg}, \\ 0; \qquad \text{otherwise.} \end{cases} \tag{10}$$

Here $\mu_j(x)$ for $j = p, d, l, width$ are the membership values in the fuzzy sets *LOW power consumption, SHORT delay*, and *SMALL wire length* respectively. β is the constant in the range $[0, 1]$. The solution that results in maximum value of $\mu(x)$ is reported as the best solution found by the search heuristic.

The membership functions for fuzzy sets *LOW power consumption, SHORT delay*, and *SMALL wire length* are shown in Figure 1. We can vary the preference of an objective j in overall membership function by changing the value of g_j . The lower bounds O_j for different objectives are computed as given in Equations 11-14:

$$O_l = \sum_{i=1}^{n} l_i^* \quad \forall v_i \ \in \ \{v_1, v_2, ..., v_n\} \tag{11}$$

$$O_p = \sum_{i=1}^{n} S_i l_i^* \quad \forall v_i \ \in \ \{v_1, v_2, ..., v_n\} \tag{12}$$

$$O_d = \sum_{j=1}^{k} CD_j + ID_j^* \quad \forall v_j \ \in \ \{v_1, v_2, ..., v_k\} \ in \ path \ \pi_c \tag{13}$$

$$O_{width} = \frac{\sum_{i=1}^{n} Width_i}{\#\ of\ rows\ in\ layout} \tag{14}$$

where O_j for $j \in \{l, p, d, width\}$ are the optimal costs for wire-length, power, delay and layout width respectively, n is the number of nets in layout, l_i^* is the optimal wire-length of net v_i, CD_i is the switching delay of the cell i driving net v_i, ID_i is the optimal interconnect delay of net v_i calculated with the help of l_i, S_i is the switching probability of net v_i, π_c is the most critical path with respect to optimal interconnect delays, k is the number of nets in π_c and $Width_i$ is the width of the individual cell driving net v_i.

3 GATS for Performance and Low Power Driven VLSI Placement

In this section, we first briefly describe GATS algorithm and then discuss the implementation details of GATS for multiobjective VLSI placement.

Figure 2 illustrates the proposed hybrid algorithm GATS. An interesting novel idea is the introduction of a population of solutions instead of single solution in

```
Algorithm GATS
S    :  Current solutions
S*   :  Best solutions
N(S) :  Neighborhood of S ∈ Ω
V*   :  Sample of neighborhood solutions
AL   :  Aspiration levels
N_G  :  Population size for GA portion
N_T  :  Population size for TS portion
N_o  :  Number of Offsprings

Begin
For fixed number of times Do
Start with random initial population N_T
    For fixed number of iterations Do
      For j = 1 To N_T
        Generate neighbor solutions V* (j) ⊂ N (S (j)) by random swap
        Find best S* (j) ∈ V* (j)
        If move S (j) to S* (j) is not in T (j) Then
            Accept move and update T (j)
        Else
            If Cost (S* (j)) < AL (j) Then
                Accept move and update T (j) and AL
            End If
        End If
      End For
    End For
    Pass best or current solutions to GA
    For fixed number of generations Do
        For j = 1 To N_o
            (x, y) ← Choose parents
            Offspring [j] ← Crossover (x, y)
            Evaluate Fitness (offspring [j])
        End For
        Population ← Select (Population, offspring, N_G)
        For j = 1 To N_G
            Apply Mutation (chromosome [j])
            Evaluate Fitness (chromosome [j])
        End For
    End For
    Pass best or current solutions to TS
End For
End.
```

Figure 2: Outline of GATS: A Hybrid of GA and TS for Multiobjective VLSI Cell Placement.

TS. This is likely to enhance the power of TS by allowing it to visit the search space in a parallel fashion. The algorithm starts by taking a random initial population of solutions. Then, for each individual in the population, a certain number of neighbor solutions are generated and the best neighbor is found. A characteristic of the move leading to the best neighbor solution is stored in a tabu list. There are as many tabu lists as the number of solutions in the population i.e., N_T. The reason for taking N_T tabu lists is obvious that the series of moves for each individual in the population is different. Therefore, each series should be stored in a separate list so that a tabu list restricts the cyclic moves on its corresponding individual only. However, the aspiration level (AL) is unique for all the individuals. The purpose is that the tabu move on an individual solution is allowed only if it results in a solution that is better than an overall unique best solution.

The above process continues for a certain number of iterations and a record is kept of the N_T individual best solutions obtained from perturbing N_T individual initial solutions. Then either these best solutions or the current solutions are passed to GA for further optimization. These semi-optimized individ-

uals are likely to produce good offsprings by mating with one another. Now, GA is run for a given number of generations on these passed solutions and a record of the best individuals is kept. Again, either the current individuals or best ones are passed back to TS. The switching between TS and GA is repeated for a given number of times.

3.1 Solution Representation and Initialization

A placement solution is an arrangement of cells in two dimensional layout surface. So we decided to represent solution in the form of a 2-D grid. Due to varying widths of the cells in a circuit, all the rows can not have equal number of cells. This fact disturbs our two dimensional representation. For instance consider a circuit comprising of 11 cells $1, 2, 3, \ldots, 11$. A possible layout may be as below:

```
3   5   8   6
9  10
7  11   1
4   2
```

The above layout is constructed by computing the average row width as explained above in the cost functions section when discussing width cost. Then we divide average row width by the smallest cell width to compute the maximum number of locations in a row. Assume that we have 4 locations and also we know from the min-cut placer information that there are 4 rows in layout. Then we start constructing the initial solution by randomly selecting a cell from 11 cells and placing it in the first row. Before placing a cell, it is checked whether adding it will violate the width constraint, and if it does, then it is placed at the start of next row. In the example above, assume that sum of widths of cells $3, 5, 8, 6$ was within allowed width constraint, but adding cell 9 was violating the width constraint, and so it was placed in second row. Similarly, all the cells were placed on the layout. As a result, we have five empty locations: two in second row, one in third row, and two in last row. In order to make it a perfect grid, we fill the empty locations by dummy cells represented by distinct negative integers as shown below:

```
3   5   8   6
9  10  -1  -2
7  11   1  -3
4   2  -4  -5
```

3.2 Cost Evaluation

Since, we are addressing a multiobjective optimization problem in which we are trying to minimize three mutually conflicting objectives, therefore we should have a measure which can quantify the overall quality of a solution with respect to all three objectives collectively. Fuzzy logic provides a convenient approach and hence used in this research. In this scheme, each solution is assigned a fitness value between 0 and 1 that is equal to the membership value in the fuzzy set of acceptable solution. This membership value is computed using Equation 10. The fitness of a solution is a measure of its proximity to the optimal solution. The higher the fitness value of a solution, the closer is it to the optimal solution. In our implementation, initial random solution is assigned a fitness value of 0 and the optimal solution is assigned a fitness value of 1. This implies that any solution may have a fitness value in range 0.0-1.0.

3.3 Neighbor Solutions Generation

In each iteration, we generate a number of neighbor solutions by making perturbations as follows: two cells are selected randomly with the condition that both of them should not be dummy cells at the same time, then their locations are interchanged. The neighborhood size i.e., the number of neighbor solutions generated in each iteration is taken depending on the circuit size i.e. number of cells in the circuit. The value of neighborhood size is varied from 20 solutions for small circuits to 100 solutions for large circuits.

3.4 Tabu List and Aspiration Level

The characteristic of the move that we keep in tabu list is the indices of the cells involved in interchange. The size of tabu list is taken also depending on the circuit size i.e. 5% of the total number of cells. We have used short term memory element in our TS implementation. The aspiration criterion used is as follows: if current best solution is the best seen so far i.e. better than the global best, then accept the current solution as new best solution by overriding the tabu restriction and update the tabu list.

4 Experimental Results and Discussion

Here, the performance of proposed hybrid algorithm GATS is compared with that of TS. Since TS outperformed GA in terms of the quality of final solution obtained, so the comparison presented in this section is of great interest.

The costs of the best solutions generated by TS and GATS are listed in Table 1. Here "L", "P" and "D" represent the wire length, power and delay costs respectively, and "T" represents execution time in seconds. Layout width was constrained not to exceed

	TS				GATS			
Circuit	L (μm)	P	D (ps)	T (s)	L (μm)	P	D (ps)	T (s)
s2081	2323	379	111	298	2162	356	110	426
s298	3579	635	127	212	3454	631	125	362
s386	6643	1595	190	524	6329	1486	191	656
s641	12620	2868	656	1505	12433	2882	664	2143
s832	18760	4311	349	981	18451	4257	351	1929
s953	27287	4230	214	1036	25967	4239	212	1846
s1196	39054	11700	332	1138	38574	11526	334	3276
s1238	39186	11594	353	1124	39065	11342	351	3667
s1488	56888	13867	662	2256	56148	14135	669	5157
s1494	54710	13533	674	2499	54914	13763	668	5643

Table 1: Comparison between TS and GATS.

more than 1.2 times the average row width by fixing the value of α in equation 9 equal to 0.2. This constraint is satisfied in obtaining all the results shown here.

The results of TS are obtained by best settings of its parameters as described above. The settings of GATS parameters used for achieving theses results are as follows. Total number of iterations run are 5000, which comprise of 2000 TS iterations and 3000 GA generations. The switch from TS to GA is made only once. The population size N_T used in TS part is 4 while in GA part the population size is 16 chromosomes. This fine tuning of parameters is made after careful study of the results obtained by choosing different settings. The population size in case of TS is reduced after observing that large population size increases run time of TS part without providing any significant performance. By taking this step, the run time of GATS is shortened very significantly. The platform used is an IBM compatible PC with an Intel Pentium-III 600Mhz CPU and 256MB RAM.

It can be observed from the results that in most of cases, GATS produced solutions which are better in quality as compared to those obtained from TS. Although, the execution time of GATS is higher than TS, but this is tolerable considering the better quality of solutions. The overall performance of GATS is comparable to that of TS, and much better than GA.

5 Conclusions

In this work, we have hybridized GA and TS for a hard multiobjective optimization problem of VLSI standard cell placement. An effort is made to simultaneously optimize three objectives namely power dissipation, performance, and interconnect wire length. The incorporation of fuzzy logic is suggested to integrate the cost values of three objectives in an aggregating cost function. The experimental results for ISCAS-89 benchmarks clearly indicate the improvement made by our GATS approach in terms of quality of the final solution obtained.

Acknowledgment:
Authors thank King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, for support under project # COE/ITERATE/221.

References

[1] Sadiq M. Sait and Habib Youssef. VLSI Physical Design Automation: Theory and Practice. *Mc Graw-Hill Book Company, Europe*, 1995.

[2] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society Press, California, December 1999.

[3] Sadiq M. Sait, Habib Youssef, Aiman Al-Maleh, and Mahmood R. Minhas. Iterative Heuristics for Multiobjective VLSI Standard Cell Placement. *INNS-IEEE International Joint Conference on Neural Networks, IJCNN2001*, July 2001.

[4] Sadiq M. Sait, Mahmood R. Minhas, and Junaid A. Khan. Performance and low power driven VLSI standard cell placement using Tabu search. *In Proceedings of the IEEE Congress on Evolutionary Computation, CEC'2002.*, 1:372–377, 2002.

[5] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transaction on Systems, MAN, and Cybernetics*, 18(1), January 1988.