Iterative Heuristics for Multiobjective VLSI Standard Cell Placement

Sadiq M. Sait Habib Youssef Aiman H. El-Maleh Mahmood R. Minhas

King Fahd University of Petroleum and Minerals
Computer Engineering Department
Dhahran 31261, Saudi Arabia
E-mail: {sadiq,youssef,aimane,minhas} @ccse.kfupm.edu.sa

Abstract

We employ two iterative heuristics for the optimization of VLSI standard cell placement. These heuristics are based on Genetic Algorithms (GAs) and Tabu Search (TS) [1] respectively. We address a multiobjective version of the problem in which, power dissipation, timing performance, and interconnect wire length are optimized while layout width is taken as a constraint. Fuzzy rules are incorporated in order to design a multiobjective cost function that integrates the costs of three objectives in a single overall cost value. A series of experiments is performed to study the effect of important algorithmic parameters of GA and TS. Both the techniques are applied to ISCAS-85/89 benchmark circuits and experimental results are reported and compared.

1 Introduction

Until the beginning of this decade, two objectives namely the optimization of interconnect wire length and performance were focused. A large number of efforts targeting either one or both of above two objectives are reported in the literature [2, 3, 4]. There has been some work for optimizing power consumption while considering the wire length and performance as constraints [5, 6], but, to our knowledge, no effort has been reported that targets the optimization of the three objectives simultaneously. This fact provides a significant motivation for the present work.

VLSI design is a complex process and is carried out at certain abstraction levels [7] as illustrated in figure 1. The problem of power optimization can be addressed at a higher level as well as at a lower level e.g., physical level [8, 9]. In this work, we address the above problem in the placement step at the physical level. Two iterative approaches based on genetic algorithm (GA) and tabu search (TS) respectively, are presented for the multiobjective optimization of the placement.

Placement is an important step in VLSI physical design responsible for arrangement of cells on a layout surface for optimizing certain objectives while satisfying some constraints. Standard cell placement is a special case where all the cells to be placed have equal height [7].

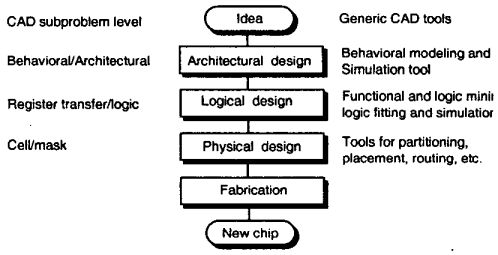

Figure 1: Various steps in VLSI design process.

This paper is organized as follows: In the next section, we formulate our problem and cost function. Section 3 presents our approaches, and then experimental results are reported and discussed in section 4.

2 Problem and Cost Function Modeling

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

2.1 Problem Formulation

We are addressing the problem of VLSI standard cell placement 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 module* $m_i \in M$*. 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 constraints.

2.2 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.

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 vi and ID_i is the interconnect delay of net vi. 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:

$$Width \leq (1 + \alpha) \times w_{avg} \tag{9}$$

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 a *GOOD* solution.

The above rule is translated to *and-like* OWA fuzzy operator [10] and the membership $\mu(x)$ of a solution x in fuzzy set *GOOD solution* is given as:

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

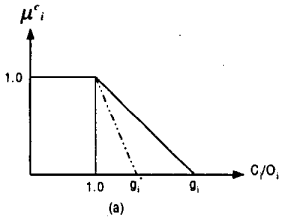

Figure 2: Membership functions

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 2. 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 Proposed Approaches

In this section, we describe the implementation details of the proposed approaches. First, we discuss the details of Genetic Algorithm for Multiobjective Placement and then, we briefly describe the implementation of Tabu Search (TS).

3.1 GA for Multiobjective Placement (GA)

A significant modification in our GA implementation is the application of mutation after selection step as will be described in the corresponding sections below.

Chromosome Encoding and Initial Solution: For a solution to be processed by GA, it is required to be represented in the form of a chromosome. A placement solution is an arrangement of cells in two dimensional layout surface. So we decided to represent a solution in the form of a 2-D grid. As we have discussed above that 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 is as follows:

3 5 8 6
9 10
7 11 1
4 2

The above layout is made after computing the average row width as explained above in section 2.2. Then we divide average row width by the smallest cell width to compute the maximum number of locations in a row. Assume we have 4 locations and it is known from the information obtained from the min-cut placer that there are 4 rows in the 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. Similarly, all the cells were placed on the layout. As a result, we have five empty locations: two in the second row, one in the third row, and two in the 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

These negative numbers are used for encoding purposes as well as for the appropriate application of genetic operators like crossover, and these do not play any role in cost computation of the solution.

Fitness Evaluation: As discussed in the previous section that fuzzy logic is used for designing the overall cost function. 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, an initial solution is assigned a fitness value of 0 whereas the optimal solution is assigned a fitness value of 1. The purpose is to normalize the fitness value of any solution in range [0,1].

Parent Choice: We have used *roulette wheel* scheme [1]. An individual chromosome is selected with a probability that is proportional to its fitness value. This scheme allows the individuals having low fitness values to be selected with a low probability.

Crossover: Crossover is the operator that causes inheritance of characteristics from one generation to the next. Various types of crossover operators including one-point or two-point simple crossover, order crossover, and partially mapped crossover (PMX) are reported in the literature. In our case, each gene in the chromosome representation is distinct and this property must remain there from generation to generation for a chromosome to represent a valid solution. Therefore, we can not use simple crossover as it may result in duplicate genes. In our work, we experiment with different types of existing crossover operators like order crossover, and PMX. In addition, we propose a modified form of PMX application named Controlled Dual PMX (CDX). CDX works as follows: We first apply PMX between parent-1 and parent-2 and then between parent-2 and parent-1. As a result, we get two offsprings. We choose the better offspring with a certain high probability P_{cdx} and choose any one of the two randomly with probability $1 - P_{cdx}$.

The crossover operation is performed with a high probability p_c. We experiment with different high crossover probabilities. Our implementation of crossover probability is as follows: after choosing two parents, we generate a random number *rand* in the range $[0, 1]$, and if $rand < p_c$, we apply crossover; otherwise, we choose another two parents and regenerate a random number and so on. In this way, it is ensured that same number of offsprings are generated in each iteration. We generate offsprings equal in number to the population size. Whenever a new offspring is generated, we check whether the width cost of the circuit is violating the width constraint. If it is so, we discard that offspring and perform another crossover after choosing other parents. This process is repeated until we have the desired number of offsprings.

Selection: We suggest a modification to simple GA by using selection before the mutation step. The motivation is to encourage the diversity in the population by ensuring the transfer of mutation effect into next generation. We have experimented with different selection schemes including *roulette (rlt), elitist-roulette (erlt), elitist-random (ernd)*, and *extended-elitist-random (eernd)* schemes. In *eerlt* selection, the best half of the chromosomes are selected and the remaining half are selected using roulette wheel. Similarly, in *erlt* selection, the best chromosome is selected among parents and offsprings, and the remaining chromosomes are selected using roulette wheel. In *eernd* selection, the best half of the chromosomes are selected and the remaining half are selected randomly. Similarly, in *ernd* selection, the best one is selected from parents and offsprings, then the remaining are selected randomly.

Mutation: In this work, we propose to apply mutation with a dynamic probability P_μ^k which is a function of the diversity of the population selected for the next generation i.e $(k+1)th$ generation. The standard deviation of the population in $(k+1)th$ generation (σ_{fit}^k) is used as a measure of the diversity. The idea is to increase the mutation probability when population tends to lose diversity. This is another effort to enhance diversity and thus GA search capability. The following equation specifies how mutation probability is updated dynamically.

$$P_\mu^k = \begin{cases} 0.05 & \text{if } \sigma_{fit}^k < 0.02 \\ 0.03 & \text{if } \sigma_{fit}^k > 0.05 \\ 0.05 - \frac{2}{3}\,(\sigma_{fit}^k - 0.02) & \text{otherwise} \end{cases} \quad (15)$$

We implement mutation as a series of random pair wise interchanges. The number of interchanges is taken depending on the size of circuit. A random fraction f between 0.03 and 0.05 is generated and $f \times n$ interchanges are made, where n is the total number of cells in the circuit.

3.2 TS Approach

In this sub-section, we describe our TS implementation very briefly. The solution encoding and initialization steps are similar to those described above for GA. 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 number of neighbor solutions generated is dependent on the circuit size i.e., it is varied from 24 for s2081 to 70 for

Table 1: Comparison between costs of the best solutions generated by GA and TS

Circuit		GA					TS				
Name	N	L (μm)	P	D (ps)	$\mu(x)$	T(s)	L (μm)	P	D (ps)	$\mu(x)$	T(s)
s2081	122	2426	388	113	0.785	2341	2323	379	111	0.823	298
s298	136	4062	838	130	0.775	2922	3579	635	127	0.821	212
s386	172	6824	1665	193	0.695	3945	6643	1595	190	0.709	524
s832	310	21015	4787	395	0.598	7206	18760	4311	349	0.658	981
s641	433	17812	4532	740	0.685	21982	12620	2868	656	0.800	1505
s953	440	31004	5027	235	0.606	11221	27287	4230	214	0.669	1036
s1238	540	50387	15035	396	0.536	16208	39186	11594	353	0.656	1124
s1196	561	48729	14755	372	0.543	16120	39054	11700	332	0.650	1138
s1494	661	69223	17169	771	0.540	26032	54710	13533	674	0.650	2499
s1488	667	69792	17346	784	0.518	21434	56888	13867	662	0.621	2256
c3540	1753	310996	109850	924	0.425	57724	164581	58143	699	0.708	19215

c3540. The characteristic of the move which we keep in tabu list is the indices of the cells involved in interchange. We have used short term memory element in our TS implementation. The aspiration criterion used is as follows: if the best neighbor solution in the current iteration is the best seen so far i.e. better than the global best, then it is accepted and tabu restriction is overridden.

4 Experimental Results and Discussion

We have experimented with GA and TS on different ISCAS-85/89 benchmark circuits. Table 1 compares the quality of the final solution generated by TS and GA. The circuits are listed in the increasing order of the number of cells N in them. Here "L", "P" and "D" represent the wire length, power and delay costs respectively, $\mu(x)$ represents fuzzy membership and "T" represents execution time in seconds. Layout width was constrained not to exceed more than 1.2 times the average row width by fixing the value of α equal to 0.2. This constraint is satisfied in obtaining all the results shown here. The shown results in case of GA are obtained in 10,000 generations by taking a population size equal to 32 chromosomes and using CDX crossover with a probability equal to 0.9. The value of parameter P_{cdx} is set equal to 0.95.

As we have discussed above, P_{cdx} determines the probability with which the better of the two offsprings is selected. The selection scheme used is extended-elitist-random selection. These settings produced the best final results. The settings for TS parameters like neighborhood size and tabu list size, which produced the shown results are the same as mentioned above in section 3.2.

From the results, it is clear that TS performs better than GA for all the circuits in terms of quality of solution as well as run time. The significant observation

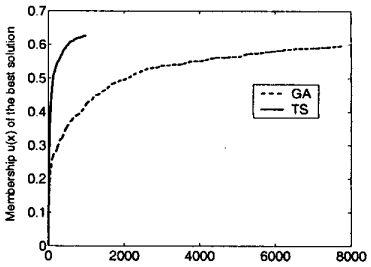

Figure 3: Membership of the best solution against run time for GA and TS in case of circuit s832.

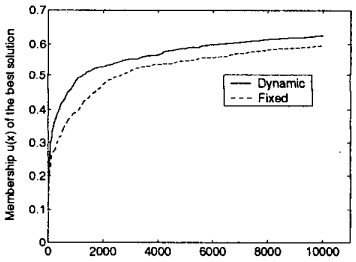

Figure 4: A comparison between using dynamic and fixed mutation probability in GA for circuit s832

is that in case of smaller circuits, the costs of final solution obtained from GA are comparable with those obtained from TS. As the size of the circuit increases, TS performs better tan GA consistently. Also, the execution time of GA increases significantly with the increase in the circuit size. Figure 3 shows the trend of best solution fuzzy membership against run time both in case of GA and TS for circuit s832. It is clear from the shown plot that TS achieves a membership in less than 1000 seconds that is better than that reached by GA in 8000 seconds.

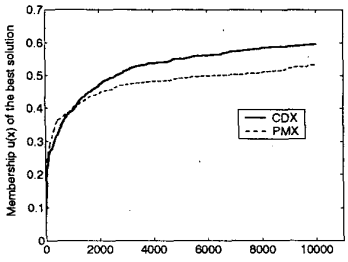

Figure 5: A comparison between PMX and CDX for s832

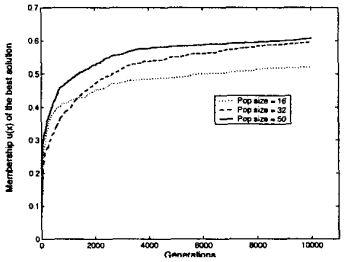

Figure 6: Effect of population size for s832

Figure 4 compares the trend of best solution cost in case of fixed and dynamic mutation probability. It is clear that using dynamic mutation probability results in slightly better performance. Next, figure 5 shows the effect of employing CDX and PMX crossover. The proposed CDX has clearly outperformed PMX. The effect of population size on the performance of GA can be seen in figure 6, where we have shown three plots for population sizes of 16, 32, and 50 respectively. It is observed that by increasing population size from 16 to 32, there is a reasonable performance improvement, but further increase in population size does not offer a significant improvement.

5 Conclusions

We have presented two iterative algorithms based on GA and TS respectively, for the multiobjective optimization of the VLSI standard cell placement. The use of fuzzy logic is proposed to integrate the three objectives namely power, delay, and wire length into a scalar cost value. Experimental results on ISCAS benchmark clearly indicate that TS outperforms GA in terms of the quality of the final solution as well as the execution time.

Acknowledgment:
The authors thank King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, for support. Project Code: COE/ITERATE/221

References

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

[2] K. Shahookar and P. Mazumder. VLSI Cell Placement Techniques. *ACM Computing Surveys*, 2(23):143–220, June 1991.

[3] R. M. Kling and P. Banerjee. ESP: Placement by Simulated Evolution. *IEEE Transaction on Computer-Aided Design*, 3(8):245–255, March 1989.

[4] K. Chaudhary A. Srinivasan and E. S. Kuh. Ritual: A Performance-driven Placement Algorithm. *IEEE Transactions on Circuits and Systems -II*, 11(39):825–840, November 1992.

[5] H. Vaishnav and Massoud Pedrarn. PCUBE: A Performance Driven Placement Algorithm for Low Power Design. *IEEE Design Automation Conference, with Euro-VHDL*, pages 72–77, 1993.

[6] Glenn Holt and Akhilesh Tyagi. GEEP: A Low Power Genetic Algorithm Layout System. *IEEE 39th Midwest Symposium on Circuits and Systems*, 3:1337–1340, August 1996.

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

[8] Massoud Pedram. CAD for Low Power: Status and Promising Directions. *IEEE International Symposium on VLSI Technology, Systems and Applications*, pages 331–336, 1995.

[9] Srinivas Devadas and Sharad Malik. A Survey of Optimization Techniques Targeting Low Power VLSI Circuits. *32nd ACM/IEEE Design Automation Conference*, 1995.

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