Performance and Low Power Driven VLSI Standard Cell Placement using Tabu Search

Sadiq M. Sait Mahmood R. Minhas Junaid A. Khan

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

Abstract - We engineer a well-known optimization technique namely Tabu Search (TS) [1] for the performance and low power driven VLSI standard cell placement problem [2], [3]. 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 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 Adaptive-bias Simulated Evolution (SimE) approach reported in [4]. The comparison shows a significant improvement over the SimE approach.

I. Introduction

The need for low power driven VLSI design has emerged rapidly in past few years. While optimizing a circuit design for power consumption, other design objectives like performance and interconnect wire length need also to be taken care of. This fact leads to the development of techniques, which target to simultaneously optimize all these design goals. Previously, the objectives of optimizing interconnect wire length and performance were focused, and a large number of efforts targeting either one or both of above two objectives are reported in the literature [5], [6]. There has been reported some work for optimizing power consumption while considering the wire length and performance as constraints [7], [8]. Recently, some efforts targeting simultaneous optimization of all three objectives are also reported in [9], [10].

VLSI design is a complex process and is carried out at certain abstraction levels [2]. The design process starts from an abstract idea, and then each intermediate step continues refining the design and the process ends with the fabrication of a new chip. The problem of power optimization can be addressed at a higher level as well as at a lower level e.g., physical level [11].

In this work, we address the above problem in the placement step at the physical level. 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. The VLSI cell placement problem can be stated as follows: Given a collection of cells or modules with ports (inputs, outputs, power and ground pins) on the boundaries, the dimensions of these cells (height, width, etc), and a collection of nets (which are sets of ports that are to be wired together), *placement* problem consists of finding suitable physical locations for each cell on the layout [2]. By suitable we mean those locations that minimize given objective functions, subject to some constraints imposed by the designer, the implementation process, or layout strategy and the design style.

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

II. Problem Formulation and Cost Functions

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

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.*

A. Cost Functions

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

A.0.a 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).

A.0.b 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:

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

A.0.c 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:

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

A.0.d 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:

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

A.0.e 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.

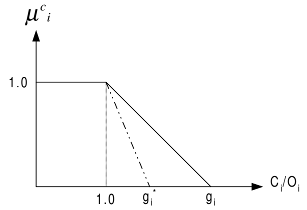


Fig. 1. Membership functions

The above rule is translated to *and-like* OWA fuzzy operator [12] and the membership $\mu(x)$ of a solution x in fuzzy set *GOOD solution* is given as:

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

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.

III. Tabu Search for Performance and Low Power Driven VLSI Placement

In this section, we first briefly describe TS algorithm and then discuss the details of our TS approach for multiobjective VLSI Placement. Tabu Search is an elegant heuristic that proceeds by making iterative perturbations while preventing cycling to certain number of recently visited points in search space. First, An initial solution is constructed and a certain number of trial moves are made to generate a set of neighbor solutions called *candidate list*. The best individual from the list is accepted and some characteristic of the move that resulted in this best solution is stored in a *tabu list*. Then, at each subsequent iteration, same number of trial moves are made but before accepting the move leading to the best neighbor solution, it is checked in tabu list. If it is not matched with any entry in the list, it is accepted and the tabu list is updated. Otherwise, the best trial solution is checked against *aspiration criterion* and if passed it is accepted and tabu list is updated. If the aspiration criterion is failed, the best trial solution is discarded and heuristic proceeds to next iteration. A commonly used aspiration criterion is that the best trial solution is the best solution seen so far. Other aspiration criteria can also be used. The structure of TS used in this work is shown in Figure 2.

```
Algorithm Tabu_Search
Ω    :  Set of feasible solutions
S    :  Current solution
S*   :  Best solution
Cost:   Objective function
N(S):   Neighborhood of S ∈ Ω
V*   :  Sample of neighborhood solutions
T    :  Tabu list
AL   :  Aspirartion level
Begin
Start with random initial solution S ∈ Ω
Initialize tabu list and aspiration level
For fixed number of iterations Do
    Generate neighbor solutions V* ⊂ N(S) by swapping
        two randomly chosen cells
    Find best S* ∈ V*
    If move S to S* is not in T Then
        Accept move and update best solution
        Store index of a swapped cell in T
    Else
        If Cost(S*) < AL Then
            Accept move and update best solution
            Store index of a swapped cell in T
        End If
    End If
End For
End.
```

Fig. 2. Tabu Search Algorithm for Multiobjective VLSI Cell Placement.

A. 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

These negative numbers are used for encoding purpose 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.

In the initialization step random encoded strings are generated. For encoding purpose we use a square grid having L slots, such that $L > N$, where N is the number of cells in the circuit. Each cell is assigned a positive integer value. Also, $L - N$ dummy cells are created, each dummy cell is assigned a negative integer in such a way that not two dummy cells have the same value. To generate the string, first row of the grid is placed first in the string followed by the next row and so on.

B. 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. A conventional approach to this problem is the use of weighted sum. In this approach, the costs of all the objectives are first normalized, then multiplied with a certain weight co-efficient, and finally added to obtain an overall cost. The weights are to taken in such a way that their sum is always 1. This approach is not used in our implementation because it is known to have certain problems. For instance, it is difficult to find values for weights as these heavily affect the relative importance of objectives.

Fuzzy logic provides a convenient alternative to weighted sum 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.

C. 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.

D. 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,

TABLE I
Comparison between costs of the best solutions generated by TS and SimE

	TS				SimE			
Circuit	L (μm)	P	D (ps)	T(s)	L (μm)	P	D (ps)	T(s)
s298	4888	947	131	15	7130	1395	152	21
s386	7264	1815	195	29	11167	2544	221	33
s832	19869	4699	350	77	28537	6577	485	114
s641	12669	2920	673	403	13773	3107	687	264
s953	30878	4929	223	97	33484	5523	250	130
s1238	44516	13402	376	272	45140	13870	397	295
s1196	40132	12104	345	430	41861	12918	357	433
s1494	63838	15604	727	248	67944	16091	809	279
s1488	66371	16303	724	210	73696	17511	891	216
s3330	190007	25640	422	3951	193731	25373	558	5610
s5378	316361	52324	337	8778	365204	56001	441	11369

then accept the current solution as new best solution by overriding the tabu restriction and update the tabu list.

IV. Experimental Results and Discussion

We have experimented with TS and SimE on number of ISCAS-89 benchmark circuits. Table I compares the quality of final solution generated by TS and SimE. The circuits are listed in order of their complexity. 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 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 settings for TS parameters like neighborhood size and tabu list size, which produced the shown results are discussed above in section 3. The platform used is an IBM compatible PC with an Intel Pentium-III 600Mhz CPU and 256MB RAM.

From the results, it is clear that TS performs better than SimE for all the circuits in terms of quality of solution. The execution time of TS is smaller than that of SimE in all the cases except s641, for which TS had to be run for longer time to beat the results obtained from SimE.

Figures 3 (a) and (b) show the trend of overall fuzzy membership of the current solution against execution time for TS and SimE respectively, in case of test case s3330. It can be seen that TS is more directed in term of solution membership as compared to SimE, which exhibits a random trend in the membership of the current solution. Also, it should be noted that TS is able to reach a membership value above 0.7 within 4000 seconds while SimE could not reach this value even after running for more than 5000 seconds. This shows the superiority of TS approach over SimE approach.

V. Conclusions

In this work, we have engineered Tabu Search (TS) algorithm 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 TS approach in terms of quality of the final solution obtained as well as the execution time. As early results have indicated superiority of SimE over Simulated Annealing [13], hence our TS approach proves to be better than SA.

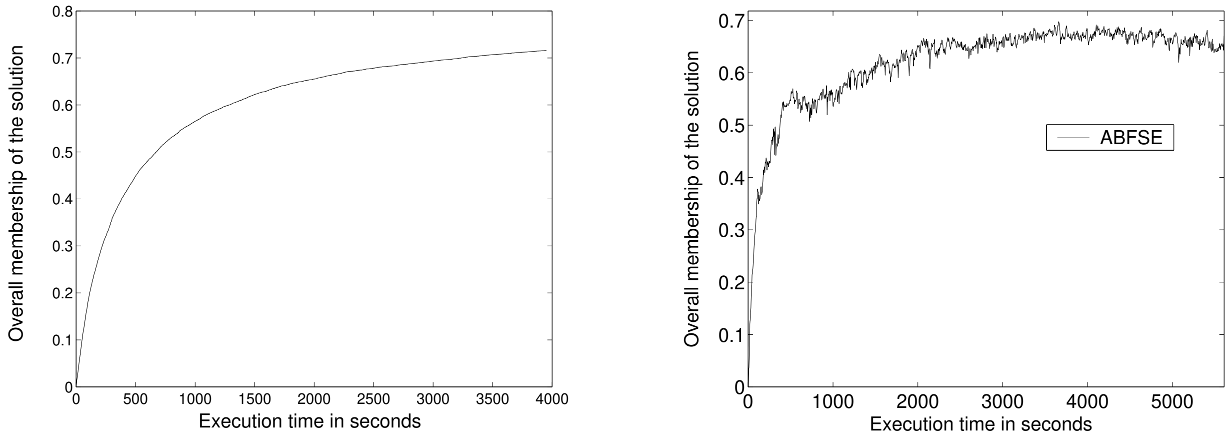
Acknowledgment:
Authors thank King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, for support under project # COE/ITERATE/221.

References

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

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

[3] Mahmood R. Minhas. Iterative Algorithms for Timing and Low Power Driven VLSI Standard Cell Placement. Master's thesis, Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Saudi Arabia, Dhahran 31261, June 2001.



(a) (b)

Fig. 3. (a) and (b) Membership values of the current solutions versus execution time for TS and SimE respectively in case of circuit s3330.

[4] Habib Youssef, Sadiq M. Sait, and Ali Hussain. Adaptive Bias Simulated Evolution Algorithm for Placement. *IEEE International Symposium on Circuits and Systems*, pages 355–358, May 2001.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Sadiq M. Sait, Habib Youssef, and Juaid A. Khan. Fuzzy Evolutionary Algorithm for VLSI Placement. *Genetic and Evolutionary Computer Conference 2001, GECCO-2001*, July 2001.

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

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

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