FAST FORCE-DIRECTED/SIMULATED EVOLUTION HYBRID FOR MULTIOBJECTIVE VLSI CELL PLACEMENT

Sadiq M. Sait

Dept. of Computer Engineering,
King Fahd University of Petroleum & Minerals,
Dhahran-31261, Saudi Arabia.
email: sadiq@ccse.kfupm.edu.sa

Junaid Asim Khan

Dept. of Electrical & Computer Engineering,
The University of British Columbia,
Vancouver, Canada.
email: junaidk@ece.ubc.ca

ABSTRACT

VLSI Standard Cell Placement is a hard optimization problem, which is further complicated with new issues such as power dissipation and performance. In this work, a fast hybrid algorithm is designed to address this problem. The algorithm employs Simulated Evolution (SE), an iterative search heuristic that comprises three steps: evaluation, selection and allocation. Solution quality is a strong function of the allocation procedure which is both time consuming and difficult. In this work a force directed approach in the allocation step of SE is used to both accelerate and improve the solution quality. Due to the imprecise nature of design information at the placement stage, objectives to be optimized are expressed in the fuzzy domain. The search evolves towards a vector of fuzzy goals. The proposed heuristic is compared with a previously presented SE approach. It exhibits significant improvement in terms of runtime for the same quality of solution.

1. INTRODUCTION

In VLSI physical design, the standard cell placement step consists of assigning modules (typically several thousands) to locations on the silicon surface under numerous design constraints while trading-off several objectives. In general, placement in VLSI physical design is a multiobjective optimization problem [1]. The most important objectives are power dissipation, delay, wirelength and area (width) of the chip [2, 3]. Several attempts using SE (Simulated Evolution) as a search heuristic and fuzzy logic to cope with the multiobjective nature of the problem have been attempted [4, 5, 6]. However, these exhibited unreasonably large runtime requirements.

In this paper, we revisit the algorithm from [6] and show how it can be accelerated in order to run on large circuits in a reasonable amount of time. The basic SE algorithm comprises three steps: *evolution*, *selection*, and *allocation*. Of these, the allocation step is the slowest, with a complexity of $O(n^2)$. In this paper we integrate a new allocation scheme into SE based on a force-directed algorithm that has a complexity of $O(n)$. We show that this hybrid approach gives results that are comparable (or of better quality especially for large test cases) but with much smaller runtimes [6].

The paper is organized as follows. Section 2 covers the problem formulation. In Section 3 the proposed algorithm is discussed. Experimental results are presented in Section 4 and conclusions in Section 5.

2. PROBLEM FORMULATION

SE has proved to be an excellent heuristic for standard cell placement problem in terms of solution quality. However, in terms of runtime, it is not as efficient as other well known deterministic placement algorithms. In this paper we have targeted the problem of decreasing the runtime of SE without degrading the overall solution quality. After inspecting the previous SE-based techniques it is observed that the main time consuming step is the allocation step [7, 4, 6]. The asymptotic time complexity of this step in these algorithms is $O(n^2)$. The runtime of SE can be considerably decreased if the allocation step is modified to consume only $O(n)$ time. In this work, we present a fuzzy, forced-directed approach to solve allocation in $O(n)$ time. For comparison we target the same standard cell placement problem and cost function presented in [1, 4, 6].

3. PROPOSED ALGORITHM

In this section we describe our Simulated Evolution based hybrid search algorithm. We begin with a brief discussion of the basic SE heuristic.

3.1. Basic Simulated Evolution (SE)

The general SE algorithm is illustrated in Figure 1 and comprises three main steps: **evaluation**, **selection**, and **alloca-**

ALGORITHM $Simulated\_Evolution(B, \Phi_{initial}, StoppingCondition)$
NOTATION
B= Bias Value. Φ= Complete solution.
m_i= Module i. g_i= Goodness of m_i.
$ALLOCATE(m_i, \Phi_i)$=Function to allocate m_i in partial solution Φ_i
Begin
Repeat
EVALUATION:
ForEach $m_i \in \Phi$ evaluate g_i;
/* Only elements that were affected by moves of previous */
/* iteration get their goodnesses recalculated*/
SELECTION:
ForEach $m_i \in \Phi$ **DO**
begin
IF $Random > min(g_i, 1)$
THEN
begin
$S = S \cup m_i$; Remove m_i from Φ
end
end
Sort the elements of S
ALLOCATION:
ForEach $m_i \in S$ **DO**
begin
$ALLOCATE(m_i, \Phi_i)$
end
Until *Stopping Condition is satisfied*
Return Best solution.
End $(Simulated\_Evolution)$

Fig. 1. Structure of the Simulated Evolution algorithm [7].

tion. In the **evaluation** step the **goodness** of each cell in its current location, in the range $[0, 1]$, is computed using some measure.

In the **selection** step, the algorithm probabilistically selects unfit elements. Elements with low goodness values have higher probabilities of getting selected for relocation. These selected elements are identified as the selection set and are removed from the solution. These selected elements are one by one reassigned to new locations in a constructive **allocation** step. The objective of this step is to improve their goodness values, thereby reducing the overall cost of the solution.

Different constructive allocation schemes are proposed in literature [8, 7]. One such scheme is **sorted individual best fit**, where all the selected elements are sorted in descending order with respect to their connectivity with the partial solution and placed in a queue. The sorted elements are removed one at a time and *trial* moves are carried out for all the available empty positions. The element is *finally* placed in a position where maximum reduction in cost for the partial solution is achieved. This process is continued until the selected queue is empty. The overall complexity of this algorithm is $O(n^2)$ where n is the number of selected elements. Other more elaborate schemes are **weighted bipartite matching allocation** and **branch-and-bound search allocation** [8]. However, these allocation strategies are more complex than "sorted individual best fit", while the quality of solution remains comparable [8]. In summary, selection and allocation steps determine and dictate the search strategy, while evaluation provides feedback to the search scheme.

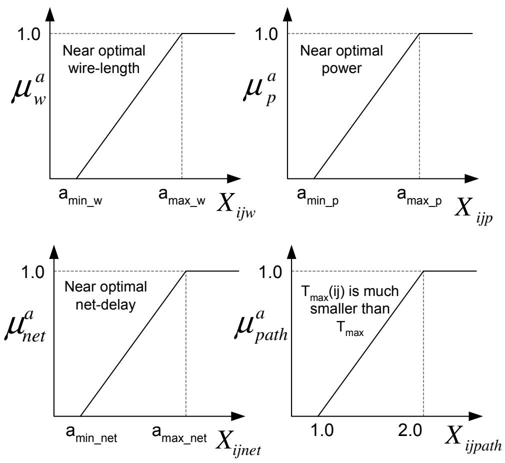

Fig. 2. Membership functions used in fuzzy allocation.

The main contribution in this paper is a new $allocation$ scheme; this will be discussed in Section 3.2. However, the evaluation and selection schemes are same as in [6], except that OWA-operators for fuzzy aggregation are replaced by the new fuzzy aggregating functions proposed in [9].

3.2. Fuzzy Force Directed Allocation

In the allocation stage, the selected cells are to be reassigned to best available locations. We consider selected cells as movable modules and remaining cells as fixed modules. In previous works [4, 5, 6], **sorted individual best fit** scheme was employed. For large circuits the run time was unreasonably high. To address this problem, a force directed allocation is proposed in this work. According to this approach optimal x-position and y-position of the cell under consideration are found. The y-position indicates the row to which the cell should be relocated. If the y-position is in between two rows then the row nearest to y-position is selected. In order to satisfy the width constraint, if the width of selected row after adding the cell is more than the maximum allowable width then the next nearest row that satisfies the width constraint is chosen. The x-position indicates the exact location of the cell in the selected row.

The basic idea behind the force directed method is that cells connected by a net exert forces on each other. Suppose a cell a is connected to another cell b by a net of weight w_{ab}. Let d_{ab} represents the distance between a and b. Then the force of attraction between the cells is proportional to the product $w_{ab} \times d_{ab}$. A cell i connected to several cells j at distance d_{ij} by wires of weights w_{ij}, experiences a total force F_i given by

$$F_i = \sum_j w_{ij} \cdot d_{ij} \quad (1)$$

The best location for a cell i is where the x-component

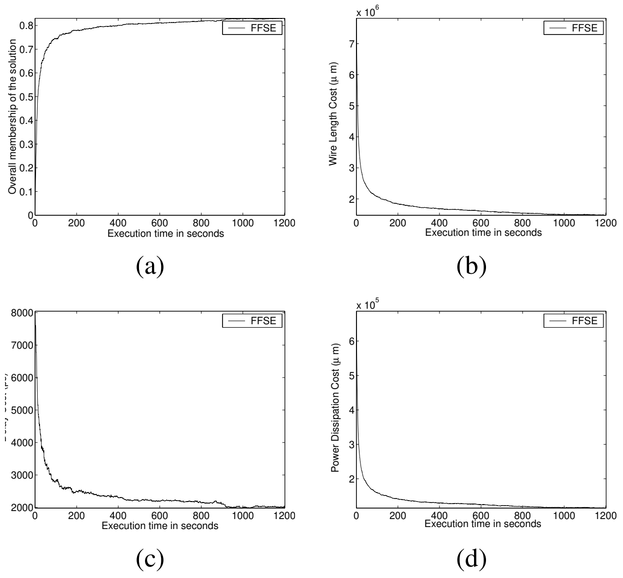

Fig. 3. (a) Overall membership; (b) Wirelength; (c) Delay; and (d) Power; versus execution time in Seconds (S15850).

and y-component of F_i are both zero. We can write these conditions as follows,

$$\sum_j w_{ij} \cdot (x_j - x_i) = 0; \quad \& \quad \sum_j w_{ij} \cdot (y_j - y_i) = 0 \quad (2)$$

Solving the above equations for x_i and y_i we have

$$x_i = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}} \qquad y_i = \frac{\sum_j w_{ij} \cdot y_j}{\sum_j w_{ij}} \quad (3)$$

Values x_i and y_i are the optimal x-position and y-position for a cell i with respect to current x and y positions of all the cells j connected to it [1]. They point to the new location that is better in terms of all objectives. For this purpose, proper weights to each of the nets connecting cell i and cell j are to be chosen. A good way to choose these weights is to use fuzzy logic. The following fuzzy rule is used to find these weights:

Rule R1: IF a net is *good in wire-length* AND *good in power* AND *good in delay* **THEN** it has a low weight.

According to this rule, a net will have a smaller weight only if it is good in terms of all the objectives. In fact weight signifies a badness factor (opposite of goodness in evaluation). The cell will try to move in the directions of those nets that have higher weight (higher badness). We use the membership functions in Figure 2, with the following base values.

$$X_{ijw}(x) = \frac{l^*_{ij}}{l_{ij}} \quad X_{ip}(x) = \frac{l^*_{ij}}{(1 + S_{ij})\, l_{ij}}$$
$$X_{inet}(x) = \frac{ID^*_{ij}}{ID_{ij}} \quad X^e_{ipath}(x) = \frac{T_{max}}{T_{max}(ij)} \quad (4)$$

where l_{ij} represents wire-length of a net ij and l^*_{ij} is its estimated lower bound. S_{ij} is its switching probability (required to estimate power in CMOS circuits). ID_{ij} is interconnect delay of net ij and ID^*_{ij} is its estimated lower bound. T_{max} is the delay of longest path and $T_{max}(ij)$ is the delay of longest path traversing net ij.

Using these base values and corresponding μ^a_i where superscript a denotes allocation, we find a goodness factor g_{ij}, using AFA and OFA operators proposed in [9], for the net connecting cells i and j, as follows,

$$g_{ij} = \mu^a_{ij} = 1 - \frac{\sum_{k=w,p,d} \bar{\mu}^{a\,2}_{k,ij}}{\sum_{j=w,p,d} \bar{\mu}^{a}_{k,ij}} \quad (5)$$

where
$$\mu^a_{d,ij} = \frac{\mu^{a\,2}_{net,ij} + \mu^{a\,2}_{path,ij}}{\mu^a_{net,ij} + \mu^e_{path,ij}} \quad (6)$$

Now the weight of the net w_{ij} is calculated as follows,

$$w_{ij} = 1 - g_{ij} \quad (7)$$

In this proposed allocation schemes it is clear that for each cell we have to find the best location only once, therefore the complexity of the proposed allocation scheme is $O(n)$ where n is the number of cells selected in selection stage of the algorithm. All other issues such as already occupied zero-force locations, cells already in their zero-force locations, etc., are resolved using the previous ad-hoc approaches available in the literature [1].

4. EXPERIMENTS AND RESULTS

Fast Fuzzy Force Directed Simulated Evolution (FFSE) and Biasless Fuzzy Simulated Evolution (BLFSE) [6], were applied on 12 ISCAS benchmark circuits. In BLFSE, execution is aborted when no improvement is observed in the last 500 iterations (maximum of 5000 iterations), whereas for FFSE the algorithm is run for a fixed 5000 iterations. The 0.25 micron CMOS digital low power standard cell library for MOSIS is used.

Table 1 compares the quality of the final solution generated by BLFSE and FFSE. The circuits are listed in order of their size (136-10383 modules). From the results, it is clear that FFSE outperformed BLFSE for all circuits in terms of execution time. Also, in most cases, FFSE shows minimal degradation in terms of solution quality but with a significant improvement in runtime. For larger circuits (S3330 & S5378), FFSE performed better than BLFSE in terms of both the final solution quality and the runtime.

Observe that the algorithm converges very fast. This behavior can be observed in Figure 3, where convergence is achieved after approximately 400 seconds (6.6 minutes), and the remaining time is spent in fine tuning the solution

Circuit		BLFSE				FFSE			
	# of Cells	L (μm)	P (μm)	D (ps)	T (s)	L (μm)	P (μm)	D (ps)	T(s)
S298	136	4548	915	139	46	4975	999	135	4.8
S386	172	8357	2036	203	117	9422	2169	213	6.8
S832	310	23140	5251	416	192	26112	5863	400	11
S641	433	12811	3072	687	175	12485	2897	674	24
S953	440	29576	5025	223	351	29988	4683	244	17
S1238	540	41318	12303	363	699	41362	12934	377	20
S1196	561	35810	11276	360	613	38282	12363	350	22
S3330	1961	183288	24797	459	5351	163756	24112	483	87
S5378	2993	326840	48360	435	11823	243721	41560	376	149
S9234	5844	UH	UH	UH	UH	655370	114231	908	440
S13207	8651	UH	UH	UH	UH	1339837	144189	1604	885
S15850	10383	UH	UH	UH	UH	1477662	115049	2006	1202

Table 1. Layout found by BLFSE, and FFSE. 'L', 'P' and 'D' represent the wire-length, power, and delay costs respectively. 'T' is the execution time in Seconds. The last 3 circuits were not tested for BLFSE because of large runtime requirement. 'UH' indicates Unreasonably High runtime requirement.

quality. It is also obvious from these results that FFSE totally avoids early random walk, which is a problem in other non-deterministic heuristics such as Simulated Annealing. Memory requirement of SE is also considerably low when compared to other iterative heuristics such as Genetic algorithm as SE keeps only one solution in its memory at a time.

5. CONCLUSION

In this paper, we have proposed Fuzzy Force Directed Simulated Evolution Algorithm for multiobjective VLSI standard cell placement. The allocation stage is improved by using a force directed strategy to reduce the execution time from $O(n^2)$ in previous SE based approaches to $O(n)$.

Fuzzy logic is used to overcome the multiobjective nature of the problem. It is employed at evaluation and allocation stages and in the choice of the best solution from the set of generated solutions.

The proposed scheme is compared with BLFSE. It is observed that FFSE performs much better than BLFSE in terms of runtime, with no significant degradation in solution quality. FFSE can be used for large circuits whereas BLSFE performs poorly for circuits with more than $2000 - 3000$ cells.

Acknowledgment: The authors and the research team acknowledge King Fahd University of Petroleum & Minerals for its support under research project COE/ITERATE/221.

6. REFERENCES

[1] Sadiq M. Sait and Habib Youssef. VLSI Physical Design Automation: Theory and Practice. *World Scientific Publications, Singapore*, 2001.

[2] Glenn Holt and Akhilesh Tyagi. EPNR: An Energy-Efficient Automated Layout Synthesis Package. *IEEE International Conference on VLSI in Computers and Processors*, pages 224–229, October 1995.

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

[4] Sadiq M. Sait, Habib Youssef, and Junaid A. Khan. Fuzzy Evolutionary Algorithm for VLSI Placement. *GECCO-2001*, July 2001.

[5] Sadiq M. Sait, Habib Youssef, Junaid A. Khan, and Aiman Al-Maleh. Fuzzy Simulated Evolution for Power and Performance Optimization of VLSI Placement. *INNS-IEEE, IJCNN2001*, July 2001.

[6] Junaid A. Khan, Sadiq M. Sait, and M. R. Minhas. Fuzzy Biasless Simulated Evolution for Multiobjective VLSI Placement. *IEEE Congress on Evolutionary Computation, CEC2002, Honolulu*, May 2002.

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

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

[9] Junaid A. Khan and Sadiq M. Sait. Fuzzy Aggregating Functions for Multiobjective VLSI Placement. *FUZZ-IEEE 2002, Honolulu*, May 2002.