FAST FUZZY FORCE-DIRECTED/SIMULATED EVOLUTION METAHEURISTIC 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

Mixed Signal Systems & Verification,
EEsof Division,
Agilent Technologies Inc., U.S.A.,
email: junaid_khan@agilent.com

ABSTRACT

VLSI standard cell placement is the process of arranging circuit components (modules) on a silicon layout. The cell placement problem is a proven NP hard combinatorial optimization problem. The complexity of this problem increases when multiple optimization objectives are considered simultaneously. In this paper, a novel technique is presented to address this hard problem, while optimizing multiple objectives. A major difficulty with such multi-objective combinatorial optimization problems is the existence of a very large solution search space, one of which is the desired optimal solution. Simulated Evolution (SE) a general iterative heuristic is used to traverse the large search space, while fuzzy logic is resorted to assist in multi-criteria decision making and overcome the imprecise nature of design information at placement stage. New fuzzy aggregation functions are proposed. SE is hybridized with force directed algorithm to speed-up the search. The proposed schemes are compared with previously presented SE based heuristics. The implementations exhibit considerable improvement in terms of both solution quality and runtime.

1. INTRODUCTION

VLSI (Very Large Scale Integration) is a technology used to implement large circuits in silicon. These large circuits are normally formed of a million or more transistors. Due to the complexity of VLSI circuits with respect to the number of transistors, designing them is a complex task. In order to overcome the complexity of design process, it is divided into several intermediate levels [1]. One of the levels or stages of the design process is the physical design stage. This stage is further divided into stages like circuit partitioning, floorplanning, placement, grid routing, global routing and channel routing. Each of the above mentioned steps is a proven NP hard combinatorial optimization problem. The work presented here deals with the placement stage, thus we limit our introduction to the problem of VLSI standard cell placement.

The VLSI standard cell placement step consists of assigning modules (typically several thousands) to locations on the silicon surface while respecting the numerous design constraints and achieving the desired objectives. In general the placement step of the VLSI physical design stage is a multiobjective optimization problem [1]. The most important objectives are interconnect delay, and total wiring length. Other objectives include power dissipation, and area (width) of the chip [2, 3].

Due to the computational complexity of the problem, it is practically not possible or feasible to find an optimal placement solution in polynomial time using deterministic algorithms. Problems of these kind can be solved using iterative heuristic algorithm. These algorithms achieve sub–optimal solutions, or sometimes have shown to achieve even optimal solutions in polynomial time durations. Several general iterative heuristics like tabu search, genetic algorithms and simulated annealing [4, 5, 6, 7] have been proposed to solve this problem. A recently invented heuristic algorithm called Simulated Evolution (SE) is one such general iterative algorithm which is very efficient in solving combinatorial optimization problems like the placement problem and has been used on several previous occasions for solving hard combinatorial optimization problems [7, 8, 9, 10, 11, 12, 13]. Iterative heuristics have high runtime requirements and also require fine tuning of parameters which are hard to predict. In this paper we present a novel method to use SE for multiobjective placement problem with linear time complexity and without the need for fine tuning any parameter.

In this section, we present a brief introduction to fuzzy logic, which is used to express heuristic knowledge and/or to combine conflicting objectives. Fuzzy logic is a branch of mathematics invented by Lotfi Zadeh to represent and manipulate fuzzy knowledge, and to infer from it crisp outcomes [14, 15, 16]. Fuzzy logic provides a methodology to map values of different criteria into linguistic variables. Approximate reasoning can be made based on these linguistic variables and their values. The decision making in fuzzy logic approach mimics the decision making approach in humans. A formal explanation of the fuzzy logic technique, terms and terminology relevant to the problem under consideration will be presented in later sections.

The paper is organized as follows. Section 2 covers the problem formulation and cost estimation models. In Section 3, fuzzy logic for VLSI cell placement is present. Also discussed are two new fuzzy aggregating functions proposed and employed in place of classical fuzzy operators. In Section 4, the structure of the general SE algorithms is discussed. An improved version of SE algorithm that uses force-directed algorithm is presented. Experimental results are presented in Section 5 and conclusions in Section 6.

2. PROBLEM FORMULATION

The placement problem can be stated as follows: Given a set of modules (cells) $M = \{m_1, m_2, \cdots, m_n\}$, and a set of signals $V = \{v_1, v_2, \cdots, v_k\}$, each module $m_i \in M$ is associated with a set of signals V_{m_i}, where $V_{m_i} \subseteq V$. Also each signal $v_i \in V$ is associated with a set of modules M_{v_i}, where $M_{v_i} = \{m_j | v_i \in V_{m_j}\}$. M_{v_i} is called a signal net. Placement consists of assigning each module $m_i \in M$ to a unique location such that certain objectives are optimized and constraints are satisfied [1]. The objectives to be optimized are power dissipation, delay, and wire-length, while area (width) of the layout is considered as constraint. These objectives and constraint are estimated as follows [1, 13].

Estimation of Wire-length: The wire-length cost can be computed by adding wire-length estimates for all the nets in the circuit.

$$\text{Cost}_{wire} = \sum_{i \in M} l_i \tag{1}$$

where l_i is the wire-length associated with net v_i and M is the set of all cells in the circuit. This wire-length is computed using Steiner tree approximation.

Estimation of Power: Approximately 90% Power dissipation in CMOS logic is due to the dynamic (switching) power. Therefore a cost proportional to power dissipation can be estimated as

$$\text{Cost}_{power} = \sum_{i \in M} S_i l_i \tag{2}$$

where S_i is the switching activity at the output node of cell i.

Estimation of Delay: The cost function due to timing performance can be expressed as:

$$\text{Cost}_{delay} = T_{\pi_c} \tag{3}$$

where T_{π_c} is the delay of most critical path in the current iteration among the set of candidate paths

$$\{\pi_1, \pi_2, \pi_3, ..., \pi_k\}$$

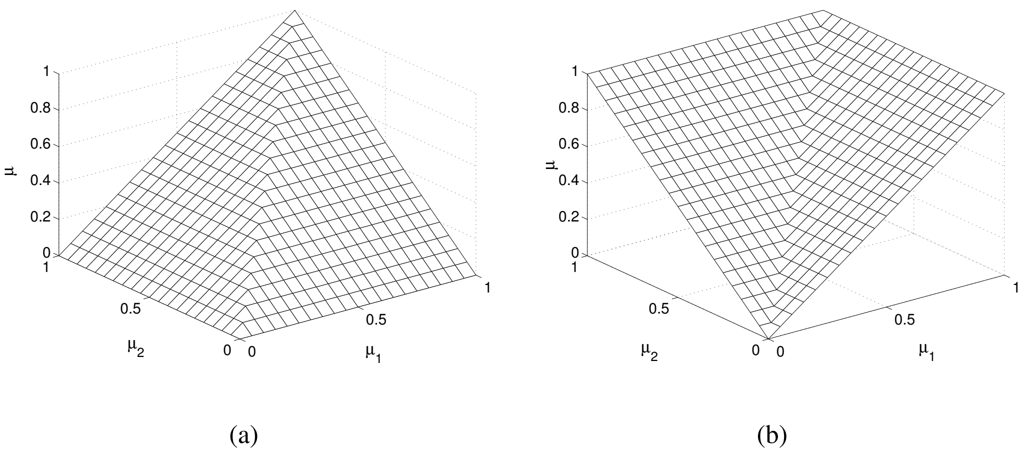


Fig. 1. (a) Fuzzy Min Operator. (b) Fuzzy Max Operator.

Layout Width: In this work layout width is considered as a constraint. The upper limit on the layout width is defined as:

$$Width_{max} = (1 + a) \times Width_{min} \tag{4}$$

where $Width_{max}$ is the maximum allowable width of the layout, and $Width_{min}$ is the lower bound on layout width. The parameter a denotes how wide the layout can be compared to the lower bound.

3. FUZZY LOGIC FOR VLSI PLACEMENT

A brief overview of the fuzzy logic concept was presented in Section 1. The details of fuzzy logic rules are explained in the following sections. A fuzzy logic rule is an **If-Then** rule. The **If** part (*antecedent*) is a fuzzy predicate defined in terms of linguistic values and fuzzy operators (**Intersection** and **Union**). The **Then** part is called the *consequent*. There are many implementations of fuzzy union and fuzzy intersection operators. Fuzzy union operators are known as **s-norm** operators while fuzzy intersection operators are known as **t-norm**. Generally, **s-norm** is implemented using $\max$ and **t-norm** as $\min$ function, i.e.,

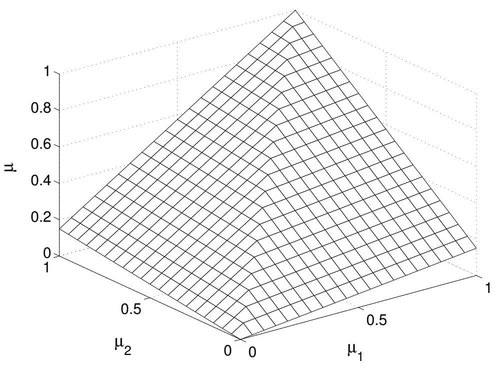
$$\mu_{A \cup B}(x) = \max\left(\mu_A(x), \mu_B(x)\right) \tag{5}$$

and

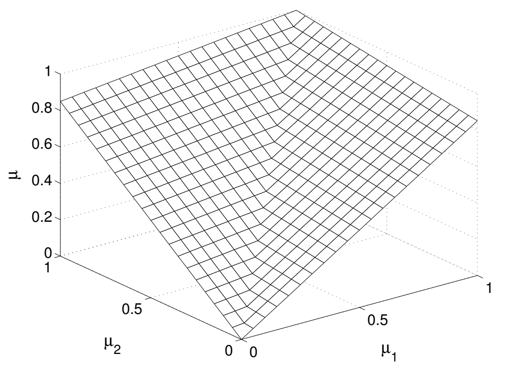
$$\mu_{A \cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right) \tag{6}$$

This is known as the $\min - \max$ logic initially introduced by Zadeh [14]. The graphical representation of these operators is given in Figure 1.

Formulation of multi-criteria decision functions do not desire pure "anding" of **t-norm** nor the pure "oring" of **s-norm**. The reason for this is the complete lack of compensation of **t-norm** for any partial fulfillment and complete submission of **s-norm** to fulfillment of any criteria. Min and Max operators do not provide such a compensation/submission as shown in Figure 1. For example in case of Min operator $min(0, 0.5) = 0$ and also $min(0, 0) = 0$, however, it is clear that the solution having individual memberships $(0, 0.5)$ is better than the solution having individual memberships$(0, 0)$, whereas min operator is not able to differentiate among these. Also the indifference to the individual criteria of each of these two forms of operators led to the development of Ordered Weighted Averaging (OWA) operators [17]. This operator falls in the category of compensatory fuzzy operators and allows easy adjustment of the degree of "anding" and "oring" embedded



(a)



(b)

Fig. 2. (a) Fuzzy And-like OWA Operator, (b) Fuzzy OR-like OWA Operator.

in the aggregation. According to [17], "orlike" and "andlike" OWA for two fuzzy sets A and B are implemented as given in Equations 7-8 respectively.

$$\mu_{A\bigcup B}(x) = \beta \times \max(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B) \tag{7}$$

$$\mu_{A\bigcap B}(x) = \beta \times \min(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B) \tag{8}$$

β is a constant parameter in the range [0,1]. It represents the degree to which OWA operator resembles the pure "or" or pure "and" respectively. However, it is difficult to select a suitable value of β without many trial runs of an optimization algorithm for each problem instance, because a suitable value of β is different for each problem instance. The graphical representation of OWA operators is shown in Figure 2.

It is clear from this figure that OWA operators provides compensation/submission, as $min(0, 0.5) > min(0, 0)$. However, there is a major drawback of using compensatory operators like OWA, because it might happen that these will optimize only a single objective. This effect can be seen in Figure 2(a) that aggregating membership of $(0, 1)$ and $(0.15, 0.15)$ are equal (0.15) for $\beta = 0.7$, however it is clear that the solution having individual memberships $(0, 1)$ has been obtained by optimization of single objective only without any effort in the optimization of other objective(s) and might not be acceptable compared to the solution having individual memberships $(0.15, 0.15)$.

In order to solve the problems of choosing the accurate value of β and undesired optimization of single objectives, a set of aggregating functions (AND like and OR like) is presented in this paper. These aggregating functions do not need any user specified parameter like β in OWA, and also provide the compensation/submission in a controlled manner and avoid accidental optimization of single objective.

3.1. Proposed Fuzzy Aggregating Functions

Two fuzzy aggregating functions, AND like fuzzy aggregation (AFA) and OR like fuzzy aggregation (OFA) are presented.

The And Like Fuzzy Aggregation (AFA) function operates on the membership values in the complementary fuzzy sets, instead of fuzzy sets itself. The details of this function are given below.

Let μ, μ_1 and μ_2 be the membership values in fuzzy sets S, S_1 and S_2. The membership $\bar{\mu}$ in $\bar{S}$ (the complementary fuzzy set of S) is obtained by using fuzzy complementary operator.

Now the *And Like Fuzzy Aggregation* (AFA) is defined as follows,

$$\bar{\mu} \quad = \quad \bar{w}_1\,\bar{\mu}_1 + \bar{w}_2\,\bar{\mu}_2 \tag{9}$$

$$\mu \quad = \quad 1 - \bar{\mu} \tag{10}$$

where

$$\bar{w}_n = \frac{\bar{\mu}_n}{\bar{\mu}_1 + \bar{\mu}_2} \tag{11}$$

If the membership value μ_1 in one fuzzy set S_1 is lower than other, then corresponding membership $\bar{\mu}_1$ in complementary fuzzy set $\bar{S}_1$ is higher than the other, resulting in higher weight $\bar{w}_1$, leading to higher membership $\bar{\mu}$ in resulting complementary fuzzy set $\bar{S}$. It results in the lower membership μ in the resulting fuzzy set S. This behavior is analogous to t-norm where, if one membership is low, then the resulting membership is also low. If the membership values in all complementary fuzzy sets are equal then equal weights are assigned and the resulting membership is high. In short, the AFA has following advantages.

1. It simulates the behavior of fuzzy AND logic (especially at the boundaries).
2. There is no need to adjust any parameter like β in OWA.
3. All the weights are controlled automatically.
4. It provides the compensation for any partial fulfillment.
5. It avoids the accidental optimization of single objective.
6. It rejects the solutions having diverse membership values in different fuzzy sets, that can be accepted in the case of "pure anding" and "andlike OWA".

Combining Equations 9, 10 and 11 and generalizing the function to n fuzzy membership values to be ANDed, we can define the AFA function as follows,

$$\mu = 1 - \frac{\sum_{i=1}^{n} \bar{\mu}_i^2}{\sum_{i=1}^{n} \bar{\mu}_i} \tag{12}$$

The Or Like Fuzzy Aggregation (OFA) function is analogous to s-norm in behavior. Unlike AFO it receives directly the membership values. The function is defined as follows,

$$\mu = w_1 \, \mu_1 + w_2 \, \mu_2 \tag{13}$$

where

$$w_n = \frac{\mu_n}{\mu_1 + \mu_2} \tag{14}$$

If the membership in one fuzzy set is higher than the membership values in the other fuzzy sets then it will be given higher weight, hence the membership value μ in resulting fuzzy set S will be higher, that is analogous to s-norm. Unlike "pure oring" it also provides interaction from other membership functions having lower values.

Combining Equations 13 and 14 and generalizing the function to n fuzzy membership values to be ORed, we can define the OFA as follows,

$$\mu = \frac{\sum_{i=1}^{n} \mu_i^2}{\sum_{i=1}^{n} \mu_i} \tag{15}$$

Figure 3 shows the behavior of proposed fuzzy aggregating functions. Figure 3(a) shows the behavior of AFA. It can be seen that the functions operates as a min operator on the extremes and acts like a compensatory operator in the middle. Due to this fact, it is not possible to unintentionally optimize only a single objective (possible in OWA and not desirable), due to the compensation. It provides compensation in a controlled manner: when the membership values to be aggregated are near each other, it behaves as a compensatory function; however if these are diverse, indicating optimization of a single objective, then it behaves as a pure min and forces the optimization algorithm to optimize other objectives as well.

Figure 3(b) illustrates the behavior of OFA. It shows that the functions behaves as pure max in boundaries and also exhibits the effect due to submission of other membership values. However, it does not waste time in differentiating the degree of submission of a particular objective, because in OR logic if one objective is fulfilled then it is sufficient.

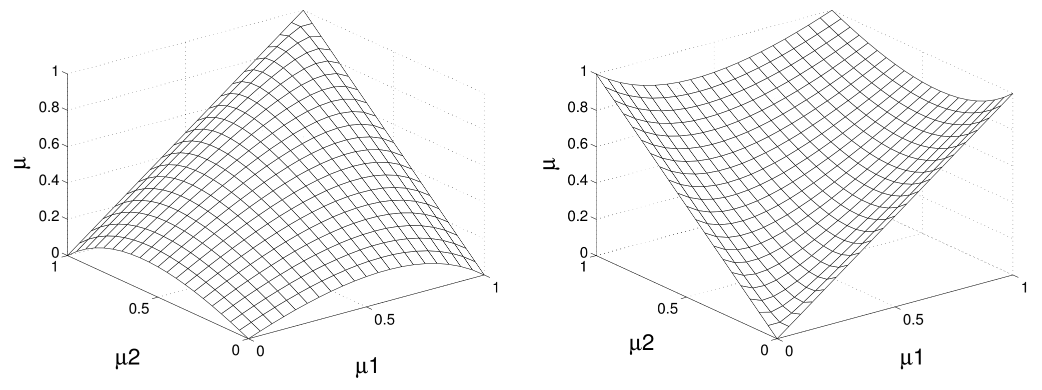


Fig. 3. (a) And Like Fuzzy Aggregation, (b) Or Like Fuzzy Aggregation.

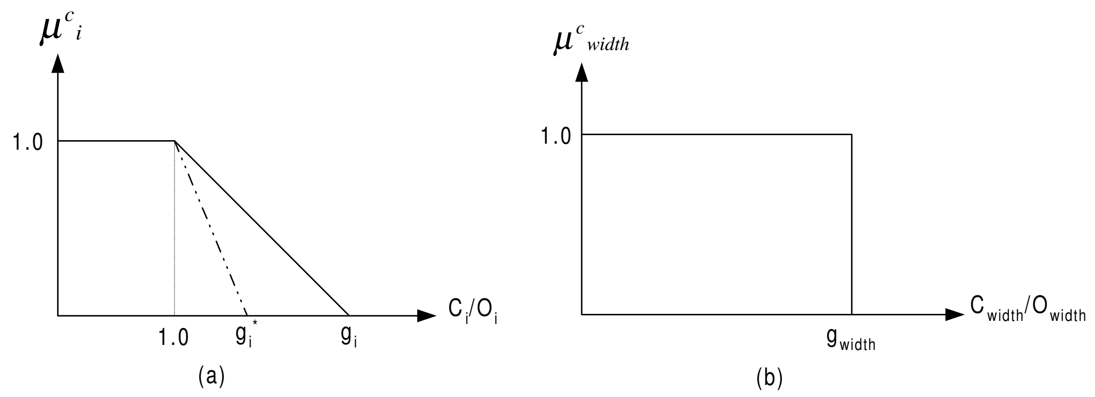


Fig. 4. Membership functions within acceptable range.

3.2. Applications of Fuzzy Aggregating Functions in Placement Problem

To combine three objectives and a constraint using the proposed aggregating functions, we use the fuzzy rule given below [13]:

Rule R1: IF a solution is within *acceptable wire-length* AND *acceptable power* AND *acceptable delay* AND *within acceptable layout width* **THEN** it is an acceptable solution.

Using the And like Fuzzy Aggregating Function (AFA), the above fuzzy rule translates to:

$$\begin{aligned} \mu^c_{pdw}(x) &= 1 - \frac{\sum_{j=p,d,w} \bar{\mu^c}^2_j(x)}{\sum_{j=p,d,w} \bar{\mu^c}_j(x)} \\ \mu^c(x) &= \min(\mu^c_{pdw}(x), \mu^c_{width}(x)) \end{aligned} \tag{16}$$

where $\mu^c_j(x)$ for $j = p, d, l, width$, are the individual membership values in the fuzzy sets *within acceptable wire-length, power, delay, and layout width* respectively. The superscript c represents “cost”. The solution that results in maximum value of $\mu^c(x)$ is reported as the best solution by the search heuristic.

The shape of membership functions for fuzzy sets *within acceptable power, delay and wire-length* are shown in Figure 4(a), whereas the constraint *within acceptable layout width* is given as a crisp set (Figure 4(b)). $O_i s$ for $i \in \{w, p, d, width\}$ represent the lower bounds for wire-length, power, delay and layout width respectively. Since layout width is a constraint, its membership value is either 1 or 0 depending on $goal_{width}$ (in our experiments $goal_{width} = 1.25$, which indicates that the maximum allowable width of the layout is $1.25 \times O_{width}$). However, for other objectives, by increasing or decreasing the value of $goal_i$ one can vary its preference in the overall membership function.

4. 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.

4.1. Basic Simulated Evolution (SE)

The general SE algorithm is illustrated in Figure 5 and comprises three main steps: **evaluation**, **selection**, and **allocation**.

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.

Allocation is the SE operator that has most impact on the quality of solution. Allocation takes as input the two sets, S and its complement, and generates a new solution S' which contains all the members of the previous solution, with the elements of S mutated according to an allocation function.

The choice of a suitable allocation function is problem specific. The decision of the allocation strategy usually requires more ingenuity on the part of the designer than the *Selection* scheme. The allocation function may be a non-deterministic function which involves a choice among a number of possible mutations (moves) for each element of S. Usually, a number of *trial-mutations* are performed and rated with respect to their *goodnesses*. Based on the resulting goodnesses, a final configuration of the population S' is decided. The goal of allocation is to favor improvements over the previous generation, without being too greedy.

The allocation operation is a complex form of genetic *mutation* which is one of the genetic operations thought to be responsible for the evolution of the various species in biological environments. The allocation function mutates the solution by altering the locations of the elements of the selected set S. However, since *mutation* is the only mechanism used by SE for inheritance and evolution, it must be more sophisticated than the one used in GA.

Different constructive allocation schemes are proposed in literature [18, 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 step 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** [18]. However, these allocation strategies are more complex than "sorted individual best fit", while the quality of solution remains comparable [18]. In summary, selection and allocation steps determine and dictate the search strategy, while evaluation provides feedback to the search scheme.

One of the contributions in this paper is a new $allocation$ scheme; this will be discussed in Section 4.2. However, the evaluation and selection schemes are same as those discussed in references [13], except that OWA-operators are replaced by the new fuzzy aggregating functions discussed earlier.

```
**ALGORITHM** $Simulated\_Evolution(B, \Phi_{initial}, StoppingCondition)$
*NOTATION*
$B$= Bias Value.     $\Phi$= 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 $\Phi_i$
**Begin**
**Repeat**
  *EVALUATION:*
        **ForEach** $m_i \in \Phi$ evaluate $g_i$;
        /* Only elements that were affected by moves of previous */
        /* iteration get their goodnesses re-calculated*/
  *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 $\Phi$
                    **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. 5. Structure of the Simulated Evolution algorithm [7].

4.2. Evaluation and Selection

Fuzzy Goodness Evaluation: A designated location of a cell is considered good if it results in short wire-length for its nets, reduced delay, and reduced power. These conflicting requirements can be expressed by the following fuzzy logic rule **R2**.

Rule R2: IF cell i is *near its optimal wire-length* AND *near its optimal power* AND (*near its optimal net delay* OR $T_{\max}(i)$ *is much smaller than* $T_{\max}$) **THEN** it has a high goodness.

where $T_{\max}$ is the delay of the most critical path in the current iteration and $T_{\max}(i)$ is the delay of the longest path traversing cell i in the current iteration.

With the AND and OR logic implemented as AFA & OFA, rule **R2** evaluates to the expression below:

$$\text{goodness}_i = \mu_i^e(x) = 1 - \frac{\sum_{j=w,p,d} \bar{\mu^e}^2_{ij}(x)}{\sum_{j=w,p,d} \bar{\mu^e}_{ij}(x)} \tag{17}$$

where

$$\mu_{id}^e(x) = \frac{\mu^{e2}_{inet}(x) + \mu^{e2}_{ipath}(x)}{\mu^e_{inet}(x) + \mu^e_{ipath}(x)} \tag{18}$$

The base values for fuzzy sets near optimal wire-length, power, net delay, and for the fuzzy set "$T_{\max}(i)$ much smaller than $T_{\max}$", for each cell, are represented by $X_{iw}(x)$, $X_{ip}(x)$, $X_{inet}(x)$ and $X_{ipath}(x)$, respectively [10]. Membership functions of these base values are shown in Figure 6.

Selection: In this stage of the algorithm, some cells are selected probabilistically depending on their goodness values. A cell i is selected if $Random > goodness_i$ where $Random$ is a Gaussian random number with $mean = G_m - G_\sigma$ and $standard\ deviation = G_\sigma$. G_m and G_σ are the mean and standard deviation of goodness values of cells in the initial solution [13].

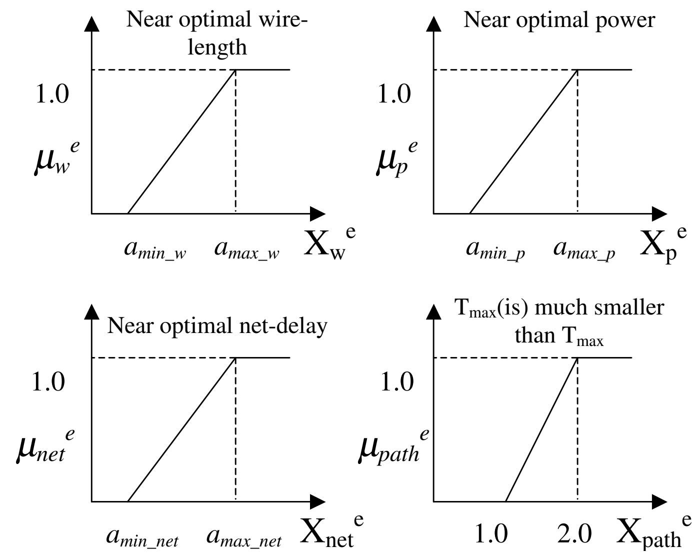


Fig. 6. Membership functions used in fuzzy evaluation.

4.3. 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 selected cells are sorted in descending order of their goodnesses with respect to their partial connectivity with unselected cells [19, 20, 7, 9, 10, 11, 12, 13].

One cell from the sorted list is selected at a time and best available location for it is found. The cell is placed at that position and removed from the selection set. The process, commonly known as **sorted individual best fit**, is repeated until the selection set is empty. The selected cell is actually moved from its current location to the location of another selected cell if the move results in the maximum gain. This procedure leads to allocation complexity of $O(n^2)$, where n is the number of cells selected in selection stage. All the other steps of SE algorithms have the complexity at most $O(n)$. Therefore allocation step is a bottleneck in terms of computational complexity of the algorithm.

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 [1]. 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

$$F_i = \sum_j w_{ij} \cdot d_{ij} \tag{19}$$

The best location for a cell i is where the x-component and y-component of F_i are both zero. We can write these conditions as follows,

$$\sum_j w_{ij} \cdot (x_j - x_i) = 0; \ \ \& \ \ \sum_j w_{ij} \cdot (y_j - y_i) = 0 \tag{20}$$

Solving the above equations for x_i and y_i we have

$$x_i = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}} \qquad y_i = \frac{\sum_j w_{ij} \cdot y_j}{\sum_j w_{ij}} \tag{21}$$

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 R3: 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). The shape of the membership functions for allocation is similar to those of evaluation (see Figure 6) with the following base values:

$$\begin{aligned} X_{ijw}(x) &= \frac{l^*_{ij}}{l_{ij}} \quad X_{ip}(x) = \frac{l^*_{ij}}{(1+S_{ij})\, l_{ij}} \\ X_{inet}(x) &= \frac{ID^*_{ij}}{ID_{ij}} \quad X^a_{ipath}(x) = \frac{T_{max}}{T_{max}(ij)} \end{aligned} \tag{22}$$

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, for the net connecting cells i and j, as follows:

$$g_{ij} = \mu^a_{ij} = 1 - \frac{\sum_{k=w,p,d} \bar{\mu^a}^2_{k,ij}}{\sum_{j=w,p,d} \bar{\mu^a}_{k,ij}} \tag{23}$$

where

$$\mu^a_{d,ij} = \frac{{\mu^a}^2_{net,ij} + {\mu^a}^2_{path,ij}}{\mu^a_{net,ij} + \mu^e_{path,ij}} \tag{24}$$

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

$$w_{ij} = 1 - g_{ij} \tag{25}$$

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 location, a cell already in its zero-force locations, etc., are resolved using the previous ad-hoc approaches available in the literature [1].

5. EXPERIMENTS AND RESULTS

Two comparison scenarios are considered to test the proposed work. The fuzzy aggregating functions are compared with OWA operator in the first scenario. In the second scenario the proposed allocation scheme is compared with the $O(n^2)$ allocation scheme proposed earlier [10, 11, 20, 19].

Circuit		AFSE				OFSE			
	# of Cells	L (μm)	P (μm)	D (ps)	T (s)	L (μm)	P (μm)	D (ps)	T(s)
S298	136	4853	925	139	82	4548	915	139	46
S386	172	7140	1653	202	153	8357	2036	203	117
S641	433	9445	2092	650	836	12811	3072	687	175
S953	440	28290	4394	236	344	29576	5025	223	351
S1238	540	36333	11329	382	566	41318	12303	362	699
S1494	661	52711	12824	763	575	54523	12986	768	762
S3330	1961	135650	17378	437	6619	183288	24797	460	5351
S5378	2993	207252	29432	341	19159	326840	48360	435	11823
S9234	5844	641670	101362	919	49479	857174	137712	923	42692

Table 1. Comparison between proposed aggregating functions and OWA. L is wire-length in μm, P is power cost in μm, D is delay in pico seconds, and T is the execution time in seconds.

5.1. Comparison of Fuzzy Aggregating Functions

The proposed allocation scheme presented in [13] is used in this test. Fuzzy Simulated Evolution using OWA (OFSE) and Fuzzy Simulated Evolution using proposed Fuzzy Aggregating Functions (AFSE) are applied on different ISCAS benchmark circuits [21]. In case of OFSE the OFA and AFA functions are replaced with OR-like OWA and AND-like OWA respectively.

Table 1 compares the quality of final solution generated using OFSE and AFSE. The circuits are listed in order of their size (136-5844 modules). It is clear that the proposed set of aggregating functions (AFSE) have performed better than OWA operators (OFSE), except for two smaller circuits. In most cases AFSE proved to be better in terms of all objectives, because of its better directed search capabilities in the solution space. However, in some cases, slight increase in the cost of one objective has resulted in larger decrease in cost of other objectives (for example, see S953). In general, AFSE performs better than OFSE in terms of quality of final solution.

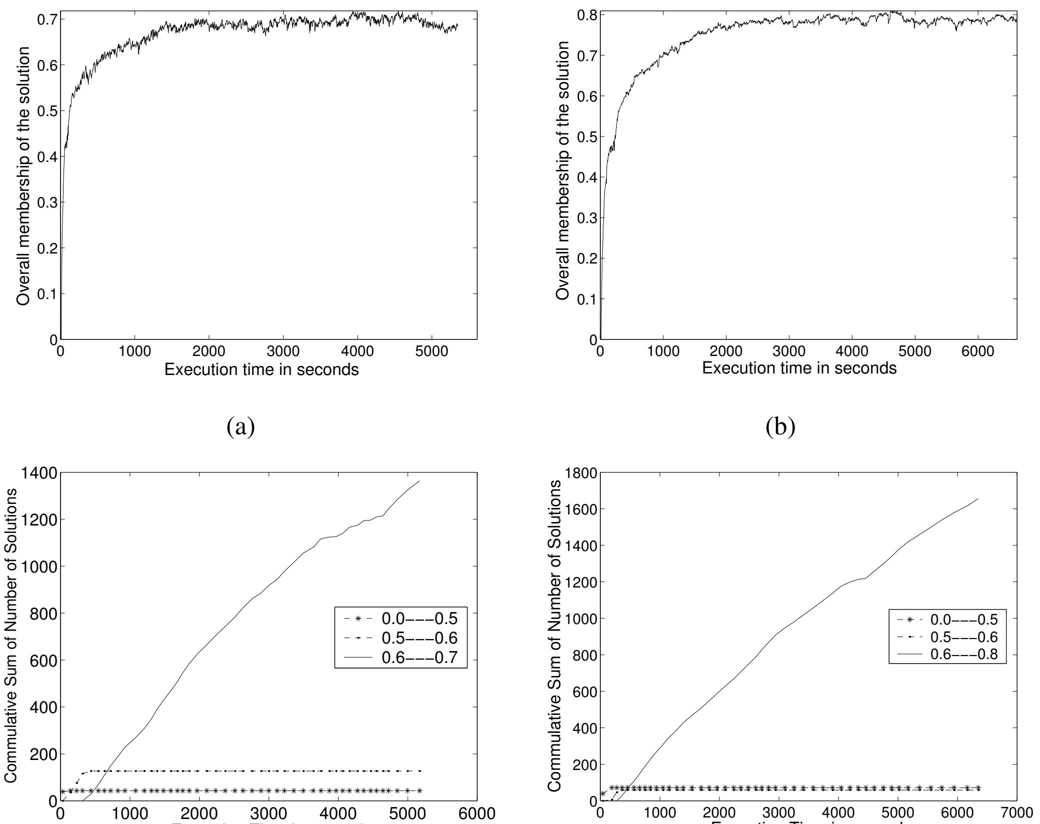
In order to compare improvement in the quality of solution versus time, the current membership values of the solution obtained by OFSE and AFSE (Figure 7(a) and (b)) are ploted. These plots are for test case S3330. It can be observed that the quality of solution improves rapidly in AFSE based search as compared to OFSE. This behavior was observed for all test cases.

Figures 7(c), and (d) track the total number of solutions found by OFSE and AFSE with respect to execution time, for various membership ranges. A key aspect to be noted is that the AFSE exhibited slightly faster evolutionary rate than OFSE. For example, after about 200 seconds, almost all new solutions discovered by AFSE have a membership more than 0.6 in the fuzzy subset of good solutions with respect to all objectives, and almost none were found with lower membership values. In contrast, for OFSE, it is after 300 seconds that the first solution with membership greater than 0.6 was found (see Figure 7). This behavior was observed for all test cases.

Individual costs versus execution time for both schemes have also been plotted in Figures 8-10. It can be seen that for objectives to be minimized, there is less randomness in the solution movement toward a better solution in case of AFSE as compared to OFSE and hence avoidance of an accidental escape from its movement toward optimal solution. Whereas OFSE provides larger randomness and may move away from optimal solution, this effect can be observed in the later stages of optimization especially in plots for wire-length and power.

5.2. Comparison with Fast Fuzzy Allocation Scheme

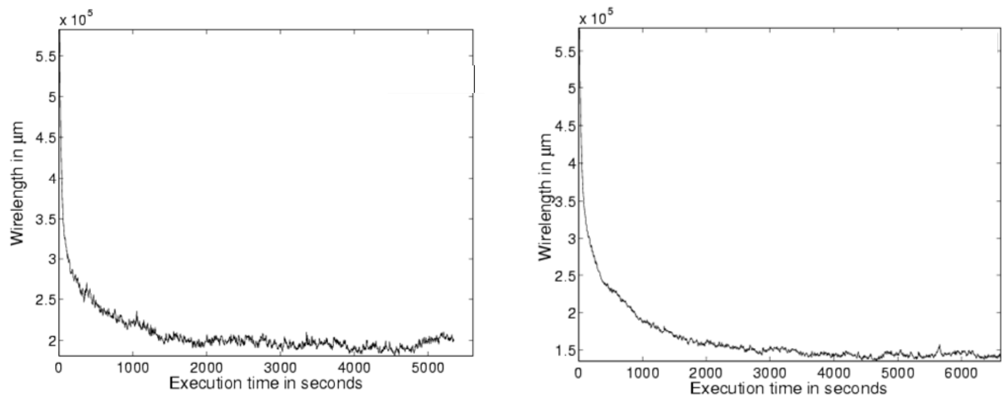
Fast Fuzzy Force Directed Simulated Evolution (FFSE) and OFSE, were applied on 12 ISCAS benchmark circuits [21]. Execution is aborted when no improvement is observed in the last 500 iterations (maximum of 5000 iterations) for OFSE, whereas the algorithm is run for a fixed 5000 iterations for FFSE. The 0.25 micron CMOS digital low power standard cell library for MOSIS is used [22].



(a) (b)

(c) (d)

Fig. 7. Plots (a) and (b) show membership values versus execution time for OFSE and AFSE respectively. Plots (c) and (d) show cumulative number of solutions visited in a specific membership range versus execution time for OFSE and AFSE.



(a) (b)

Fig. 8. Wire-length versus execution time: (a) OFSE and (b) AFSE.

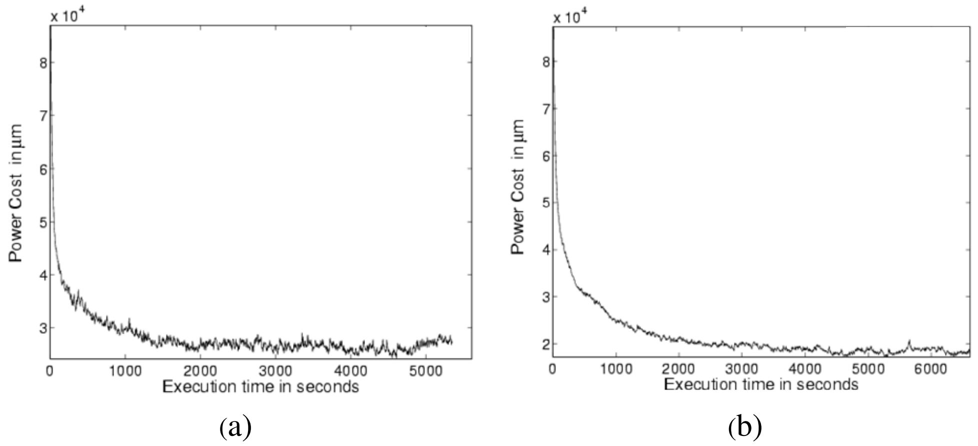


Fig. 9. Power-cost versus execution time: (a) OFSE and (b) AFSE.

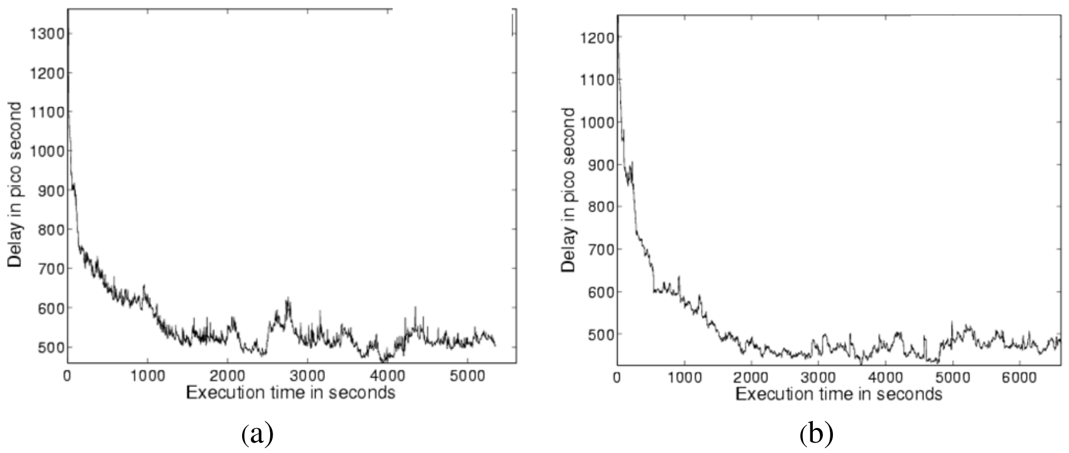


Fig. 10. Delay versus execution time: (a) OFSE and (b) AFSE.

Circuit		OFSE				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	x	x	x	x	655370	114231	908	440
S13207	8651	x	x	x	x	1339837	144189	1604	885
S15850	10383	x	x	x	x	1477662	115049	2006	1202

Table 2. Layout found by OFSE, and FFSE. "L", "P" and "D" represent the wire-length, power, and delay costs and "T" is execution time (sec). Last 3 circuits were not tested for OFSE because of large runtime requirements.

Table 2 compares the quality of final solution generated by OFSE and FFSE. The circuits are listed in order of their size (136- 10383 modules). From the results, it is clear that FFSE has outperformed OFSE for all circuits in terms of execution time. For larger circuits (S3330 & S5378), FFSE is better than OFSE in terms of quality of final solution. In some cases FFSE has not more than 10% degradation in terms of quality of solution but with a significant improvement in run-time.

It can be observed that the algorithm converges very fast. This behavior can be observed in Figure 11, where convergence is achieved after approximately 400 seconds (6.6 minutes), and the remaining time is spent in fine tuning the solution quality.

6. CONCLUSION

A fast fuzzy force-directed simulated evolution algorithm for multiobjective VLSI standard cell placement was proposed and presented in this paper. An improvement in the execution time from $O(n^2)$, in previous SE based approaches, to $O(n)$ is achieved by using force-directed allocation methodology during the allocation stage. Fuzzy logic is applied to handle the multi-objective nature of the problem. Fuzzy logic is also employed at evaluation and allocation stages and for the selection of best solution from the set of generated solutions as well. A set of new fuzzy functions are defined and employed to eliminate the problems of extensive experimentation, tuning and re-runs as was the case when OWA operators were used. The proposed scheme is compared with OFSE. It is observed that FFSE perform much better than OFSE in terms of execution time, with no significant degradation in terms of quality of solution. FFSE can be used for large circuits whereas OFSE cannot be used for circuits with more than 2000-3000 cells. From experimentation and results it was also observed that FFSE totally avoids early random walk, which is a problem in other non-deterministic heuristics such as Simulated Annealing. Comparatively very low amounts of memory is required for SE than other iterative heuristics such as Genetic algorithm since SE retains only one solution at a single instance of time in the memory.

Acknowledgment: The authors and the research team acknowledge King Fahd University of Petroleum & Minerals for its support under research project COE/Cell.Place/263.

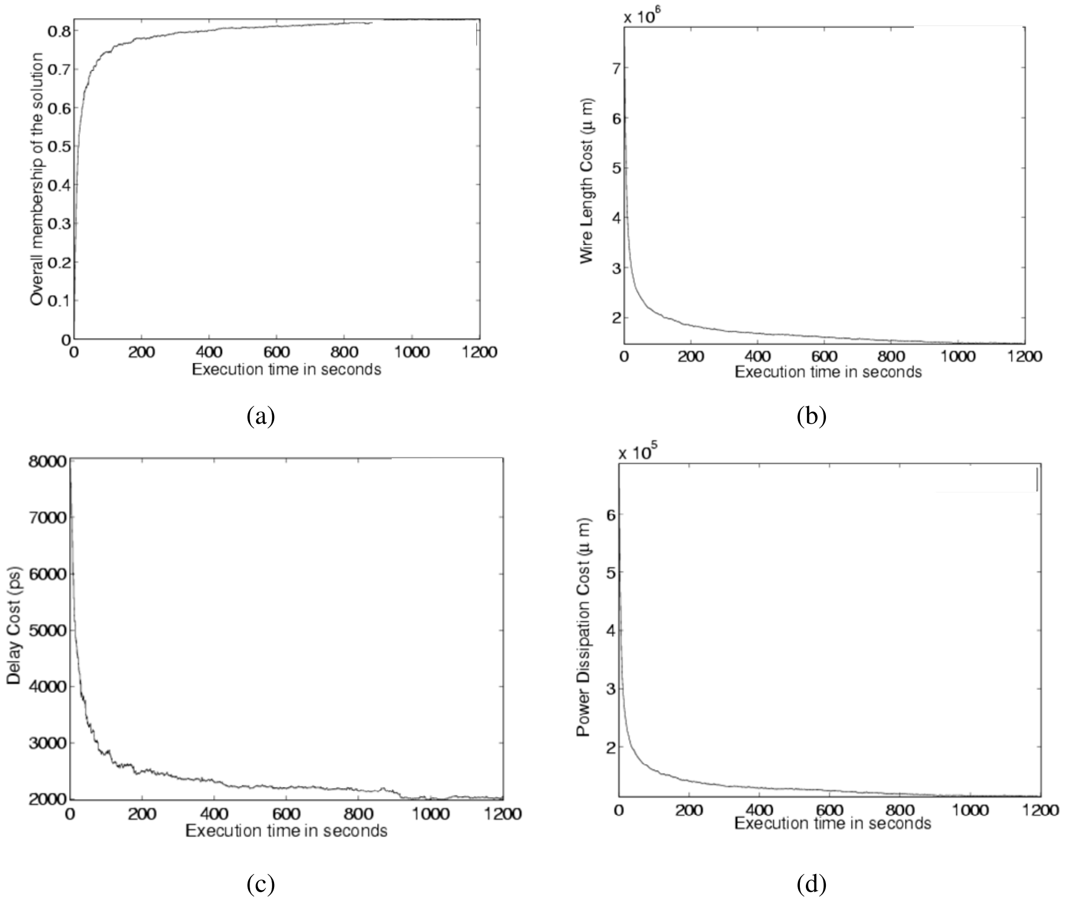


Fig. 11. (a) Overall Membership (b) Wire-length cost (c), Circuit delay, and (d) Power cost versus execution time in seconds for circuit S15850 using FFSE.

7. REFERENCES

[1] Sadiq M. Sait and Habib Youssef. VLSI Physical Design Automation: Theory and Practice. *McGraw-Hill Book Company, Europe (also copublished by IEEE Press, USA)*, 1995.

[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 Midwest Symposium on Circuits and Systems*, 3:1337–1340, August 1996.

[4] F. Glover, E. Taillard, and D. de Werra. A user's guide to tabu search. *Annals of Operations Research*, 41:3–28, 1993.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, INC., 1989.

[6] S. Kirkpatrick, Jr. C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):498–516, May 1983.

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

[9] R. M. Kling. *Optimization by Simulated Evolution and its Application to Cell Placement*. PhD thesis, University of Illinois, Urbana, 1990.

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

[11] 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.

[12] Sadiq M. Sait, Habib Youssef, Junaid A. Khan, and Aiman Al-Maleh. Fuzzified Iterative Algorithms for Performance Driven Low Power VLSI Placement. *IEEE, ICCD2001*, September 2001.

[13] Junaid A. Khan, Sadiq M. Sait, and Mahmood R. Minhas. Fuzzy Biasless Simulated Evolution for Multiobjective VLSI Placement. *IEEE, CEC2002, Honolulu*, May 2002.

[14] L. A. Zadeh. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transaction Systems Man. Cybern*, SMC-3(1):28–44, 1973.

[15] L. A. Zadeh. The Concept of Linguistic Variable and its Application to Approximate Reasoning. *Information Science*, 8:199–249, 1975.

[16] R. Yager. Second Order Structures in Multi-criteria Decision Making. *International Journal of Man-Machine Studies*, pages 36:553–570, 1992.

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

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

[19] Sadiq M. Sait, Habib Youssef, and Ali Hussain. Fuzzy Simulated Evolution Algorithm for Multiobjective Optimization of VLSI Placement. *IEEE Congress on Evolutionary Computation*, pages 91–97, July 1999.

[20] 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.

[21] F. Brglez. A D&T Special Report on ACD/SIGDA Design Automation Benchmarks: Catalyst or Anathema? *IEEE Design & Test*, pages 87–91, September 1993.

[22] Tanner Consulting and Engineering Services. *Digital Low Power Standard Cell Library for MOSIS TSMC CMOS 0.25 Process Deep Sub-Micron Technology*. Tener Research, Inc., 1999.