Fuzzy Evolutionary Algorithm for VLSI Placement

Sadiq M. Sait **Habib Youssef** **Junaid A. Khan**

Computer Engineering Department
King Fahd University of Petroleum and Minerals
KFUPM Box # 673, Dhahran-31261, Saudi Arabia
e-mail: {sadiq,youssef,jakhan}@ccse.kfupm.edu.sa
Telephone: : +966-3-860-6665, Fax: : +966-3-860-3059.

Abstract

This paper presents a Fuzzy Simulated Evolution (FSE) algorithm for VLSI standard cell placement. This is a hard multiobjective combinatorial optimization problem with no known exact and efficient algorithm that can guarantee finding a solution of specific or desirable quality. Approximation iterative heuristics such as Simulated Evolution are best suited to perform an intelligent search of the solution space. Due to the imprecise nature of design information at the placement stage the various objectives and constraints are expressed in the fuzzy domain. The search is made to evolve toward a vector of fuzzy goals. The proposed heuristic is compared with genetic algorithm. FSE was able to achieve better solutions than GA in a fraction of the time.

1 INTRODUCTION

In VLSI design, the placement problem consists of assigning modules to locations on the silicon surface under numerous design constraints while trading-off several objectives. The number of modules can be in range of thousands. Even in its simplest form, placement is a generalization of the quadratic assignment problem [7].

Formally this problem can be stated as follows: Given a set of modules $M = \{m_1, m_2, \cdots, m_n\}$, 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 problem is to assign each module $m_i \in M$ to a unique location such that a given cost function is optimized and constraints are satisfied. Objectives addressed in this work are the minimization of wire-length, power dissipation, and circuit delay. Layout area is considered as a constraint. These are estimated as follows.

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

$$\text{Cost}_{wire} = \sum_{j \in M} l_j \tag{1}$$

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

Estimation of Power: In CMOS circuits, over 90% power dissipation is due to the switching activity [2, 1], expressed as:

$$P_t \simeq \sum_{i \in M} \frac{1}{2} \cdot C_i \cdot V_{DD}^2 \cdot f \cdot S_i \cdot \beta \tag{2}$$

where P_t denotes the total power, V_{DD} is the supply voltage, S_i is the switching activity at the output node of cell i (module m_i) which indicate the number of gate output transition per clock cycle, f is the clock frequency. The node total capacitance is denoted by C_i, and β is a technology dependent constant. Assuming that clocking frequency and power voltages are fixed, total power dissipation of the circuit is a function of C_i and S_i of the various gates in the logic circuit. The capacitive load C_i of a gate comprises input gate capacitances of the cells driven by cell i and that of interconnects capacitance at the cell output node, as shown in the following equation:

$$C_i = C_i^r + \sum_{j \in M_i} C_j^g \tag{3}$$

where C_j^g is the input capacitance for cell (or gate) j, C_i^r represents the interconnect capacitance at the output node of cell i and M_i is the set of fanout cells of cell i.

In case of standard cell design, cell properties are fixed for a particular library, hence we cannot reduce C_j^g. Further C_i^r are related to the corresponding interconnect wire-lengths l_i, hence cost due to the overall power in VLSI circuits can be termed as:

$$\text{Cost}_{power} = \sum_{i \in M} S_i l_i \tag{4}$$

Estimation of Delay: Let path π consist of nets $\{v_1, v_2, ..., v_k\}$, then its path delay T_π is expressed by the following equation:

$$T_\pi = \sum_{i=1}^{k} (CD_i + ID_i) \tag{5}$$

where CD_i is the switching delay of the cell driving net vi and ID_i is the interconnect delay of net vi. The overall circuit delay is equal to T_{π_c}, where π_c is the longest path in the layout (most critical path).

CD_i is constant and only ID_i depends on placement. Using the RC delay model, this delay is estimated as:

$$ID_i = (LF_i + R_i^r) \times C_i \tag{6}$$

where LF_i is load factor of the driving block (independent of layout), R_i^r is the interconnect resistance of net v_i, and C_i is the load capacitance of cell i given in Equation 3. The cost function due to timing performance can be expressed as:

$$\text{Cost}_{delay} = T_{\pi_c} \tag{7}$$

Layout Width: In our work layout width is considered as a constraint. The upper limit on the layout width is defined in Equation 8:

$$Width_{max} = (1 + a) \times Width_{opt} \tag{8}$$

where $Width_{max}$ is the maximum allowable width of the layout, $Width_{opt}$ is the minimum possible layout width obtained by adding the widths of all cells and dividing it by the number of rows in the layout. The parameter a denotes how wide the layout can be as compared to the optimal one.

During placement most of the cost parameters cannot be precisely determined. For example, the exact amount of wire-length and area of the layout can be known only after the subsequent stage of design (routing). Also, the amount of power dissipated, or the performance of the circuit, depends on the amount of wire-length as well as the operation dynamics of the circuit. For these reasons it is much easier for a designer to describe desirable characteristics of placement solutions in linguistic terms which is the basis of fuzzy logic [11]. In this work the evaluation of the *goodness* values of individual modules in their current locations (a requirement of simulated evolution heuristic) as well as the quality of the overall placement solution are described using fuzzy rules.

2 SIMULATED EVOLUTION

Placement is a hard combinatorial optimization problem with no known exact and efficient optimization algorithms that can find a solution of a given quality. Approximation iterative heuristics such as simulated annealing, genetic algorithms, simulated evolution, tabu search, are robust search methods for this category of problems [8].

Simulated Evolution (SE) is a general iterative heuristic proposed by Ralph Kling [6]. It falls in the category of algorithms which emphasize the behavioral link between parents and offspring, or between reproductive populations, rather than the genetic link [4]. This scheme combines iterative improvement and constructive perturbation and saves itself from getting trapped in local minima by following a stochastic perturbation approach. It iteratively operates a sequence of evaluation, selection and allocation steps on one solution. The selection and allocation steps constitute a compound move from current solution to another feasible solution of the state space. SE proceeds as follows. It starts with a randomly or constructively generated valid initial solution. A solution is seen as a set of movable elements (modules). Each element m_i has an associated goodness measure g_i in the interval [0,1]. The main loop of the algorithm consists of three steps (See Figure 1): **evaluation**, **selection** and **allocation**. These steps are carried out repetitively until some stopping condition is satisfied. In the **evaluation** step, the goodness of each element is estimated. In the **selection** step, a subset of elements are selected and removed from current solution. The lower the goodness of a particular element, the higher is its selection probability. A bias parameter B is used to compensate for inaccuracies of the *goodness* measure. Finally, the **allocation** step tries to assign the selected elements to better locations. Other than these three steps, some input parameters for the algorithm are set in an earlier step known as **initialization**.

```
Algorithm Simulated_Evolution(B, Φ_initial, StoppingCondition)
NOTATION
B= Bias Value.
Φ= Complete solution.
m_i= Module i.
g_i= Goodness of m_i.
ALLOCATE(m_i, Φ_i)=Function to allocate m_i in partial solution Φ_i
Begin
Repeat

  EVALUATION:
        ForEach m_i ∈ Φ evaluate g_i;
        /* Only elements that were affected by moves of previous */
        /* iteration get their goodnesses recalculated*/
  SELECTION:
        ForEach m_i ∈ Φ DO
              begin
                  IF Random > Min(g_i + B, 1)
                  THEN
                      begin
                          S = S ∪ m_i; Remove m_i from Φ
                      end
              end
      Sort the elements of S
  ALLOCATION:
        ForEach m_i ∈ S DO
              begin
                  ALLOCATE(m_i, Φ_i)
              end
Until  Stopping Condition is satisfied
Return Best solution.
End (Simulated_Evolution)
```

Figure 1: Structure of the simulated evolution algorithm.

3 FUZZY SIMULATED EVOLUTION

In order to apply simulated evolution one has to design a suitable goodness measure, a cost function, and an appropriate allocation operator. These three together have the most impact on the behavior of the SE algorithm. Due to the multiobjective nature of the placement problem, the goodness measure, cost function, and the allocation operator should take into consideration all objectives.

Balancing different objectives by weight functions is difficult, or at best controversial. Fuzzy logic is a convenient vehicle for solving this problem. It allows to map values of different criteria into linguistic values, which characterize the level of satisfaction of the designer with the numerical value of the objectives. All these numerical values operate over values from the interval [0,1] defined by the membership functions for each objective. For placement, the designer seeks to find solutions optimized with respect to *wire-length*, *delay*, and *power dissipation*.

3.1 FUZZY GOODNESS EVALUATION

Following the generation of an initial solution, the goodness of each cell in its current location is determined. 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 conveniently expressed by the following fuzzy logic rule.

Rule R1: 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 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.

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*. In our case, the linguistic value used in the consequent part identifies the fuzzy subset of good locations for that particular cell. 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., $\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$, and $\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$. This is known as the $\min - \max$ logic initialy introduced by Zadeh [11]. For example, according to the min-max logic the rule above evaluates to the following:

$$\mu_i^e(x) = \min\left(\mu_{iw}^e(x), \mu_{ip}^e(x), \max(\mu_{inet}^e(x), \mu_{ipath}^e(x))\right) \quad (9)$$

where the superscript e is used here to represent evaluation, x represents the location of cell i, $\mu_i^e(x)$ represents the membership in the fuzzy set of high goodness, $\mu_{iw}^e(x)$ and $\mu_{ip}^e(x)$ represent the memberships in fuzzy subsets of *near optimal wire-length* and *low power* as compared to other cells; $\mu_{inet}^e(x)$ and $\mu_{ipath}^e(x)$ are the memberships in fuzzy sets of *near optimal net delay* as compared to other cells and "$T_{max}(i)$ is *much smaller than* T_{max}".

However, 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. 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 [9]. This operator falls in the category of compensatory fuzzy operators and allows easy adjustment of the degree of "anding" and "oring" embedded in the aggregation. According to [9], "orlike" and "andlike" OWA for two fuzzy sets A and B are implemented as given in Equations 10 and 11 respectively.

$$\mu_{A\bigcup B}(x) = \beta \times \max(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B) \tag{10}$$

$$\mu_{A\bigcap B}(x) = \beta \times \min(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B) \tag{11}$$

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.

With the AND and OR fuzzy operators implemented as OWA operators, rule **R1** evaluates to the expression below:

$$\begin{aligned} g_i = \mu_i^e(x) = \beta^e \times min(\mu_{iw}^e(x), \mu_{ip}^e(x), \mu_{id}^e(x)) \\ +(1-\beta^e) \times \frac{1}{3} \sum_{j=w,p,d} \mu_{ij}^e(x) \end{aligned} \tag{12}$$

where

$$\begin{aligned} \mu_{id}^e(x) = \beta_d^e \times max(\mu_{inet}^e(x), \mu_{ipath}^e(x)) \\ + (1-\beta_d^e) \times \frac{1}{2}(\mu_{inet}^e(x) + \mu_{ipath}^e(x)) \end{aligned} \tag{13}$$

g_i is the goodness of cell i. β^e and β_d^e are constants between 0 and 1 to control OWA operators. Whereas $\mu_{id}^e(x)$ represents the membership in fuzzy set of *good timing performance*, it is obtained after applying orlike OWA to $\mu_{inet}^e(x)$ and $\mu_{ipath}^e(x)$.

$\mu_{ipath}^e(x)$ is included in the computation of $\mu_{id}^e(x)$ because if a cell is not on the critical path then it must have high goodness with respect to the delay objective.

If a cell i drives the net v_i, $\{v_1, v_2, ..., v_k\}$ is the set of nets connected to cell i and v_p is the net driven by the predecessor cell of cell i on the longest path traversing cell i, then base values $X_{iw}(x)$, $X_{ip}(x)$, $X_{inet}(x)$ for fuzzy sets near optimal wire-length, power, net delay and $X_{ipath}(x)$ for fuzzy set "$T_{\max}(i)$ much smaller than $T_{\max}$" are computed as given in Equations 14-17,

$$X_{iw}(x) = \frac{\sum_{j=1}^{k} l_j^*}{\sum_{j=1}^{k} l_j} \tag{14}$$

$$X_{ip}(x) = \frac{\sum_{j=1}^{k} S_j \; l_j^*}{\sum_{j=1}^{k} S_j \; l_j} \tag{15}$$

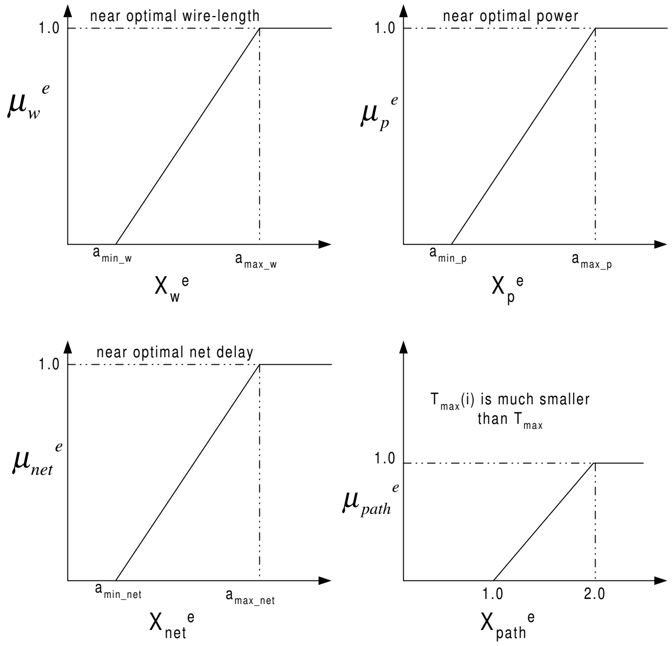

Figure 2: Membership functions used in fuzzy evaluation.

$$X_{inet}(x) = \frac{ID_i^* + ID_{ip}^*}{ID_i + ID_{ip}} \tag{16}$$

$$X_{ipath}^e(x) = \frac{T_{max}}{T_{max}(i)} \tag{17}$$

where l_j^* is the optimal wire-length of net v_j, computed by placing all the cells connected to v_j next to each other on the layout surface and then estimating the wire-length; the product $S_j \times l_j$ is related to the switching power dissipated in net v_j; ID_i^* is the optimal interconnect delay of net v_i, ID_{ip} and ID_{ip}^* are the actual and optimal interconnect delays of net driven by the predecessor cell of cell i on the current longest path traversing cell i. Membership functions of these base values are shown in Figure 2.

The values of a_{min} and a_{max} depend on the statistical nature of the base values. A typical frequency of occurance plot is shown in Figure 3, where we have plotted $X_{net}^e(x)$ and $X_w^e(x)$ versus the number of cells having these base values. It is clear from this figure that these plots have nearly bell-shaped behavior. Therefore we can say that around 95% cells have base values in the range $[\bar{X}_i - 2\sigma_i, \bar{X}_i + 2\sigma_i]$, where $\bar{X}_i$ is the mean value of $X_i(x)$ and σ_i is the standard deviation of $X_i(x)$ for $i = w, p, net$. The values of a_{min} and a_{max} are therefore computed as:

$$a_{min\_i} = \bar{X}_i - 2\sigma_i \;\; and \;\; a_{max\_i} = \bar{X}_i + 2\sigma_i \tag{18}$$

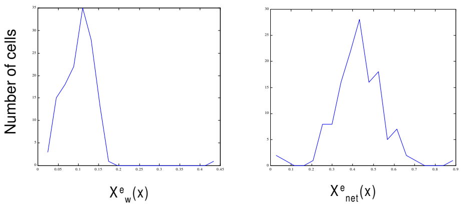

Figure 3: Base values vs frequency of occurance plot.

The values of a_{min} and a_{max} are computed in the beginning and then recomputed again when the size of selection set is around 90 percent of the initial value.

3.2 SELECTION

In this stage of the algorithm some cells are selected probabilistically depending on their goodness values. Bias concept in selection step, present in the original SE algorithm [5], is the major drawback of the heuristic. It is not easy to select value of bias because it varies from problem to problem. Also in the case of placement it varies from circuit to circuit [10]. To overcome this problem another selection scheme is proposed. According to this scheme a random number is generated in the range $[0, M]$ and compared with g_i. If the generated random number is greater than g_i then the cell is selected for allocation. The value of M is calculated as follows:

$$M = \bar{G} + 2\sigma_g \tag{19}$$

where $\bar{G}$ and σ_g are the average goodness and standard deviation of goodness values of cells in current iteration. Value of M is calculated in the beginning and updated only once, when the size of selection set is 90% of its initial size.

3.3 ALLOCATION

In allocation stage the selected cells are to be placed in the best available locations. In our proposed scheme we have considered selected cells as movable modules and remaining cells as fixed modules. Selected cells are sorted in descending order of their goodnesses with respect to their partial connectivity with unselected cells. Ties are broken with respect to their goodness values. One cell from the sorted list is selected at a time and its location is swapped with other movable cells in the current solution. The swap that results in the maximum gain is accepted and the cell is removed from the selection set.

The goodness of the new location is characterized by the following fuzzy rule:

Rule R2: IF a swap results in *reduced overall wire-length* AND *reduced overall power* AND *reduced delay* AND *within acceptable layout width* **THEN** it gives good location.

The above rule is interpreted as follows.

$$\mu^a_{i\_pwd}(l) = \beta^a \times min(\mu^a_{ip}(l),\ \mu^a_{iw}(l),\ \mu^a_{id}(l)) + (1-\beta^a) \times \frac{1}{3} \sum_{j=p,w,d} \mu^a_{ij}(l) \tag{20}$$

$$\mu^a_i(l) = min(\ \mu^a_{i\_width}(l),\ \mu^a_{i\_pwd}(l)\ ) \tag{21}$$

the superscript a is used here to represent allocation. $\mu^a_i(l)$ is the membership of cell i at location l in the fuzzy set of good location. $\mu^a_{i\_pwd}(l)$ is the membership in the fuzzy set of "reduced wire-length and reduced power and reduced delay". $\mu^a_{iw}(l)$, $\mu^a_{ip}(l)$, $\mu^a_{id}(l)$, and $\mu^a_{iwidth}(l)$ are the membership in the fuzzy sets of reduced wire-length, reduced power, reduced delay and within acceptable width respectively.

Notice that the third AND operator in the above fuzzy rule is implemented as a pure min because the width constraint has to be always satisfied.

If a cell i swaps its location with cell j then the base values are computed as shown in Equations 22-25:

$$X^a_{iw}(l) = \frac{(\sum_{m=1}^{k_i} l_{im} + \sum_{m=1}^{k_j} l_{jm})_n}{(\sum_{m=1}^{k_i} l_{im} + \sum_{m=1}^{k_j} l_{jm})_{n-1}} \tag{22}$$

$$X^a_{ip}(l) = \frac{(\sum_{m=1}^{k_i} S_{im} l_{im} + \sum_{m=1}^{k_j} S_{jm} l_{jm})_n}{(\sum_{m=1}^{k_i} S_{im} l_{im} + \sum_{m=1}^{k_j} S_{jm} l_{jm})_{n-1}} \tag{23}$$

$$X^a_{id}(l) = \frac{(ID_i + ID_{ip} + ID_j + ID_{jp})_n}{(ID_i + ID_{ip} + ID_j + ID_{jp})_{n-1}} \tag{24}$$

$$X^a_{i\_width}(l) = \frac{Width_n}{Width_{opt}} \tag{25}$$

where, subscript n and $n-1$ show the iteration numbers, $\{v_{i1}, v_{i2}, ..., v_{ik_i}\}$ is the set of nets connected to cell i, $Width_n$ is the actual width at n^{th} iteration.

Membership functions for these base values are shown in Figure 4. The values of a_w, a_p, a_d and a_{width} depend upon priority on the optimization level of the respective objective. Typical values for a_w, a_p and a_d are in the range $[0.75, 0.95]$, whereas a_{width} is in the range $[0.2, 0.5]$. In our case we have set $a_w = 0.75$, $a_p = 0.75$, $a_d = 0.85$ and $a_{width} = 0.25$.

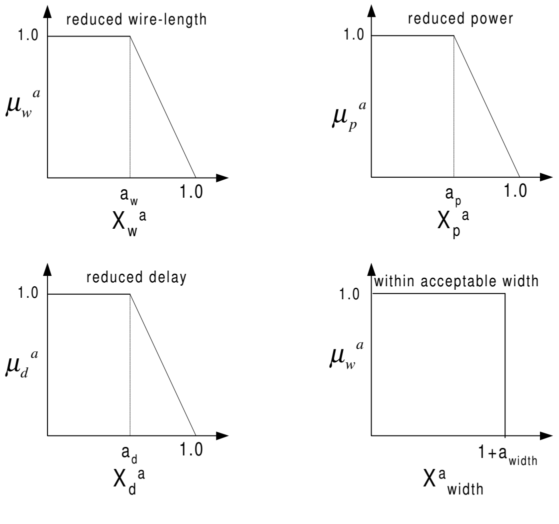

Figure 4: Membership functions used in allocation.

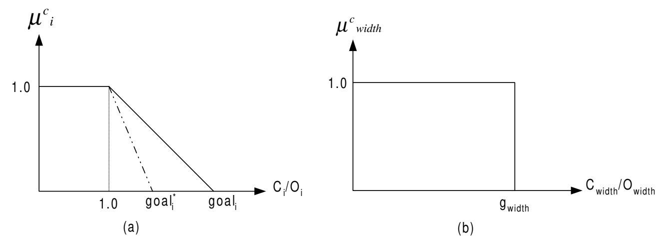

Figure 5: Membership functions within acceptable range.

3.4 FUZZY COST MEASURE

In order to select the best solution found so far, it is required to develop some cost measure. In case of multi-objective placement, the best placement is one which results in lowest cost in terms of all objectives. However, such a solution most likely does not exist. Some techniques to cope with this problem are mentioned in [12]. All these techniques introduce some tradeoff between different objectives. In this work, a goal directed search approach is adopted, where the best placement is one that satisfies as much as possible a user specified vector of fuzzy goals [3].

In order to combine three parameters and one constraint, following fuzzy rule is suggested.

Rule R3: 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.

The above fuzzy rule translates to the following:

$$\mu^c_{pdl}(x) = \beta^c \times min(\mu^c_p(x), \mu^c_d(x), \mu^c_l(x)) + (1-\beta^c) \times \frac{1}{3} \sum_{j=p,d,l} \mu^c_j(x) \quad (26)$$

$$\mu^c(x) = min(\mu^c_{pdl}(x), \mu^c_{width}(x)) \quad (27)$$

where $\mu^c(x)$ is the membership of solution x in fuzzy set of acceptable solutions, $\mu^c_{pdl}(x)$ is the membership in fuzzy set of "acceptable power AND acceptable delay AND acceptable wire-length", whereas $\mu^c_j(x)$ 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. β^c is the constant in the range $[0, 1]$, in our case we chose $\beta^c = 0.7$, the superscript c represents "cost". The solution that results in maximum value of $\mu^c(x)$ is reported as the best solution found by the search heuristic.

The membership functions for fuzzy sets *within acceptable power, delay and wire-length* are shown in Figure 5(a), whereas the constraint *within acceptable layout width* is given as a crisp set as shown in Figure 5(b).

Since layout width is a constraint, hence its membership value is either 1 or 0 depending upon $goal_{width}$ (in our case $goal_{width} = 1.25$). However, for other objectives by increasing and decreasing the value of $goal_i$ we can vary its preference in overall membership function. The lower bounds ($O_i s$ for $i \in \{l, p, d, width\}$) are computed as follows: $O_l = \sum_{i=1}^n l^*_i$, $O_p = \sum_{i=1}^n S_i l^*_i, \forall v_i \in \{v_1, v_2, ..., v_n\}$; $O_d = \sum_{j=1}^k CD_j + ID^*_j \ \ \forall v_j \ in \ path \ \pi_c$; and $O_{width} = Width_{opt}$; where k is the number of nets in π_c.

4 GA BASED OPTIMIZATION

For the comparison purpose we have also implemented **genetic algorithm (GA)** [8]. Membership value in the fuzzy set of acceptable solution given in Equation 27 is used as the **fitness** measure of a chromosome (solution). In parent selection step for crossover **roulette wheel** selection scheme [8], is used. **Partially mapped crossover (PMX)** is used to generate new offsprings. For the selection of next generation **Extended Elitism Random Selection** scheme is used, where half of the chromosomes in the next population are the best among offspring and current population and half are selected randomly. A variable **mutation** is used in the range $[0.03, 0.05]$ that depends upon the standard deviation of fitness in the current population. Stopping criteria is the maximum number of generations.

5 EXPERIMENTS AND RESULTS

We have applied GA and FSE on eleven different ISCAS-85 and ISCAS-89 benchmark circuits. In case of FSE algorithm, execution is aborted when no improvement is observed in the best solution found so far in 500 consecutive iterations. In case of GA stopping criteria is 10,000 generations.

Table 1 compares the quality of final solution generated by FSE and GA. The circuits are listed in order of their size (122-1753 modules). From the results, it is clear that GA performs better than FSE for smaller circuits, but for circuits with large number of cells FSE outperforms GA. In all circuits it is observed that GA takes considerably large amount of execution time as compared to FSE.

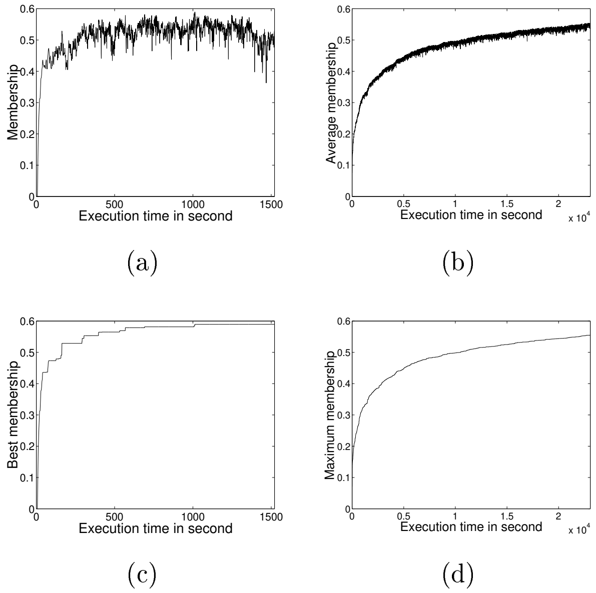

Figure 6: Comparison of FSE and GA. (a) and (c) represent current and best fitness (membership) of the solution obtained by FSE; (b) and (d) represent average and best solution obtained by GA, plotted versus execution time in seconds.

To compare improvement in the quality of solution versus time, we have plotted current and best membership values of the solution obtained by FSE versus actual execution time in Figure 6-(a) and (c), for comparison the average and best fitness (membership) values in a current population obtained by GA versus execution time are plotted in Figure 6-(b) and (d). These plots are for test case S1196. It can be observed that quality of solution improves rapidly in FSE based search as compared to GA. Due to lack of space plots for other circuits are not included; however both heuristics exhibited similar behavior on all circuits.

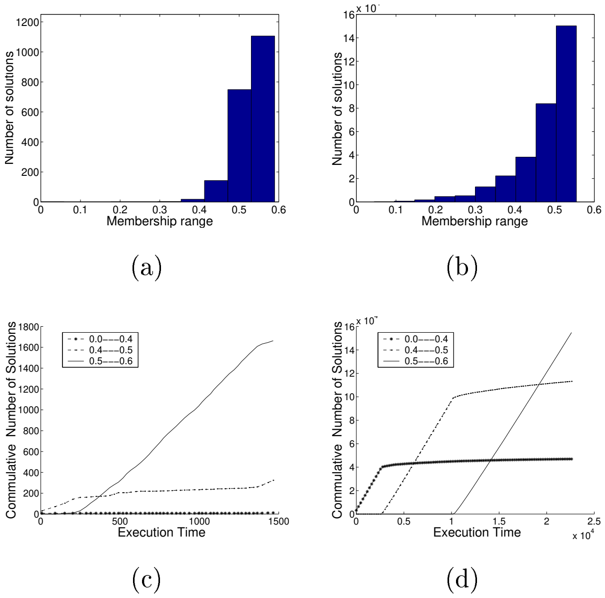

Figure 7: Comparison of FSE and GA based on search efforts in particular membership ranges; (a) and (b) show number of solutions visited in particular membership ranges for FSE and GA respectively; (c) and (d) show cumulative number of solutions visited in a specific membership range versus execution time in seconds for FSE and GA respectively.

Figures 7(a) and (b) show the quality of solution subspace searched by FSE and GA. It is evident from the figure that both heuristics concentrated in high fitness subspaces which indicates that they were properly engineered to solve this particular problem. The figure also shows that FSE was able to evolve much faster toward a better solution subspace (after few hundred seconds). On the other hand GA required generations in the order of thousands, where each generation consisted of 32 solutions.

Figures 7(c) and (d) track with time the total number of solutions found by FSE and GA for various membership ranges. These are very informative plots as they show, that as time progressed, the solutions found by each heuristic were getting better. Note however that FSE exhibited much faster evolutionary rate than GA. For example, after about 400 seconds, almost all new solutions discovered by FSE have a membership in the interval 0.5-0.6 in the fuzzy subset of good solutions with respect to all objectives, and almost none were found with lower membership values (see Figure 7(c)). In contrast, for GA, it is only after 10,000 seconds that the first solution with membership in the interval 0.5-0.6 was found (see Figure 7(d)). This behavior was observed for all test cases.

Circuit	GA					FSE				
	L (μm)	P (μm)	D (ps)	W (μm)	T (s)	L (μm)	P (μm)	D (ps)	W (μm)	T(s)
S2081	2426	388	113	142	2341	2693	462	112	152	43
S298	4062	838	130	171	2922	4989	1013	133	181	104
S386	6824	1665	193	181	3945	7088	1640	197	186	152
S832	21015	4787	395	232	7206	24705	5827	390	258	1643
S641	18320	4365	736	254	21982	13906	3321	702	296	618
S953	32031	5156	230	262	11221	32340	5242	245	262	1278
S1238	52679	15473	410	279	16208	39629	12377	371	310	1168
S1196	51804	15259	370	292	23070	42426	12745	364	325	1521
S1494	71021	17497	803	336	26032	56961	14071	719	360	3378
S1488	69792	17346	784	334	21434	57091	13887	710	358	3529
C3540	310996	109850	924	427	43232	164897	58055	734	507	18318

Table 1: Layout found by FSE and GA, "L", "P" "D" and "W" represent the wire-length, power, delay, and width costs and "T" represents execution time in seconds

6 CONCLUSION

In this paper, we have proposed an evolutionary algorithm for low power high performance VLSI standard cell placement. We have used FSE as search heuristic and GA for comparison. Fuzzy logic is used to overcome the multi-objective nature of the problem. In FSE, fuzzy logic was implemented at evaluation and allocation stages and to choose the best solution from the set of solutions generated by FSE. In GA fuzzy logic is used in the fitness evaluation.

It is observed that FSE performs much better than GA in terms of execution time, it also performs better than GA in terms of final solution in bigger circuits. Also the quality of solution improved more rapidly in FSE based search as compared to GA.

Acknowledgments

Authors acknowledge King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, for all support. This work is carried out under university funded project number COE/ITERATE/221.

References

[1] A. Chandrakasan, T. Sheng, and R. W. Brodersen. Low Power CMOS Digital Design. *Journal of Solid State Circuits*, 4(27):473–484, April 1992.

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

[3] Henrik Esbensen and Ernest S. Kuh. Design space exploration using the genetic algorithm. *ISCAS '96. IEEE International Symposium on Circuits and Systems*, pages 500–503, 1996.

[4] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, Jan 1994.

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

[6] Ralph Michael Kling. *Optimization by Simulated Evolution and its Application to cell placement.* Ph.D. Thesis, University of Illinois, Urbana, 1990.

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

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

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

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

[11] 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.

[12] Eckart Zitzler. Evolutionary optimization for multiobjective optimization: Methods and applications. *DTS thesis, Swiss Federal Institute of Technology Zurich*, November 1999.