Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Engineering Applications of Artificial Intelligence 16 (2003) 407–423
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

Simulated evolution for timing and low power VLSI standard cell placement

Sadiq M. Sait*, Junaid A. Khan

Computer Engineering Department, King Fahd University of Petroleum & Minerals, KFUPM Box 673, Dhahran-31261, Saudi Arabia

Abstract

This paper presents a Fuzzy Simulated Evolution algorithm for VLSI standard cell placement with the objective of minimizing power, delay and area. For this hard multiobjective combinatorial optimization problem, no known exact and efficient algorithms exist that 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. Variants of the algorithm which include adaptive bias and biasless simulated evolution are proposed and experimental results are presented. Comparison with genetic algorithm is discussed.

1. Introduction

Until the beginning of this decade, two main objectives of VLSI physical design were the minimization of interconnect wirelength and the improvement of timing performance (Sait and Youssef, 1995). Since the mid 1990s, there has been a rapid increase in the use of portable devices such as laptop computers, mobile phones, PDAs, and many others. Such kind of devices rely on a battery for their power needs. The battery power is limited by the factors of its size and weight, both of which are desired to be as small as possible. Also, the battery technology is expected to improve by only 30% in the near future.

Another compelling reason for the desire of low-power consumption is the increasing density of VLSI circuits. With the rapid advancement in technology, VLSI circuits are reducing in size resulting in higher transistor density on a chip. The present technology allows integration of millions of transistors on a single chip and the still advancing technology is allowing further high integration. The excessive power consumption of the circuit results in heating and thus becoming a hindrance towards high integration and hence the feasible packaging of circuits (Narayanan et al., 1999).

Also, present day circuits operate at much higher clock frequencies than before. Therefore, the power dissipation which is a function of the clock frequency, is getting significantly prominent. This phenomenon is becoming an obstacle in further increase of clock frequency. Due to these reasons, there is an emerging need for minimizing the power requirement of VLSI and other electronic digital circuits.

VLSI design is a complex process and is therefore broken down into a number of intermediate steps (Sait and Youssef, 1995). The design cycle starts from an abstract idea, and then each intermediate step continues refining the design. The process ends with the fabrication of a new chip as illustrated in Fig. 1.

Placement is a phase in physical design responsible for the arrangement of cells on a layout surface while optimizing certain objectives. The placement problem can be stated as follows. Given a collection of cells or modules with pins (inputs, outputs, power and ground pins) on the boundaries, the dimensions of these cells, and a collection of nets (which are sets of pins that are to be wired together), the process of *placement* consists of finding suitable physical locations for each cell on the entire layout. By suitable we mean those locations that minimize given objective functions, subject to certain constraints imposed by the designer, the implementation process, or layout strategy and the design style (Sait and Youssef, 1995). The cells may be standard-cells, macro blocks, etc.

*Corresponding author. Tel.: +996-3-860-3059; fax: 966-3-860-2217.
E-mail address: sadiq@ccse.kfupm.edu.sa (S.M. Sait).

0952-1976/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2003.08.004

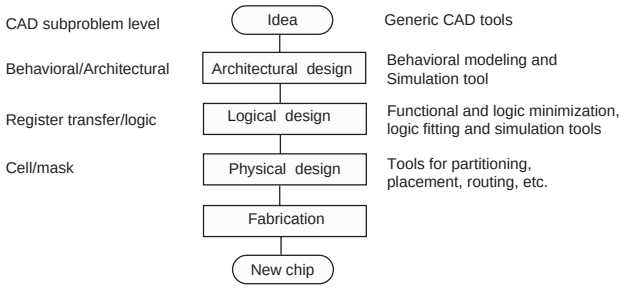

Fig. 1. Various steps in VLSI design process.

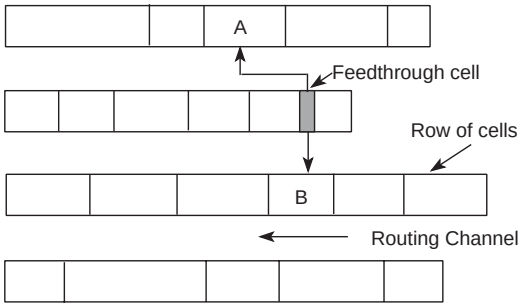

Fig. 2. Layout of a standard-cell placement.

In standard-cell placement, all cells in the cell library are of the same height with variable widths (depending upon the complexity of the cell). Each cell has its ports at the top and at the bottom. Fig. 2 shows the diagram of a standard-cell layout.

The placement of cells in order to optimize even a single objective, e.g., interconnect wirelength is an NP-complete problem (Sait and Youssef, 1995). The simplest case of the problem, namely one-dimensional placement, is hard to solve due to the large size of search space. For n given cells, there are as many as $n!/2$ possible linear arrangements. Therefore, it is not possible to check all the arrangements in polynomial time. This fact favors the application of non-deterministic iterative algorithms such as Simulated Annealing, Genetic Algorithm, Tabu Search etc., over deterministic ones.

The optimization for power consumption can be performed at various levels of VLSI design including behavioral level, architectural level, logic level, and physical level. This work addresses the multiobjective problem of simultaneously optimizing power consumption, timing performance, and wirelength of VLSI circuits at the placement step in physical level. Standard-cell layout is considered in which all the cells in a circuit have the same height, but varying widths (Sait and Youssef, 1995). Simulated Evolution is engineered for solving this optimization problem.

The rest of this paper is organized as follows. Section 2 presents related work, and a brief review of literature on VLSI cell placement for low performance and power, along with an overview of multiobjective optimization and fuzzy logic.

In Section 3, the multiobjective placement problem is formulated. The cost functions for objectives, i.e., interconnect wirelength, timing performance, power consumption as well as for width constraint are defined. Also, the overall cost function used in multiobjective optimization is designed using fuzzy operators. Section 4 discusses the implementation details of the proposed Simulated Evolution algorithm. Experimental results for the application of the proposed techniques to ISCAS-85/89 benchmarks are discussed and compared in Section 5. Some conclusions and possible future directions of work are given in Section 6.

2. Related work

A typical VLSI design process is divided into several levels of design abstraction as shown in Fig. 1. Efforts have been made in reducing power dissipation and delay at nearly every level of abstraction.

In this section, a brief survey of some important studies in low power and performance-driven VLSI design at different levels of abstraction are presented. Since we are dealing with multiobjective VLSI cell placement problem, literature survey on solving multiobjective optimization problem is also included.

2.1. Low-power VLSI design

In CMOS VLSI circuits, around 90% power dissipation is due to the switching activity, therefore most of the efforts were focused on reducing this switching activity at various levels of abstraction.

Architectural level: At this level, by reducing the number of control paths, power consumption can be reduced. This technique was used in Devadas and Malik (1995), where several transformations were used to reduce the amount of resources. These transformations can be guided with power estimation techniques as discussed in Chatterjee and Roy (1994). If number of modules were available with known power/delay estimations then appropriate selection of module can lead to lower power with the same throughput (Devadas and Malik, 1995). In Bright and Arslan (1997), a novel technique was used to optimize the circuit for low power with the increased timing performance. High level transformations were used to increase operating speed assuming 5 V supply, and then lowering the supply voltage which induces delay to compensate the increase in the speed due to the transformation. In Ramprasad et al. (1999), a source-coding framework was presented to reduce the switching activities in address and data busses. In Panda and Dutt (1999), a memory mapping

scheme is used to reduce the switching activity in the address bus.

Logic level: At logic level, power dissipation can be optimized either in the combinational part or sequential part of the circuit. Combinational power dissipation depends on the probability of the gate evaluating to a 1 or 0. This probability can be changed by utilizing the don't care sets. The methods of don't care optimization to reduce switching activity was presented in Shen et al. (1992).

Spurious transitions account for between 10% and 40% of the switching activity power in typical combinational logic circuits (Ghosh et al., 1992). In order to reduce spurious switching activity, the delays of paths that converge at each gate in the circuit should be made roughly equal. This can be accomplished by selectively adding unit-delay buffers to the inputs of a gate in a circuit. Methods to reduce spurious switching activity were proposed in Lemonds and Shetti (1994).

In technology-independent optimization, factorization of logic expression can be done to reduce transistor count. Common sub-expressions can be found across multiple functions and can be reused. Kernel extraction algorithm were used to perform multi-level logic optimization for area (Brayton et al., 1987) and switching activity power dissipation (Roy and Prasad, 1992).

In sequential circuits, optimization can be done at two levels of abstraction (a) State transition graph level and (b) Logic-gate and flip-flop level. State encoding can be done to reduce switching activity. If the Hamming distance between two states is large then two states should be given uni-distance code in order to minimize switching activity thereby minimizing the power dissipation (Devadas and Malik, 1995).

An encoded scheme to reduce switching activity in data path can be used to reduce power. In order to minimize the switching activity in the bus, the value to be transferred is sent as true value or complimented value. The decision which value is better to send depends upon the previous value in the bus. For example if the previous value is "0000" and new value is "1110" then complemented value "0001" will be sent. An additional control line is used to indicate that true value is sent or complimented value is sent (Stan and Burleson, 1994). Other encoding schemes are one-hot encoding (Chren, 1995) and Gray encoding (Hakenes and Manoli, 1999).

In sequential circuits, the values stored in the particular registers need not be updated in every clock cycle, therefore gated-clock can be used to minimize the switching activity at the input of the flip-flops (Chandrakasan, 1994).

Low-power physical design: At physical design level, power dissipation cannot be minimized by reducing the switching probabilities of the gates in the circuit. However, at physical design level power can be minimized by reducing the circuit capacitances. Physical design consists of partitioning, floorplanning, placement on the layout floor, routing of the placed circuit, etc., (Sait and Youssef, 1995). Efforts to reduce power at physical design level have also been reported in the literature. In Vaishnav and Pedram (1999), power optimization is done along with performance improvement at partitioning stage. In Prabhakaran et al. (1999), simulated annealing was used in the floorplan. Switching activity of the interconnect between two modules was used to decide which modules have to be adjacent in the floorplan. In Chao and Wong (1995), a floorplan for low-power design was presented where the objective was to reduce total power consumption and area during the selection and placement of the circuit modules. In Roy (1999), low-power FPGA place and route problems were discussed under timing constraint. Power dissipation was reduced by reducing the routing capacitances taking into account capacitance due to unprogrammed anti-fuses.

A modification of VPNR (a standard cell place and route tool) was discussed in Holt and Tyagi (1995), where energy minimization objective was included and the result was EPNR (Energy based Place aNd Routing). During the placement phase in EPNR, cells were placed such that the nets with higher switching probability have lower wire length, even if it results in increasing the length of low switching nets. During routing phase, high switching nets are given low chance of distorting as compared to low switching nets. When EPNR was tested on MCNC benchmark circuits, it provided 18.5% energy saving at the cost of 6.2% increase in area. In Holt and Tyagi (1996), the same authors gave the idea of GEEP which is a low-power Genetic Algorithm layout system. It was demonstrated that GEEP reduces interconnect capacitance by an average of 20% over recursive min-cut area optimizing placement.

PCUBE, a performance-driven placement algorithm for low power was introduced in Vaishnav and Pedrarn (1993). The problem was formulated as a constrained programming problem and is solved into two phases (a) global optimization and (b) slot assignment. In both phases total weighted wire-lengths, with switching probability of nets as weights was taken as the objective function. Constraint on total path delay was also used. Average reduction in power consumption of 7% was reported with 8% increase in wire-length and 2% increase in circuit delay.

Other methods of power optimization include lowering the power supply voltage. This lowers the power consumption but unfortunately the delay increases with the effect being more drastic at voltages close to threshold voltage (Nagendra et al., 1996). Proper reordering and resizing of the transistor can be done to reduce short-circuit power dissipation during the transition time (Nagendra et al., 1996). Multiple supply

voltages can also be used to reduce power in the circuit. For this purpose clustered voltage scaling (CVS) scheme was proposed in Usami et al. (1996), where the power supply voltages of non-timing critical parts of the circuits are reduced whereas the supply voltage of timing critical parts are kept. In Igarashi et al. (1997), a novel multiple power supply scheme, "Row by Row optimized Power Supply (RRPS) scheme", which provides different voltage supply to each cell row in the layout was proposed. In Sundararajan and Parhi (1999), buffer re-distribution is done to resize the gates in order to reduce power dissipation.

2.2. Performance-driven VLSI design

The performance (speed) of a circuit may be characterized by the longest combinational delay from an input pin to an output pin. If the path delays are to be kept below a maximum value then the wiring delays must be kept in check. Since placement affects the wiring requirements of a layout, the objective of the placement problem can be altered to satisfy the path timing requirements. With the advances in integration technology, sizes of transistors have been decreasing and their switching speeds increasing. This trend has been so marked in recent years that wiring delays are becoming more noticeable when compared to switching delays. Not only in technologies such as Emitter Coupled Logic, (ECL) but also in CMOS, this effect is pronounced.

The need for performance-driven placement arose as early as 1984 (Dunlop et al., 1984). Since then, a large number of approaches to timing-driven placement have been reported. General approaches that have been suggested to correct long path timing problems include making changes to the logic (Darringer et al., 1984), transistor sizing to speed up some of the circuit elements on the slow paths, etc. It is also possible to make a circuit faster without making any changes in its logic design. This can be achieved at the level of physical design by imposing timing constraints on the interconnects and paths of the design. Attempts that use the strategy to make the physical design sensitive to the timing have been reported (Dunlop et al., 1984; Youssef et al., 1992).

The reported approaches that use the above strategy fall into three general classes. The first class transforms the timing constraints on the critical paths (or sometimes all the paths) into *weights on nets*. These weights are used to categorize the nets and influence the placement procedure (Dunlop et al., 1984). The second class transforms the path timing constraints into *timing bounds on the nets*. The net timing bounds are converted into length bounds and supplied to the placement program which tries to satisfy them (Youssef et al., 1992). The third approach consists of supplying to the placement procedure a set of the critical paths, together with their timing requirements. These paths are monitored during the placement process (Sutanthavibul et al., 1993). This third approach is adopted in this work.

2.3. Multiobjective optimization

Many real-world optimization problems involve two types of difficulties: (a) multiple, conflicting objectives, and (b) a highly complex search space. On the one hand, instead of a single optimal solution, competing goals give rise to a set of compromise solutions, generally denoted as Pareto-Optimal. In the absence of preference information, none of the corresponding trade-offs can be said to be better than the others. On the other hand, the search space can be too large and too complex to be solved by exact methods. Thus, efficient optimization strategies are required that are able to deal with both difficulties. VLSI standard cell placement problem is not far from these real-world problems as it also involves *multiple, possibly conflicting objectives and a highly complex search space*.

A general *multiobjective optimization problem* (MOP) includes a set of n parameters (decision variables), a set of k objectives, and a set of m constraints. Objective functions and constraints are functions of the decision variables. The optimization goal is defined as

$$\text{minimize } y = f(x) = (f_1(x), f_2(x), \ldots, f_k(x)) \tag{1}$$

$$\text{subject to } e(x) = (e_1(x), e_2(x), \ldots, e_m(x)) \leqslant 0, \tag{2}$$

where

$$x = (x_1, x_2, \ldots, x_n) \in X,$$
$$y = (y_1, y_2, \ldots, y_k) \in Y$$

and x is the decision vector, y is the objective vector, X is denoted as the decision space, and Y is called the objective space. The constraints $e(x) \leqslant 0$ determine the set of feasible solutions. The feasible set X_f is defined as the set of decision vectors x that satisfy the constraints $e(x)$ (Zitzler, 1999).

In the placement problem the three objectives, namely, wire-length, power, and delay, are to be minimized under width constraint. Then an optimal solution might be a placement which achieves minimal wire-length, minimal power dissipation, with minimal delay and does not violate the width limitations. If such a solution exists, we actually only have to solve a single objective optimization (SOP). The optimal solution for any objective is also the optimum for other objectives (Zitzler, 1999). However, what makes MOP difficult is the common situation when the individual optima corresponding to the distinct objective functions are sufficiently different. Then the problem has usually no unique, perfect solution, but a set of equally efficient, or non-inferior, alternative solutions, known as the

Pareto-optimal set (Fandel and Gal, 1980). Placement is not far from this difficulty, because it is possible that for a particular change in placement solution there is a decrease in one cost but it may result in the increase in other cost. For example it is possible that certain change results in decrease in overall wire-length, but the wire-length of the nets having high switching probability may increase resulting in high-power dissipation, or the wire-length of the nets on long path may increase resulting in increased overall delay. Therefore, it is needed to solve placement problem as an MOP. There are several approaches to solve an MOP. The first is *Search and Decision Making*. Depending on how optimization and the decision process are combined, multiobjective optimization methods can be broadly classified into three categories (Zitzler, 1999): (a) Decision making before search, (b) Search before decision making, and (c) Decision making during search. Another aggregation method is *goal programming*, where the decision maker has to assign targets or goals that he/she wishes to achieve for each objective. This technique is useful if a linear or piecewise-linear approximation of the objective functions can be made. The third approach is to employ *fuzzy logic*, which we have adopted in this paper. Details of fuzzy logic are given below.

2.4. Fuzzy logic

A crisp set is normally defined as a collection of elements $x \in X$, where each element can either belong to a set or not. However, in real-life situations, objects do not have crisp [0 or 1] membership criteria. Fuzzy set theory (FST) aims to represent vague information, like "very hot" and "quite cold", which are difficult to represent in classical (crisp) set theory. In fuzzy sets, an element may partially belong to a set. Formally, a fuzzy set is characterized by a membership function which provides a measure of the degree of presence for every element in the set (Zadeh, 1973).

Like crisp sets, set operations such as union, intersection, and complementation, etc., are also defined on fuzzy sets. There are many operators for fuzzy union and fuzzy intersection. For fuzzy union, the 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. 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 (Yager, 1977). This category of operators allows easy adjustment of the degree of "anding" and "oring" embedded in the aggregation. "Orlike" and "Andlike" OWA for two fuzzy sets A and B are implemented as given in Eqs. (3) and (4), respectively,

$$\mu_{A\cup B}(x) = \beta \times \max(\mu_A, \mu_B) + (1-\beta) \times \tfrac{1}{2}(\mu_A + \mu_B), \quad (3)$$

$$\mu_{A\cap B}(x) = \beta \times \min(\mu_A, \mu_B) + (1-\beta) \times \tfrac{1}{2}(\mu_A + \mu_B), \quad (4)$$

where 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.

Fuzzy reasoning: Fuzzy reasoning is a mathematical discipline invented to express human reasoning in vigorous mathematical notation. Unlike classical reasoning in which propositions are either true or false, fuzzy logic establishes approximate truth value of propositions based on linguistic variables and inference rules. In order to represent imprecise ideas, Zadeh (1973) introduced the concept of linguistic variable. A linguistic variable is a variable whose values are words or sentences in natural or artificial language. The set of values a linguistic value can take is called a term set. This set is constructed by means of primary terms and by placing modifiers known as hedges such as "more", "many", "few", etc., before primary terms. The term set represents a precise syntax in order to form a vast range of values the linguistic variable can take. The linguistic variables can be composed to form propositions using connectors like AND, OR and NOT.

To solve MOP using fuzzy logic, at first all the objectives are defined in terms of linguistic variable. A linguistic rule is made using ("and" and "or" logic) in order to combine these linguistic variable. Each linguistic variable is also mapped to a fuzzy membership value in the fuzzy set of good in terms of that objective. This membership value is the functions of some base value based on the numerical value of the actual cost. All the membership values are combined into one membership value, using operators. In our work OWA is used. The selection of minor max OWA will depend on the predefined linguistic rule. The combined membership value is now used as aggregating function. The best solution is the one, which results in the highest combined membership value.

3. Problem formulation and fuzzy cost measure

Formally, the placement problem can be stated as follows: Given a set of modules $M = \{m_1, m_2, \ldots, m_n\}$, and a set of signals $V = \{v_1, v_2, \ldots, v_k\}$, each module $m_i \in M$ is associated with a set of signals V_{mi}, where $V_{mi} \subseteq V$. Also each signal $v_i \in V$ is associated with a set of modules M_{vi}, where $M_{vi} = \{m_j | v_i \in V_{mj}\}$. M_{vi} 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 width is considered as constraint (Sait and Youssef, 1995).

3.1. Cost functions

In this section cost estimations for the objective functions to be minimized and constraint to be satisfied are discussed.

Cost estimation for wire-length in the circuit layout: The wire-length cost can be computed by adding wire-length estimates for all the nets in the circuit (Sait and Youssef, 1995).

$$\text{Cost}_{\text{wire}} = \sum_{j \in M} l_j, \tag{5}$$

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.

Cost estimation for overall power in the circuit: In standard CMOS circuits, power dissipation is computed using the following equation:

$$P_t = \sum_{i \in M} \left(\frac{1}{2} C_i V_{DD}^2 f S_i \beta \right) + \sum_{i \in V} Q_{SC_i} V_{DD} f S_i + I_{leak} V_{DD}, \tag{6}$$

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, i.e., the number of gate output transition per clock cycle, and f is the clock frequency.

The first term in Eq. (6) represents the power required to charge or discharge the circuit nodes. Node total capacitance is denoted by C_i. β is the technology-dependent constant.

The second term in Eq. (6) represents the short-circuit current from V_{DD} to ground during output transition. Q_{SC_i} represents the quantity of charge carried by the short-circuit current per transition.

The third term represents the static power dissipation due to leakage current I_{leak}, where I_{leak} is the reverse-biased diode current between source and substrate and between drain and substrate.

In VLSI circuits with well-designed logic gates, over 90% power dissipation is due to the first term in Eq. (6), (Devadas and Malik, 1995; Chandrakasan et al., 1992). Therefore, most of the work was done to reduce dissipation due to this component. Also in the case of standard cell placement, cells are obtained from a technology library so not much can be done to reduce the second and third term in Eq. (6). Since this term contributes around 90% of the overall power, therefore, Eq. (6) is approximated as follows:

$$P_t \simeq \sum_{i \in M} \frac{1}{2} C_i V_{DD}^2 f S_i \beta. \tag{7}$$

Assuming that clocking frequency and power voltages are fixed, the total power dissipation of the circuit is a function of total capacitance and the switching probabilities of 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 interconnect capacitance at the cell output node, as shown in the following equation:

$$C_i = C_i^r + \sum_{j \in M_i} C_j^g, \tag{8}$$

where C_j^g is the input capacitance for gate j, C_i^r represents the interconnect capacitance at the output node of cell i.

In case of standard cell design, cell properties are fixed for a particular library, hence C_j^g cannot be reduced, therefore in standard cell placement cost due to the overall power in VLSI circuits can be termed as,

$$\text{Cost}'_{\text{power}} = \sum_{i \in M} S_i C_i^r. \tag{9}$$

Also interconnect capacitances are related to the corresponding interconnect wire-lengths, therefore $\text{Cost}'_{\text{power}}$ can be further modified as

$$\text{Cost}_{\text{power}} = \sum_{i \in M} S_i l_i. \tag{10}$$

Cost estimation for the delay in the circuit: The overall performance of VLSI circuits depends upon how fast it can process signals, i.e., its clock speed. The propagation delay of signals in a VLSI circuit consists of two elements, switching delay of gates and interconnect delay. Due to improved technology, libraries are available with considerably low switching delay. This along with increased gate density in the chip make the interconnect delay factor prominent in overall circuit delay. Thus, most of the work now is in reducing the interconnect delay (Sait and Youssef, 1995).

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

$$T_\pi = \sum_{i=1}^{k} (\text{CD}_i + \text{ID}_i), \tag{11}$$

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

Placement is independent of CD_i as it is technology-dependent parameter, therefore, this work is concentrating on reducing ID_i. Using the RC delay model, this delay is due to the load factor, the interconnect

resistance, and the load capacitance, as given in Eq. (12)

$$\mathrm{ID}_i = (\mathrm{LF}_i + R_i^{\mathrm{r}}) \times C_i, \tag{12}$$

where LF_i is load factor of the driving block, that is 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 Eq. (8). The cost function due to timing performance in the placement problem can be expressed as

$$\mathrm{Cost}_{\mathrm{delay}} = T_{\pi_c}. \tag{13}$$

Layout width: In standard-cell design, cells have fixed height and variable widths. Cells are placed in rows having same heights and separated by routing channels. The overall area of the layout is represented by the rectangle that bounds all the rows and routing channels. In this work the channel heights are initially estimated, using an area efficient placement tool and assumed to be fixed. This leaves only layout width that effects the layout area. In this work layout width is considered as a constraint. The upper limit on the layout width is defined as

$$\mathrm{Width}_{\max} = (1 + a) \times \mathrm{Width}_{\mathrm{opt}}, \tag{14}$$

where $\mathrm{Width}_{\max}$ is the maximum allowable width of the layout, $\mathrm{Width}_{\mathrm{opt}}$ is the minimum possible layout width obtained by adding the widths of all cells and divided by number of rows in the layout and a is a user-defined parameter that denotes how wide a layout can be as compared to $\mathrm{Width}_{\mathrm{opt}}$.

3.2. Fuzzy goal-based aggregation

In order to select the best solution found so far, a goal-directed search approach is adopted, where the best placement is the one that satisfies a user-specified vector of fuzzy goals as much as possible (Hussain, 1998). In order to combine three parameters and one constraint, the following fuzzy rule is proposed.

Rule R0: 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 ordered weighted averaging operator (OWA) (Yager, 1988), the above fuzzy rule translates to the following:

$$\mu^{\mathrm{c}}_{\mathrm{pdl}}(x) = \beta^{\mathrm{c}} \times \min(\mu^{\mathrm{c}}_{\mathrm{p}}(x), \mu^{\mathrm{c}}_{\mathrm{d}}(x), \mu^{\mathrm{c}}_{\mathrm{l}}(x)) + (1 - \beta^{\mathrm{c}}) \times \frac{1}{3} \sum_{j=\mathrm{p,d,l}} \mu^{\mathrm{c}}_j(x), \tag{15}$$

$$\mu^{\mathrm{c}}(x) = \min(\mu^{\mathrm{c}}_{\mathrm{pdl}}(x), \mu^{\mathrm{c}}_{\mathrm{width}}(x)), \tag{16}$$

where $\mu^{\mathrm{c}}(x)$ is the membership of solution x in fuzzy set of acceptable solutions, $\mu^{\mathrm{c}}_{\mathrm{pdw}}(x)$ is the membership in fuzzy set of "acceptable power AND acceptable delay AND acceptable wire-length", whereas $\mu^{\mathrm{c}}_j(x)$ for $j = \mathrm{p, d, w, width}$, are the individual membership values in the fuzzy sets *within acceptable power*, *delay*, *wire-length*, *and layout width*, respectively. In our case we chose $\beta^{\mathrm{c}} = 0.7$, the superscript c represents "cost". The solution that results in maximum value of $\mu^{\mathrm{c}}(x)$ is reported as the best solution found by the search heuristic. 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.

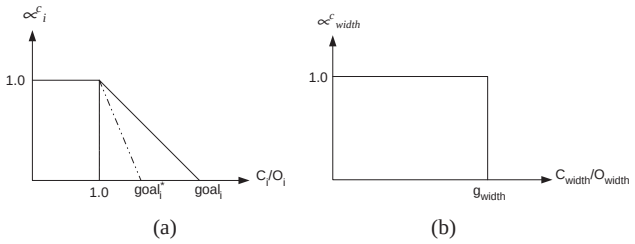

Fig. 3. Membership functions within acceptable range.

The shape of membership functions for fuzzy sets *within acceptable power*, *delay and wire-length* are as shown in Fig. 3(a), whereas the constraint *within acceptable layout width* is given as a crisp set as shown in Fig. 3(b). Since layout width is a constraint, its membership value is either 1 or 0 depending on $\mathrm{goal}_{\mathrm{width}}$ (in our case $\mathrm{goal}_{\mathrm{width}} = 1.25$). However, for other objectives, by increasing or decreasing the value of goal_i one can vary its preference in the overall membership function. $O_i s$ for $i \in \{\mathrm{w, p, d, width}\}$ represent the lower bounds for wire-length, power, delay and layout width, respectively.

4. Simulated evolution for performance-driven, low-power placement

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 (Sait and Youssef, 1999).

Simulated Evolution (SE) is a general iterative heuristic proposed by Ralph Kling (1990). 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 (Sait and Youssef, 1999). 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. SE adopts the generic state model described above, where a solution is seen as a population of movable elements. Each element i is characterized by a *goodness* measure $g_i \in [0,1]$ where $g_i = O_i/C_i$. C_i is the estimated real cost of element e_i in its position in current state, and O_i is a lower bound on the cost of e_i. The main loop of the algorithm consists of three steps as depicted in Fig. 4: *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. The *evaluation* step estimates the *goodness* of each element in its current location. The goodness of an element is a ratio of its optimum cost to its actual cost estimate, and therefore belongs to the interval [0,1]. It is a measure of how near each element is to its optimum position. The higher the goodness of an element, the closer is that element to its optimum location with respect to the current configuration. In *selection* step, the algorithm probabilistically selects elements for relocation. Elements with low goodness values have higher probabilities of getting selected. A selection *bias* (B) is used to compensate for errors made in the estimation of goodness. Its objective is to inflate or deflate the goodness of elements. A high positive value of *bias* decreases the probability of selection and vice versa.

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 $Random > Min(g_i + B, 1)$
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. 4. Structure of the simulated evolution algorithm.

Large selection sets also degrade the solution quality due to uncertainties created by large perturbations. Similarly, for high bias values the size of the selection set is small, which degrades the quality of solution due to limitations of the algorithm to escape local minima. A carefully tuned bias value results in good solution quality and reduced execution time (Kling, 1990). 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*.

4.1. Fuzzy simulated evolution (FSE)

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, the 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 method 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*.

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.

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

$$g_i = \mu_i^{\mathrm{e}}(x) = \beta^{\mathrm{e}} \times \min(\mu_{i\mathrm{w}}^{\mathrm{e}}(x), \mu_{i\mathrm{p}}^{\mathrm{e}}(x), \mu_{i\mathrm{d}}^{\mathrm{e}}(x)) + (1-\beta^{\mathrm{e}}) \times \frac{1}{3} \sum_{j=\mathrm{w,p,d}} \mu_{ij}^{\mathrm{e}}(x), \quad (17)$$

where

$$\mu^{e}_{id}(x) = \beta^{e}_{d} \times \max(\mu^{e}_{inet}(x), \mu^{e}_{ipath}(x)) + (1 - \beta^{e}_{d}) \times \frac{1}{2}(\mu^{e}_{inet}(x) + \mu^{e}_{ipath}(x)) \quad (18)$$

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

$\mu^{e}_{ipath}(x)$ is included in the computation of $\mu^{e}_{id}(x)$ 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, $\{v_1, v_2, \ldots, v_k\}$ 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 Eqs. (19)–(22) below

$$X_{iw}(x) = \frac{\sum_{j=1}^{k} l^{*}_{j}}{\sum_{j=1}^{k} l_j}, \quad (19)$$

$$X_{ip}(x) = \frac{\sum_{j=1}^{k} S_j l^{*}_{j}}{\sum_{j=1}^{k} S_j l_j}, \quad (20)$$

$$X_{inet}(x) = \frac{ID^{*}_{i} + ID^{*}_{ip}}{ID_i + ID_{ip}}, \quad (21)$$

$$X^{e}_{ipath}(x) = \frac{T_{max}}{T_{max}(i)}, \quad (22)$$

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 Fig. 5.

The values of a_{min} and a_{max} depend on the statistical nature of the base values. A typical frequency of occurrence plot is shown in Fig. 6, where we have plotted $X^{e}_{net}(x)$ and $X^{e}_{w}(x)$ 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

$$a_{min\_i} = \bar{X}_i - 2\sigma_i \quad \text{and} \quad a_{max\_i} = \bar{X}_i + 2\sigma_i. \quad (23)$$

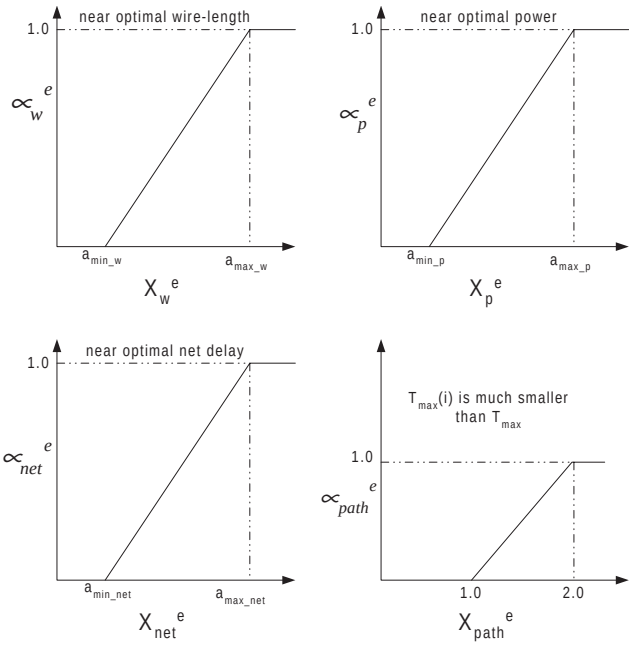

Fig. 5. Membership functions used in fuzzy evaluation.

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% of the initial value.

Selection: As mentioned earlier, in this stage of the algorithm some elements (i.e., cells) are selected probabilistically depending on their goodness values. Bias concept in selection step, present in the original SE algorithm (Kling and Banerjee, 1989), is the major drawback of the heuristic. It is not easy to determine the value of bias because it varies from problem to problem. Also in the case of placement it varies from circuit to circuit. To overcome this problem an adaptive bias scheme (**ABFSE**) proposed in Youssef et al. (2001) is used. According to this scheme the value of bias for kth iteration is computed as follows:

$$B_k = 1 - \bar{G}_{k-1}, \quad (24)$$

where $\bar{G}_k$ and B_k are the average goodness of cells and bias value for kth iteration.

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:

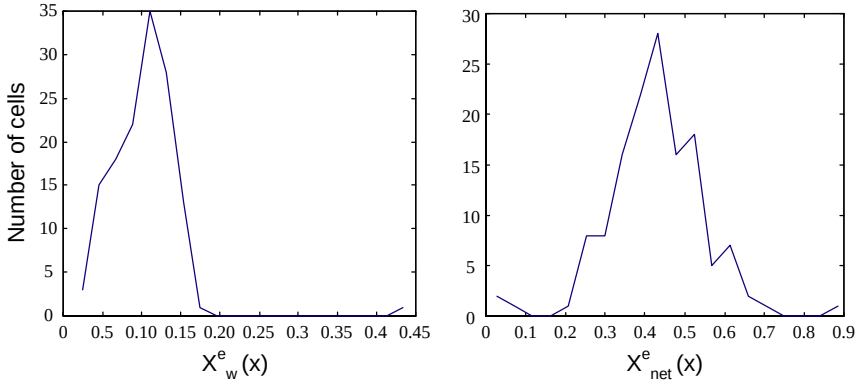

Fig. 6. Base values versus frequency of occurance plot.

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:

$$\mu^{\mathrm{a}}_{i\_\mathrm{pwd}}(l) = \beta^{\mathrm{a}} \times \min(\mu^{\mathrm{a}}_{i\mathrm{p}}(l), \mu^{\mathrm{a}}_{i\mathrm{w}}(l), \mu^{\mathrm{a}}_{i\mathrm{d}}(l)) + (1-\beta^{\mathrm{a}}) \times \frac{1}{3} \sum_{j=\mathrm{p,w,d}} \mu^{\mathrm{a}}_{ij}(l), \tag{25}$$

$$\mu^{\mathrm{a}}_{i}(l) = \min(\mu^{\mathrm{a}}_{i\_\mathrm{width}}(l), \mu^{\mathrm{a}}_{i\_\mathrm{pwd}}(l)). \tag{26}$$

The superscript a is used here to represent allocation. $\mu^{\mathrm{a}}_{i}(l)$ is the membership of cell i at location l in the fuzzy set of good location. $\mu^{\mathrm{a}}_{i\_\mathrm{pwd}}(l)$ is the membership in the fuzzy set of "reduced wire-length and reduced power and reduced delay". $\mu^{\mathrm{a}}_{i\mathrm{w}}(l)$, $\mu^{\mathrm{a}}_{i\mathrm{p}}(l)$, $\mu^{\mathrm{a}}_{i\mathrm{d}}(l)$, and $\mu^{\mathrm{a}}_{i\mathrm{width}}(l)$ 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 Eqs. (27)–(30):

$$X^{\mathrm{a}}_{i\mathrm{w}}(l) = \frac{(\sum_{m=1}^{k_i} l_{im} + \sum_{m=1}^{k_j} l_{jm})_n}{(\sum_{m=1}^{k_i} l_{im} + \sum_{m=1}^{k_j} l_{jm})_{n-1}}, \tag{27}$$

$$X^{\mathrm{a}}_{i\mathrm{p}}(l) = \frac{(\sum_{m=1}^{k_i} S_{im} l_{im} + \sum_{m=1}^{k_j} S_{jm} l_{jm})_n}{(\sum_{m=1}^{k_i} S_{im} l_{im} + \sum_{m=1}^{k_j} S_{jm} l_{jm})_{n-1}}, \tag{28}$$

$$X^{\mathrm{a}}_{i\mathrm{d}}(l) = \frac{(\mathrm{ID}_i + \mathrm{ID}_{i\mathrm{p}} + \mathrm{ID}_j + \mathrm{ID}_{j\mathrm{p}})_n}{(\mathrm{ID}_i + \mathrm{ID}_{i\mathrm{p}} + \mathrm{ID}_j + \mathrm{ID}_{j\mathrm{p}})_{n-1}}, \tag{29}$$

$$X^{\mathrm{a}}_{i\_\mathrm{width}}(l) = \frac{\mathrm{Width}_n}{\mathrm{Width}_{\mathrm{opt}}}, \tag{30}$$

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

Membership functions for these base values are shown in Fig. 7. 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 $a_{\mathrm{w}} = 0.75$, $a_{\mathrm{p}} = 0.75$, $a_{\mathrm{d}} = 0.85$ and $a_{\mathrm{width}} = 0.25$.

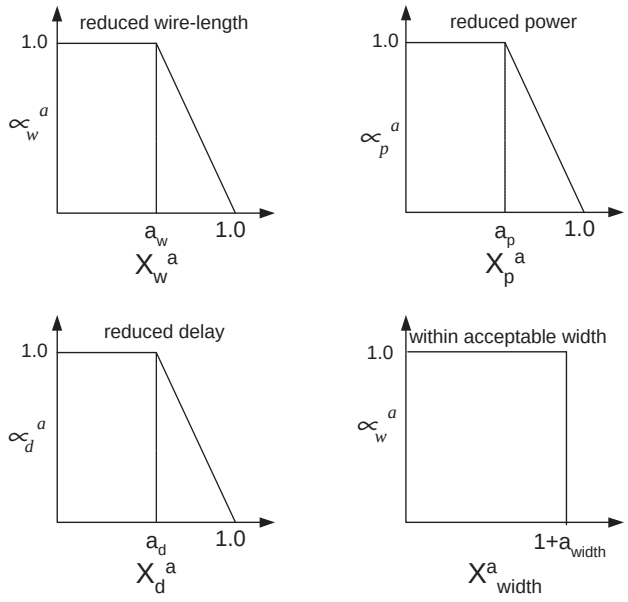

Fig. 7. Membership functions used in allocation.

4.2. Biasless simulated evolution (BLFSE)

As mentioned earlier, in SE it is not possible to find accurate goodness value for a cell, because accurate optimum location of a cell is unknown. Further, goodness only gives the local view of cell, and in order to compensate the error in goodness calculation and to limit the size of selection set, authors in Kling and Banerjee (1989) proposed a bias parameter where a cell i is selected if $\mathrm{goodness}_i + B < \mathrm{Random}$, where Random is a uniformly distributed random number in the range $[0, 1]$, B is the selection bias with its typical values in the range $[-0.2, 0.2]$, and $\mathrm{goodness}_i$ is the goodness value of

cell i. However, it is not easy to select the value of B, because it varies for each problem instance and requires the undesirable human interaction. Addressing this problem an adaptive bias scheme was proposed in Youssef et al. (2001), where in kth iteration bias B_k is computed as follows:

$$B_k = 1 - G_{k-1} \tag{31}$$

where G_{k-1} is the average goodness of all cells at the end of $k-1$th iteration. This has the following advantages.

- No trial runs are required to find the fixed bias value and bias is adjusted automatically according to problem state.
- For poor quality solutions G_k is low, resulting in high bias value and low cardinality of selection set, avoiding large unnecessary perturbations.
- In any iteration only cells having goodness $< G_{k-1}$ have non-zero probability of selection, thus search is always focused on relocating poorly placed cells.

However, along with these benefits the adaptive bias scheme has following drawbacks,

- All the cells with goodness$_i > G_{k-1}$ have zero probability of selection. This fact may lead the search to some local optimal solution, because statistically half of the cells have zero probability of selection.
- If G_{k-1} is low then size of selection set is small and when G_{k-1} is high then size of selection set is large, which contradicts the basic idea of Simulated Evolution algorithm, where size of the selection set has to be decreased with increase in average goodness. Also it is against the behavior of any other iterative search algorithm where big perturbations are made when the solution is bad and smaller perturbations are made with improvement in the quality of solution.

To solve the above problems a *biasless selection* scheme is proposed in this work. In the next section the biasless selection scheme is explained.

Biasless selection: In this scheme the selection bias B is totally eliminated and a cell is selected if Random > goodness$_i$. This can be done as follows.

When number of cells in the problem is large, as in the case of VLSI placement, the goodness distribution among the cells is Gaussian, with mean G_m and standard deviation G_σ. In the proposed selection scheme, instead of using uniformly distributed random number, a Gaussian random number is used. By using Gaussian random number the problem of having cells with zero selection probability can be avoided. The mean R_m and standard deviation R_σ of the random number are calculated as follows:

$$R_m = G_m - G_\sigma, \tag{32}$$

$$R_\sigma = G_\sigma. \tag{33}$$

If we use G_m as the mean of random number then it is most likely that around 50% cells will be selected which is not desirable in case of large number of cells. To avoid larger selection set we change the mean of random number to $G_m - G_\sigma$, so that only 12–13% cells will be selected in the initial iterations. These values are determined only in the first iteration. However, with increased number of iterations average goodness will increase and this may cause very small number of cells to be selected. To avoid this, the mean of random number is updated when the number of selected cells goes down to 5% of total number of cells in the problem, as follows:

$$R_m = R_m + 0.1 \times R_\sigma. \tag{34}$$

By using this scheme, (1) there is no cell in the design having zero selection probability, hence avoiding local optima, (2) size of selection set reduces with increase in the average goodness value, and (3) update procedure avoids the extremely small selection set.

5. Experiments and results

In this section, results obtained from different schemes of Simulated Evolution are discussed. Details of the different circuits used in the experiments conducted using these techniques are given.

In the experiments 13 different ISCAS-85 benchmark circuits are used. We have chosen sequential circuits because in real life most of the circuits are sequential in nature. Also sequential circuits have more severe delay problems than combinational circuits because most of the combinational circuits are well structured. For each circuit, number of cells in the circuit, number of rows in the layout and average channel height are given in Table 1.

Number of rows in layout and average channel heights are estimated by using a min-cut based layout placer. The details of the cell characteristics are obtained from the 0.25 μ MOSIS TSMC CMOS technology library (Tanner Consulting and Engineering Services).

5.1. Comparison of ABFSE and BLFSE

The results of fuzzy-based adaptive bias (ABFSE) are compared with BLFSE in Table 2.

In all the cases it is seen that BLFSE performs better than ABFSE in terms of all objectives. However, in terms of execution time BLFSE and ABFSE are same in most of the cases.

In order to compare improvement in the quality of solution versus execution time, different costs are plotted versus execution time in Fig. 8 for BLFSE and

in Fig. 9 for ABFSE. It can be observed that quality of solution improves rapidly in case of BLFSE-based search as compared to ABFSE. The reason is that in BLFSE, at the beginning big compound moves improve the quality of solution rapidly and the size of the selection set decreases as solution moves towards its optimal location. In ABFSE it is totally different, where from the beginning smaller compound moves are made hence improvement in the quality of solution is slow from the beginning.

The size of selection set versus execution time is shown in Fig. 8(f) for BLFSE and in Fig. 9(f) for ABFSE. It can be seen that in ABFSE size of selection set increases slightly when the solution moves towards optimal solution, leading to larger compound moves (perturbations). It is against the basic idea of any search algorithm, because larger perturbations are desired when the solution is away from optimal and smaller perturbations are desired when the solution is near its optimal location in the search space. In case of BLFSE the size of selection set decreases with the improved quality of solution, which is desirable.

In both cases the layout width versus number of iterations are shown in Figs. 8(e) and 9(e). It can be seen that width is always under the limit set for it, that is width constraint is always satisfied.

In Fig. 10, cardinality of selection set (number of cells selected) is plotted against number of solutions having this cardinality. It can be observed that in BLFSE most of the solutions have smaller selection set as compared to ABFSE, although some solutions have very large selection set in BLFSE in the beginning. Also, because of the use of Gaussian random number in selection set no cell has zero probability of selection, as in the case of ABFSE.

Figs. 11(a) and 12(a) show the quality of solution subspace searched by BLFSE and ABFSE. It is evident that both schemes concentrated in high membership subspace which indicates that these are properly engineered to solve the placement problem. These figures also shows that BLFSE put more effort in high membership subspace as compared to ABFSE.

Figs. 11(b) and 12(b) track with time the total number of solutions found by BLFSE and ABFSE for various membership ranges. These are very informative plots as they show, that as time progressed, the solutions found by each scheme were getting better. Note however that BLFSE exhibits much faster evolutionary rate than ABFSE. For example, after about 300 s, almost all new solutions discovered by BLFSE have membership in the range 0.6–0.8 in the fuzzy subset of good solutions with respect to all objectives, and almost none were found with lower membership values (see Fig. 11(b)). In contrast, for ABFSE, it is only after 1000 s that the first solution with membership in the interval 0.6–0.8 was found (see Fig. 12(b)). This behavior was observed for all circuits.

In general we can say that BLFSE performs better in terms of final solution. Also, quality of solution

Table 1
Circuit and layout details

Circuit		Layout	
Name	Number of cells	Number of rows	Average channel height in μm
S2081	122	4	6.66
S298	136	5	7.08
S386	172	5	7.68
S641	433	7	9.72
S832	310	7	9.78
S953	440	8	11.76
S1196	561	9	11.58
S1238	540	9	11.64
S1488	667	11	10.92
S1494	661	11	10.56
S3330	1961	17	11.46
S5378	2993	20	12.3
S9234	5844	28	12.36

Table 2
Layout found by BLFSE, and ABFSE. "L", "P" and "D" represent the wire-length, power, and delay costs and "T" represents execution time in seconds

Circuit	BLFSE				ABFSE			
	L (μm)	P (μm)	D (ps)	T (s)	L (μm)	P (μm)	D (ps)	T (s)
S298	4548	915	139	46	7130	1395	152	21
S386	8357	2036	203	117	11 167	2544	221	33
S832	23 140	5251	416	192	28 537	6577	485	114
S641	12 811	3072	687	175	13 773	3107	687	264
S953	29 576	5025	223	351	33 484	5523	250	130
S1238	41 318	12 303	363	699	45 140	13 870	397	295
S1196	35 810	11 276	360	613	41 861	12 918	357	433
S1494	54 523	12 986	768	762	67 944	16 091	809	279
S1488	57 730	13 810	700	374	73 696	17 511	891	216
S3330	183 288	24 797	459	5351	193 731	25 373	558	5610
S5378	326 840	48 360	435	11 823	365 204	56 001	441	11 369

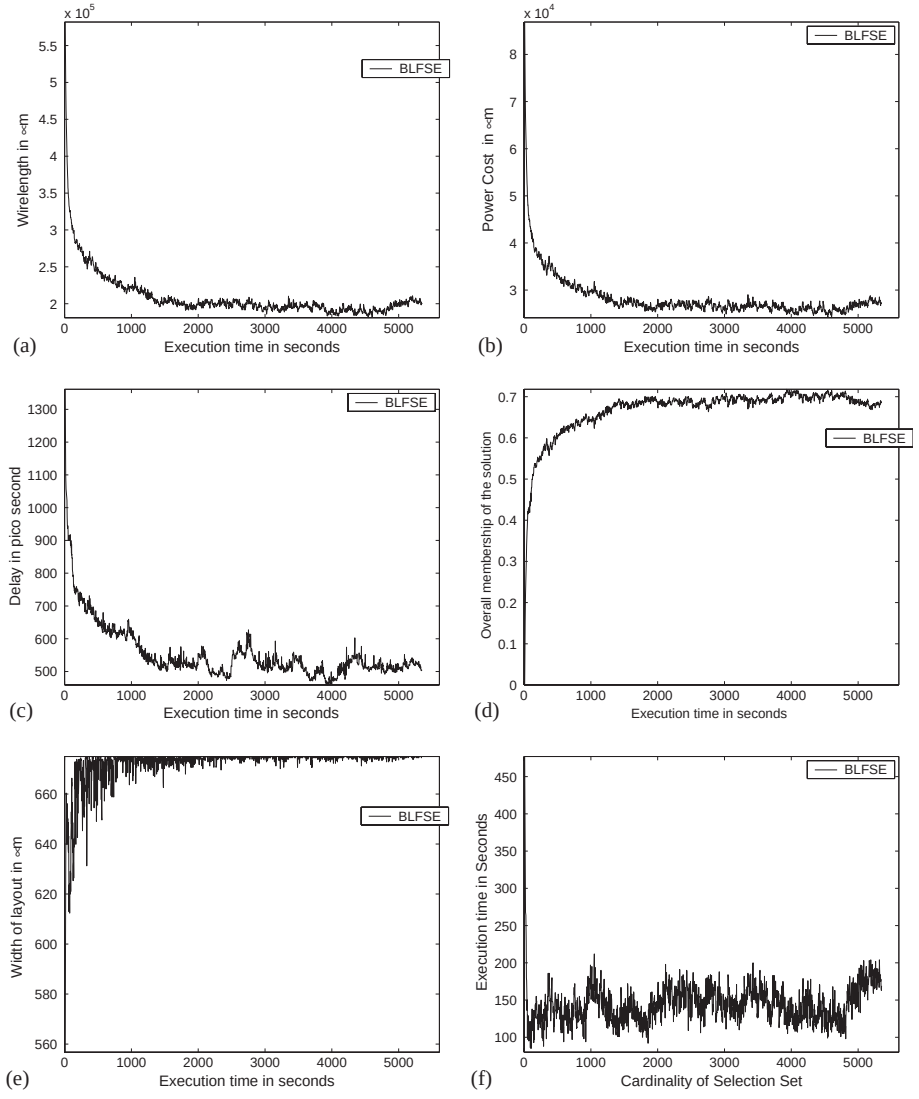

Fig. 8. BLFSE (a) wire-length (μm), (b) power cost (μm), (c) delay (ps) (d) overall membership, (e) layout width (μm) and (f) number of selected cells versus execution time in seconds for S3330, using BLFSE.

improves rapidly in case of BLFSE as compared to ABFSE.

5.2. GA-based optimization

For the comparison purpose we have also implemented GA (Sait and Youssef, 1999). Membership value in the fuzzy set of acceptable solution given in Eq. (16) is used as the *fitness* measure of a chromosome (solution). In parent selection step for crossover *roulette wheel* selection scheme (Sait and Youssef, 1999) is used. *Partially mapped crossover* (*PMX*) is used to generate new offsprings. For 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 criterion is the maximum number of generations.

We have applied GA on ISCAS-85 and ISCAS-89 benchmark circuits. For GA, stopping criterion is 10,000 generations.

To observe the improvement in quality of solution versus time, we have plotted the average and best fitness (membership) values in current population obtained by GA versus execution time in Fig. 13(a) and (b). These plots are for test case S1196.

Fig. 14(a) and (b) shows the quality of solution subspace searched by GA, which required generations in the order of thousands, where each generation consisted of 32 solutions.

We also experimented with GA and compared its performance with BLFSE on number of ISCAS-89 benchmark circuits. Table 3 compares the quality of

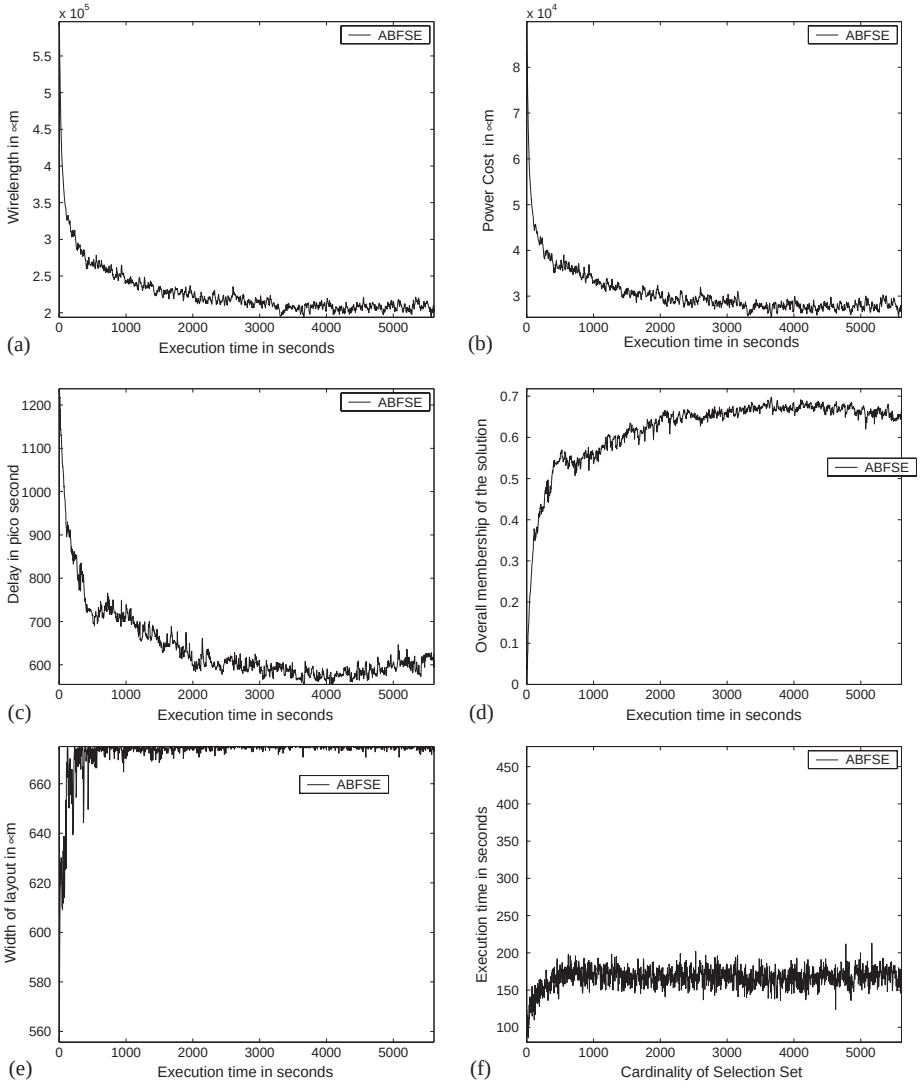

Fig. 9. ABFSE (a) wire-length (μm), (b) power cost (μm), (c) delay (ps) (d) overall membership, (e) layout width (μm) and (f) number of selected cells versus execution time in seconds for S3330, using adaptive bias.

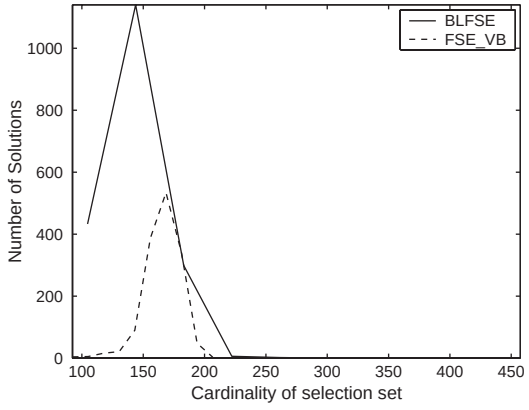

Fig. 10. Comparison of BLFSE and ABFSE with respect to Cardinality of selection set, for S3330.

final solution generated by GA and BLFSE. The circuits are listed in order of their complexity. Here "L", "P" and "D" represent the wire-length, power, and delay costs respectively, and "T" represents execution time in seconds.

We observe from Table 3 that for all the three objectives (i.e., wire-length, power, and delay), BLFSE performs better than GA. The exceptions are *s298*, *s386*, and *s832*, where BLFSE has inferior solutions than GA, for all three objectives. Also, BLFSE has much less execution time than GA for all cases, because GA operates on a population of solutions in each iteration, while BLFSE mutates one solution per iteration.

6. Conclusion

In this work, we have engineered Simulated Evolution (SimE) algorithm for a hard multiobjective optimization problem of VLSI standard cell placement. An effort is

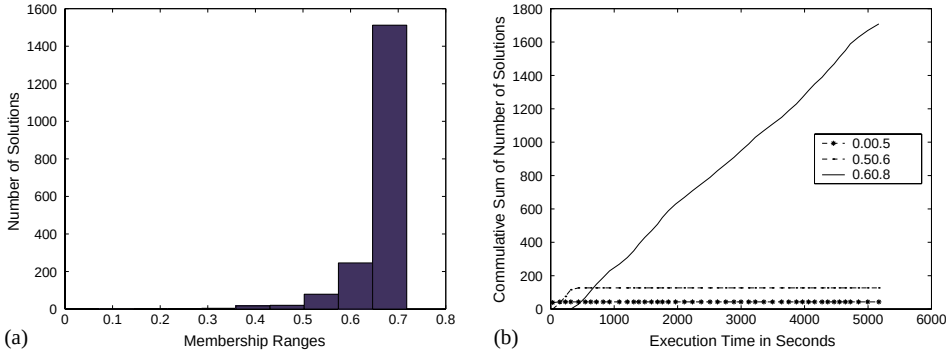

Fig. 11. (a) Number of solutions versus membership range. (b) Cumulative number of solution versus execution time in seconds for different membership ranges, using BLFSE.

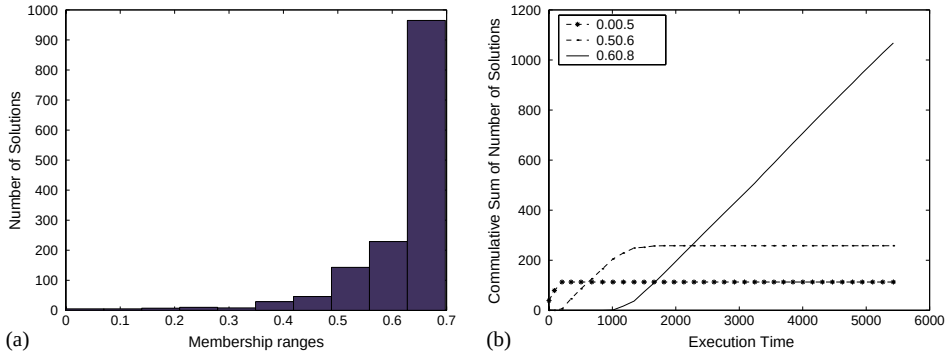

Fig. 12. (a) Number of solutions versus membership range. (b) Cumulative number of solution versus execution time in seconds for different membership ranges, using ABFSE.

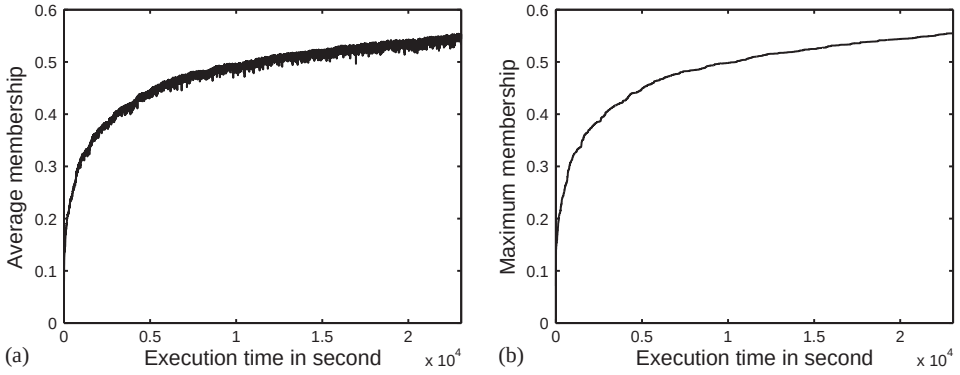

Fig. 13. Behavior of GA. (a) and (b) represent average and best solution obtained by GA, plotted versus execution time in seconds.

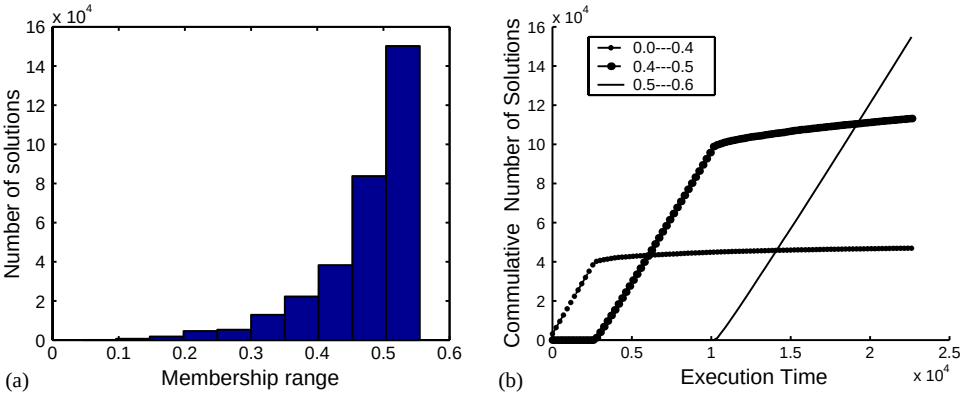

Fig. 14. Performance of GA based on search efforts in particular membership ranges; (a) shows number of solutions visited in particular membership ranges for GA (b) shows cumulative number of solutions visited in a specific membership range versus execution time in seconds for GA.

Table 3
Comparison between GA and BLFSE

Circuit	GA				BLFSE			
	L (μm)	P (μm)	D (ps)	T (s)	L (μm)	P (μm)	D (ps)	T (s)
S298	4062	838	130	2922	4548	915	139	46
S386	6824	1665	193	3945	8357	2036	203	117
S641	18 320	4365	736	21 982	12 811	3072	687	175
S832	21 015	4787	395	7206	23 140	5251	416	192
S953	32 031	5156	230	11 221	29 526	5025	223	351
S1196	51 804	15 259	370	23 070	35 810	11 276	359	613
S1238	52 679	15 473	410	16 208	41 318	12 303	362	699
S1488	69 792	17 346	784	21 434	57 730	13 810	700	374
S1494	71 021	17 497	803	26 032	54 523	12 986	768	762

made to simultaneously optimize three objectives, namely, power dissipation, performance, and interconnect wire-length. The incorporation of fuzzy logic is suggested to integrate the cost values of three objectives in an aggregating cost function. Furthermore, different variants of SimE are suggested. The experimental results for ISCAS benchmarks indicate the Biasless SimE (BLFSE) has better performance than Adaptive Bias SimE (ABFSE). Moreover, BLFSE has a better performance than genetic algorithm.

Acknowledgements

Authors and research team acknowledge King Fahd University of Petroleum & Minerals for all support under research COE/ITERATE/221. Thanks to Mr. Salman Khan for his assistance in editing and proof-reading of the manuscript.

References

Brayton, R., et al., 1987. A multiple-level logic optimization system. IEEE Transaction on Computer Aided Design of Integrated Circuits, CAD 6 (6), 1062–1081.

Bright, M.S., Arslan, T., 1997. A genetic algorithm for the high-level synthesis of DSP systems for low power. Genetic Algorithms in Engineering Systems: Innovations and Applications, Glasgow, UK 446, 174–179.

Chandrakasan, A.P., 1994. Low power digital CMOS design. Ph.D. Thesis, University of California at Berkeley, August 1994.

Chandrakasan, A., Sheng, T., Brodersen, R.W., 1992. Low power CMOS digital design. Journal of Solid State Circuits 4 (27), 473–484.

Chao, K.-Y., Wong, D.F., 1995. Floorplanning for low power design. IEEE International Symposium on Circuits and Systems, Seattle, WA, USA 1, 45–48.

Chatterjee, A., Roy, R., 1994. Synthesis of low power linear DSP circuits using activity metrics. In: International Conference on VLSI Design, India, January 1994.

Chren, W.A., 1995. Low delay power product CMOS design using one-hot residue cooling. In: Proceedings of the International Symposium on Low Power Design, Laguna, CA, USA, April 1995.

Darringer, J., et al., 1984. LSS: a system for production logic synthesis. IBM Journal of Research Development 5 (28), 259–274.

Devadas, S., Malik, S., 1995. A survey of optimization techniques targeting low power VLSI circuits. 32nd ACM/IEEE Design Automation Conference, San Francisco, CA, USA.

Dunlop, A.E., et al., 1984. Chip layout optimization using critical path weighting. Proceedings of 21st Design Automation Conference, New Mexico, USA, pp. 133–136.

Fandel, G., Gal, T., (Eds.) 1980. Multiple Criteria Decision Making Theory and Applications. Lecture Notes in Economics and Mathematics Systems, Springer, Berlin, p. 177.

Ghosh, A., et al., 1992. Estimation of average switching activity in combinational and sequential circuits. In: Proceedings of the 29th Design Automation Conference, Anaheim, CA, USA, June 1992, pp. 253–259.

Hakenes, R., Manoli, Y., 1999. Improving micro-controller power consumption through a segmented gray code program counter. IEEE International Conference on Computer Design, Austin, Texas, USA, October 1999, pp. 277–278.

Holt, G., Tyagi, A., 1995. EPNR: an energy-efficient automated layout synthesis package. IEEE International Conference on VLSI in Computers and Processors, October 1995, pp. 224–229.

Holt, G., Tyagi, A., 1996. GEEP: a low power genetic algorithm layout system. IEEE 39th Midwest Symposium on Circuits and Systems, Ames, Iowa, Vol. 3, August 1996, pp. 1337–1340.

Hussain, A.S., 1998. Fuzzy simulated evolution algorithm for VLSI cell placement. Master Thesis, King Fahd University of Petroleum and Minerals, Dhahran, December 1998.

Igarashi, M., et al., 1997. A low-power design method using multiple supply voltage. IEEE International Symposium on Low Power Electronics and Design, Monterey, CA, USA, August 1997, pp. 36–41.

Kling, R.M., 1990. Optimization by simulated evolution and its application to cell placement. Ph.D. Thesis, University of Illinois, Urbana.

Kling, R.M., Banerjee, P., 1989. ESP: placement by simulated evolution. IEEE Transaction on Computer-Aided Design 3 (8), 245–255.

Lemonds, C., Shetti, S.S.M., 1994. A low power 16 by 16 multiplier using transition reduction circuitry. In: Proceedings of the International Conference on Low Power Design, California, April 1994, pp. 139–142.

Nagendra, C., Irwin, M.J., Owens, R.M., 1996. Area-time-power tradeoffs in parallel adders. IEEE Transactions on Circuits and Systems—II Analog and Digital Signal Processing 43 (10), 689–702.

Narayanan, U., Stamoulis, G.I., Roy, R., 1999. Characterizing individual gate power sensitivity in low power design. 12th

International Conference on VLSI Design, Goa, India, January 1999, pp. 625–628.

Panda, P.R., Dutt, N.D., 1999. Low-power memory mapping through reducing address bus activity. IEEE Transaction on Very Large Scale Integration (VLSI) Systems 7 (3), 309–320.

Prabhakaran, P., Banerjee, P., et al., 1999. Simultaneous scheduling, binding and floorplanning for interconnect power optimization. 12th International Conference on VLSI Design, Goa, India, January 1999, pp. 423–427.

Ramprasad, S., Shanbhag, N.R., Hajj, I.N., 1999. A coding framework for low-power address and data busses. IEEE Transactions on Very Large Scale Integrated (VLSI) Systems 7 (2), 212–221.

Roy, K., 1999. Power-dissipation driven FPGA place and route under timing constraints. IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Application 46 (5), 634–637.

Roy, K., Prasad, S., 1992. SYCLOP: synthesis of CMOS logic for low power applications. In: Proceedings of the International Conference on Computer Design: VLSI in Computer and Processors, Cambridge, MA, USA, October 1992, pp. 464–467.

Sait, S.M., Youssef, H., 1995. VLSI Physical Design Automation: Theory and Practice. McGraw-Hill Book Company, Europe.

Sait, S.M., Youssef, H., 1999. Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems. IEEE Computer Society Press, California, December 1999.

Shen, A., et al., 1992. On average power dissipation and random pattern testability of combinational logic circuits. In: Proceedings of the International Conference on Computer-Aided Design, Santa Clara, CA, USA, November 1992, pp. 402–407.

Stan, M., Burleson, W., 1994. Limited weight codes for low power I/O. In: Proceedings of the International Workshop on Low Power Design, California, USA, April 1994, pp. 209–214.

Sundararajan, V., Parhi, K.K., 1999. Low power gate resizing of combinational circuits by buffer re-distribution. Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI, Atlanta, GA, USA.

Sutanthavibul, S., Shragowitz, E., Rung-Bin Lin, 1993. An adaptive timing-driven placement for high performance VLSI's. IEEE Transactions on Computer Aided Design 10(12), 1488–1489.

Tanner Consulting and Engineering Services, 2000. Digital Low Power Standard Cell Library for MOSIS TSMC CMOS 0.25 Process Deep Sub-Micron Technology, Tanner Research, Inc.

Usami, K., et al., 1996. Low power design techniques for ASICs by partially reducing supply voltage. Proceedings IEEE International ASIC Conference, Rochester, NY, September 1996, pp. 301–304.

Vaishnav, H., Pedrarn, M., 1993. PCUBE: a performance driven placement algorithm for low power design. IEEE Design Automation Conference, with Euro-VHDL, Dallas, Texas, pp. 72–77.

Vaishnav, H., Pedram, M., 1999. Delay-optimal clustering targeting low-power VLSI circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18 (6), 799–812.

Yager, R., 1977. Multiple objective decision-making using fuzzy sets. International Journal of Man–Machine Studies, pp. 9, 375–382.

Yager, R.R., 1988. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on Systems, Man, and Cybernetics 18 (1), 183–190.

Youssef, H., Rung Bin Lin, Shragowitz, E., 1992. Bounds on net delays for VLSI circuits. IEEE Transactions on Circuits and Systems—II 11(39), 815–824.

Youssef, H., Sait, S.M., Hussain, A., 2001. Adaptive bias simulated evolution algorithm for placement. IEEE International Symposium on Circuits and Systems, Sydney, Australia, May 2001, pp. 355–358.

Zadeh, L.A., 1973. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man and Cybernetics SMC-3 (1), 28–44.

Zitzler, E., 1999. Evolutionary optimization for multiobjective optimization: methods and applications. DTS Thesis, Swiss Federal Institute of Technology, Zurich, November 1999.