

Fuzzy Simulated Evolution Algorithm for VLSI Cell Placement

by

Ali Syed Hussain

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

December, 1998

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

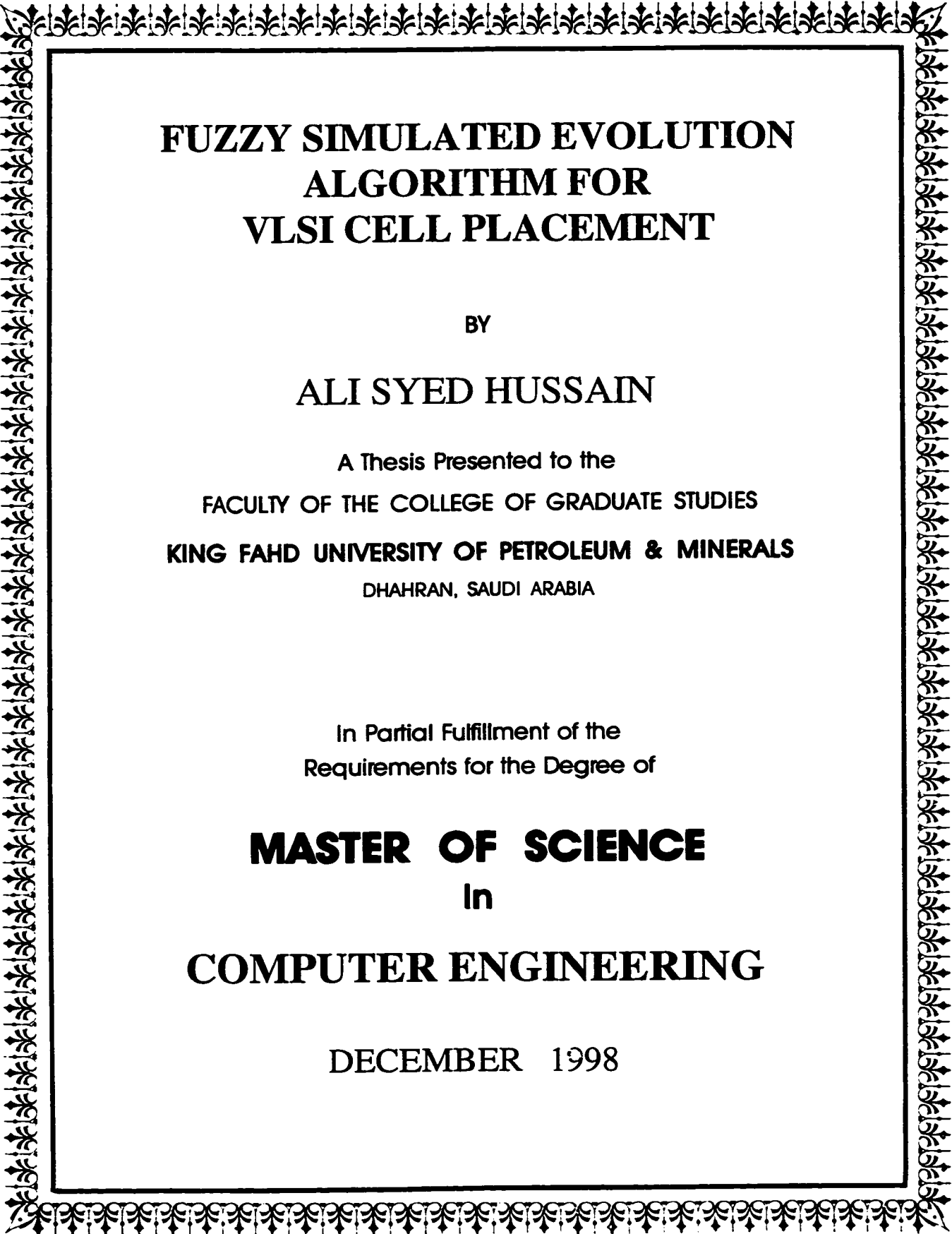
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

NOTE TO USERS

The original document received by UMI contains pages with indistinct print. Pages were microfilmed as received.

This reproduction is the best copy available

UMI



**FUZZY SIMULATED EVOLUTION
ALGORITHM FOR
VLSI CELL PLACEMENT**

BY

ALI SYED HUSSAIN

A Thesis Presented to the
FACULTY OF THE COLLEGE OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
COMPUTER ENGINEERING

DECEMBER 1998

UMI Number: 1393213

UMI Microform 1393213
Copyright 1999, by UMI Company. All rights reserved.

**This microform edition is protected against unauthorized
copying under Title 17, United States Code.**

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA
COLLEGE OF GRADUATE STUDIES

This thesis, written by

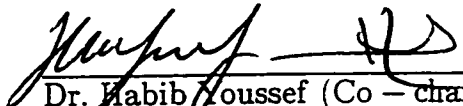
ALI, SYED HUSSAIN

under the direction of his Thesis Advisor and approved by his Thesis Committee,
has been presented to and accepted by the Dean of the College of Graduate Studies,
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

Thesis Committee


Dr. Sadiq M. Sait (Chairman)


Dr. Habib Joussef (Co - chairman)


Dr. Hasan Barada (Member)


Abdullah Almojel
Department Chairman


Dean, College of Graduate Studies

14/2/99
Date



Dedicated

to

my beloved parents

Acknowledgments

All praise be to Allah, *Subhanahu-wa-ta-Aaala*, for his limitless help and guidance. May Allah bestow peace on his prophet, Muhammad (pbuh), and his family. I acknowledge the support and facilities provided by King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

My family, especially my most loving Mummy and Daddy were constant source of motivation. Their love and support carried me through some difficult moments in my life. Their prayers, guidance and inspiration led to this accomplishment.

I would like to express my profound gratitude and appreciation to my thesis committee chairman Dr. Sadiq M. Sait and co-chairman Dr. Habib Youssef, for their guidance and patience throughout this thesis. It was because of them that the work at any point of time, never got stressful. Their continuous support, advice and encouragement can never be forgotten. Thanks are also due to my thesis committee member Dr. Hassan Barada for his interest, cooperation and constructive criticism.

I also wish to thank the Chairman of Computer Engineering Department, Dr. Abdullah Almojel and Dean of College of Computer Science and Engineering, Dr.

Khalid AlTawil for all the support they provided in order to achieve this work. Also, my thanks go to the faculty and staff members of Computer Engineering Department for their encouragement. I would like to thank all the research assistant community in KFUPM especially Salman, Naved, Tariq, Faisal and Mansoor for their moral support. Thanks are also due to Ahmad for helping me in arabic translation of thesis abstract.

Contents

Acknowledgments	ii
List of Tables	ix
List of Figures	x
Abstract (English)	xiii
Abstract (Arabic)	xiv
1 Introduction	1
2 Preliminaries	6
2.1 Introduction	6
2.2 Problem Definition	7
2.3 Standard Cell Layout	7
2.4 Cost Function	8
2.4.1 Wirelength Estimation of VLSI Layout	9

2.4.2	Layout Area	10
2.4.3	Circuit Delay	11
2.5	Net Delay Bounds	13
2.6	Technology Details	16
2.7	Simulated Evolution (SE) Algorithm	17
2.7.1	Description of SE Algorithm	17
2.8	Fuzzy Logic	22
2.8.1	Fuzzy Set Theory (FST)	22
2.8.2	Fuzzy reasoning	25
2.9	Conclusion	27
3	Literature Review	28
3.1	Introduction	28
3.2	General Placement Techniques	29
3.2.1	Constructive Placement	29
3.2.2	Iterative Placement Schemes	30
3.3	Fuzzy Logic Based Placement Schemes	36
3.4	Applications of SE algorithms	38
3.5	Conclusion	41
4	Fuzzy Logic Based Simulated Evolution Algorithm For VLSI Cell Placement	42

4.1	Introduction	42
4.2	Proposed Scheme and Implementation Details	44
4.2.1	Initial Solution	44
4.2.2	Fuzzy Goal-based Cost Measure	45
4.2.3	Proposed Evaluation Scheme	49
4.2.4	Selection	51
4.2.5	Allocation	51
4.2.6	Stopping Criterion	57
4.3	Experiments and Results	58
4.3.1	Effect of the Size of the Fuzzy Window on FSE_FA	61
4.3.2	Comparison of SE, FSE_WA and FSE_FA	62
4.3.3	Comparing FSE_WA and FSE_FE	67
4.4	Conclusion	75
5	Effect of Selection Bias on Algorithm Efficiency	77
5.1	Effects of Bias on Quality of Solution	80
5.2	Normalized Goodness Values	84
5.3	Variable Bias: Function of Average Cell Goodness ($B = \mathcal{F}(G)$)	88
5.4	Comparison of Bias Schemes	89
5.5	Fuzzy Allocation and Variable Bias SE algorithm (FSE_VB)	95
5.6	Conclusion	97

6 Conclusion	99
6.1 Future Research	103
BIBLIOGRAPHY	105

List of Tables

2.1	Technology parameters used in this research.	16
4.1	Classification of our SE implementations	59
4.2	Parameter values for different stages of the SE algorithm	60
4.3	The characteristics of circuits and layouts used in our experiments. (<i>LH</i> = layout heights in micron, <i>Avg. RCH</i> = average routing channel height in micron, O_1 = optimum wire length in micron, O_2 = Sum of the switching delay of the longest path in <i>nsec</i> , O_3 = optimum row length of the layout in micron.)	60
4.4	Results of FSE_WA for different weights for circuit c499	60
4.5	Results of FSE_FA for different sizes of <i>fuzzy window</i> (<i>w</i>). Where wire length (<i>L</i>) and layout width (<i>W</i>) are in micron. The circuit delay (<i>D</i>) and execution time of the algorithm (<i>T</i>) are in <i>ns</i> and minutes respectively.	63
4.6	Best layout found by SE, FSE_WA and FSE_FA.	65

4.7	Best layout found by FSE_FE and FSE_WA.	75
5.1	Effect of bias on quality of solution generated and execution time of the SE algorithm. The wire length cost of the solution (L) is in micron and the execution time (T) is in minutes.	83
5.2	Comparison of solution quality and execution times of best fixed bias and normalized goodness SE algorithm. The wire length cost of the solution (L) is in micron and the execution time (T) is in minutes. B is best fixed bias value.	86
5.3	Comparison of solution quality and execution times of best fixed bias and variable bias ($B = \mathcal{F}(G)$) SE algorithms. The wire length cost of the solution (L) is in micron and the execution time (T) is in minutes. B is best fixed bias value.	90
5.4	Best layout found by FSE_VB and FSE_WA. The wire length cost (L) and layout width (W) are in micron. The circuit delay cost (D) is in ns and execution time of the algorithm (T) is in minutes.	96

List of Figures

2.1	Layout of a standard cell placement.	8
2.2	Steiner tree approximation for estimation of interconnect length. The length of the tree is distance of the bisecting line (W) and depth is $\sum_{i=1}^n h_i$	10
2.3	Structure of the simulated evolution algorithm.	19
2.4	Sorted individual best fit placement.	22
2.5	Membership function for a fuzzy set A.	23
4.1	Range of acceptable solution set.	46
4.2	Membership function <i>within acceptable range</i> . By lowering the goal g_i to g_i^* the preference for objective “ i ” has been increased.	47
4.3	Membership functions used in fuzzy evaluation stage. (a) function for near optimum wire length (μ_1^e) (b) function for near net delay bound (μ_2^e).	51

4.4	Membership functions for three fuzzy variables used in <i>Fuzzy Allocation Scheme</i> of the SE algorithm.	56
4.5	Wire length costs of solutions against iteration count: (a) length of the current solution (after every 20 iterations) (b) length of the best solution	68
4.6	Circuit delay of solutions: (a) delay of the current solution (after every 20 iterations) (b) delay of the best solution.	69
4.7	Layout width against iteration count: (a) width of the current solution (after every 20 iterations) (b) width of the best solution.	70
4.8	Overall cost of the solutions represented as membership values of solutions in fuzzy set <i>acceptable solution</i> : (a) current membership values (after every 20 iterations) (b) best membership values.	71
4.9	Average cell goodness against iteration count: (a) cell goodness of the current solution (after every 20 iterations) (b) goodness of the best solution.	72
4.10	Cardinality of selection sets of FSE_FA, SE and FSE_WA.	73
5.1	The effect of bias on the Simulated Evolution algorithm. (a) quality of solution generated by SE (b) execution time of the algorithm.	81
5.2	Cardinality of selection set against against frequency of solutions for different fixed bias SE algorithm.	82

5.3	Comparison of best fixed bias and normalized goodness SE algorithm.	
	(a) size of the selection set (b) solution quality.	87
5.4	Comparison of different bias schemes (a) current wire length cost (b)	
	best wire length cost	91
5.5	Cardinality of selection set against frequency of solutions for different	
	bias schemes for the SE algorithm.	93
5.6	Comparison of different bias SE algorithms (a) average cell goodness	
	(b) best average cell goodness	94

THESIS ABSTRACT

Name: ALI, SYED HUSSAIN
Title: FUZZY SIMULATED EVOLUTION ALGORITHM
FOR VLSI CELL PLACEMENT
Major Field: COMPUTER ENGINEERING
Date of Degree: DECEMBER 1998

Simulated Evolution (SE) is an iterative heuristic to generate near optimal solutions to NP-Hard problems. VLSI cell placement is an NP-Hard problem with many conflicting objectives namely, wire length, circuit delay and area. The best solution for such a problem is the one which scores lowest with respect to all objectives. However, such a solution most likely does not exist. In order to identify the best solution generated by the Simulated Evolution algorithm, we propose a novel approach of fuzzy goal-based cost measure. This approach overcomes the problems related to the controversial weighted sum approach. It also allows easy incorporation of user preferences for different objectives. We have also proposed fuzzification of allocation and evaluation stages of the SE algorithm. The allocation scheme tries to minimize multiple objectives and adds controlled randomness as opposed to original deterministic allocation schemes. Experiments with benchmark tests demonstrate a noticeable improvement in solution quality. The fuzzy evaluation scheme combines wire length and net delay bounds for computation of individual cell goodnesses. It results in improvements in circuit delay but slight increase in the wire length. We have also proposed a variable bias scheme in place of fixed bias in the SE algorithm. The fixed bias scheme requires several trial runs to determine best value. In contrast, the variable bias is a function of average cell goodness, automatically adjusts its value and saves the SE algorithm from trial runs. The results of the variable bias scheme are comparable to the best fixed bias scheme.

MASTER OF SCIENCE DEGREE

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
Dhahran, Saudi Arabia

December 1998

خلاصة الرسالة

اسم الطالب الكامل: سيد حسين علي

عنوان الدراسة: خوارزم النشوء المحاكى الغائم لوضع الخلايا في الدوائر المتكاملة عالية الكثافة جدا

التخصص: هندسة الحاسب الآلي

تاريخ الشهادة: ديسمبر 1998

النشوء المحاكى (Simulated Evolution) عبارة عن مستكشف تسلسلي يستخدم لإيجاد حلول قريبة من الحل الأمثل للمشاكل التي لا يمكن حلها بكثيرات الحدود (NP-Hard Problems). وضع الخلايا في الدوائر المتكاملة عالية الكثافة جدا مشكلة لا يمكن تمثيلها بكثيره حدود ولها أهداف متعارضة ألا وهي طول التوصيلات، زمن التنفيذ والمساحة الكلية. أفضل الحلول لمشكلة كهذه هو الذي يحقق أقل القيم في جميع الأهداف إلا أن حلا كهذا لا يوجد غالبا. لتعريف الحل الأمثل الذي يقدمه خوارزم النشوء المحاكى، نقدم طريقة جديدة لحساب تكلفة الحل بطريقة غامضة معتمدة على الهدف. تغلب هذه الطريقة على المشاكل المتعلقة بطريقة المجموع الموزون. إضافة إلى ذلك فإن هذه الطريقة تسمح باعتبار أفضليات المستخدم لمختلف الأهداف. كما نقدم طريقة لتغميض مرحلة التوزيع ومرحلة التقييم في الخوارزم المذكور. طريقة التوزيع تحاول تقليل قيم الأهداف المتعارضة وإضافة عشوائية محكمة خلافا للطريقة الأصلية ذات التوزيع المحدد. التجارب على دوائر الاختبار تثبت تطورا ملحوظا في نوعية الحل. طريقة التقييم الغامض تجمع طول التوصيلات وزمنها لحساب جودة الخلايا متفرقة. ينتج عن ذلك تحسن في زمن التنفيذ وزيادة بسطة في طول التوصيلات. إضافة إلى ذلك، قدمنا طريقة تحفيز متغير بدلا من التحفيز الثابت في الخوارزم الأصلي. التحفيز الثابت يتطلب تجارب عديدة لتحديد القيمة الأنسب. في المقابل، التحفيز المتغير دالة تعتمد على متوسط جودة الخلية، تضبط قيمتها تلقائيا لتوفر التجارب. نتائج التحفيز المتغير قريبة من أفضل نتائج التحفيز الثابت.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران، المملكة العربية السعودية

التاريخ: ديسمبر 1998

Chapter 1

Introduction

In Very Large Scale Integration (VLSI) circuit design, *Placement* is the process of arranging circuit blocks on a layout. In *standard cell* design, where circuit blocks are of fixed height and variable widths, placement consists of determining optimum positions of all blocks on the layout to satisfy a number of objectives [1, 2]. The simplest version of the placement problem is the optimization of wire length for one dimensional placement. Even this simplified version is NP-Hard [3, 4]. More complex versions of this problem are characterized by multiple objectives and constraints such as interconnect timing delay, area, wire length etc. It is not possible to enumerate all combinations and come up with the best solution. Therefore we have to use some intelligent methods known as “heuristics”, to get near optimal solutions in reasonable amount of time. These heuristics fall into two categories: *constructive* and *iterative*.

Constructive heuristics produce a complete solution by making deterministic moves. Some examples of constructive schemes are force directed placement, numerical optimization, min-cut placement etc. These schemes are reviewed in [1, 2]. Constructive techniques are fast but fall short of generating good layouts. It is due to the fact that they always have local view of the search space [5].

Iterative heuristics attempt to improve a complete solution by making controlled stochastic moves [5]. Generally, we can classify iterative schemes into two subclasses. Schemes which always accept good solutions like local search, force directed interchange etc., and schemes which can accept bad solutions probabilistically like simulated annealing [6], genetic algorithms [7], simulated evolution [8] and stochastic evolution [37]. Probabilistic iterative schemes generally outperform constructive and greedy schemes because their hill climbing property (i.e., accepting bad solutions) saves them from getting trapped in local optima [5].

Iterative heuristics have been used for the VLSI cell placement. The use of genetic algorithm (GA) for placement is proposed in [9, 10, 11, 12, 13]. Similarly use of simulated annealing (SA) [6] for VLSI cell placement is discussed and reported in [1, 2, 14, 15, 16]. There are some concerns about the execution time of these schemes [17] and premature convergence of GA [18]. In order to overcome these problems, Kling and Banerjee proposed the *Simulated Evolution* (SE) heuristic [19], which combines iterative improvement and constructive perturbation. It saves itself from getting trapped in local optima by using stochastic selection of design compo-

nents for perturbation. The advantage of this heuristic is reduced execution time than simulated annealing and genetic algorithm [8].

For multiobjective optimization, fuzzy logic provides an easy way of expressing expert human knowledge in decision making process of any iterative heuristic. The advantage of fuzzy logic over classical crisp logic is that it establishes an approximate truth value of propositions in accordance with the rules designed by the expert, while in crisp logic the proposition will be either true or false. Furthermore, it provides a rigorous algebra for dealing with imprecise information.

Objectives of Research

In this thesis we present a Fuzzy Simulated Evolution Algorithm for multiobjective optimization of VLSI Standard Cell Placement. In our scheme we minimize three cost parameters of the placement layout: interconnection wire length, delay and layout width. We propose a novel way of identifying the best solution generated by the SE algorithm. This scheme uses a *fuzzy goal-based cost measure* which combines multiple cost parameters into a scalar cost measure. Apart from proposing this fuzzy cost measure, we have also investigated the fuzzification of other stages of the algorithm. The SE algorithm consists of three distinct steps: **evaluation**, **selection** and **allocation**. We propose fuzzification of **allocation** and **evaluation** stages of the simulated evolution algorithm. We propose a “fuzzy controlled stochastic allocation” instead of the previously purely constructive sorted individual best

fit allocation strategy. The experiments show that the proposed fuzzy allocation scheme based SE algorithm (FSE_FA) results in an overall improved solution quality compared to a weighted average allocation based algorithm (FSE_WA) as well as wire length based single objective SE algorithm (SE) as originally proposed by Kling and Banerjee [8]. Our proposed fuzzy evaluation scheme combines wire length and net delay bounds for evaluating placement of a circuit block in a location. The experiments indicate that for bigger circuits the proposed evaluation scheme based SE algorithm (FSE_FE) results in a reduction in circuit delay compared to the wire length only evaluation scheme.

Selection *bias* is a parameter of the SE algorithm. We have investigated the effects of this parameter on the simulated evolution algorithm. We propose three dynamic bias variants. The results of our modifications are compared with fixed bias [8] and normalized goodness measure [20].

Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 gives preliminary information helpful in understanding the rest of the thesis. This chapter formally defines the VLSI cell placement problem, describes the standard cell layout style and placement problem, computation of cost function and contains a review of fuzzy logic. It also reviews the classical simulated evolution algorithm.

In Chapter 3 we review related literature. This chapter covers VLSI cell place-

ment schemes, fuzzy logic based placement schemes and different studies which use SE in computer aided design problems.

In Chapter 4 we propose fuzzy goal-based cost measure. Furthermore, fuzzy allocation and evaluation schemes for the simulated evolution algorithm for VLSI cell placement are also proposed. Details of our implementations of these schemes are given. We compare and contrast these schemes with multiobjective weighted average SE algorithm and single objective SE implementation similar to the one proposed by Kling and Banerjee [8].

In Chapter 5 we investigate the effect of selection bias on simulated evolution algorithm. In this chapter we have proposed variable bias concept. Our proposals allow a bias which is a function of the problem instance. We also compare results of proposed scheme with the existing schemes like fixed bias [8] and normalized goodness [20]. The thesis ends with conclusion and future work in Chapter 6.

Chapter 2

Preliminaries

2.1 Introduction

Necessary problem specific information is included in this chapter. This information is needed to understand concepts, terminology and related work described in the subsequent chapters. We formally define the VLSI cell placement problem in Section 2.2. We will also describe the Standard Cell Placement (Section 2.3) and cost computation functions for such a placement (Section 2.4). We have used net delay bounds in our implementation of the simulated evolution algorithm. The transformation of the path constraints into net delay bounds is given in Section 2.5. Details of the environment used in our experiments are given in Section 2.6.

This research is based on the use of the simulated evolution algorithm. Thus, a quick review of this heuristic is given in Section 2.7. A primer on fuzzy logic is

given in Section 2.8. The chapter ends with a conclusion in Section 2.9.

2.2 Problem Definition

Given a set of modules $M = \{m_1, m_2, \dots, m_n\}$ and a set of signals $S = \{S_1, S_2, \dots, S_k\}$, we associate with each module $m_i \in M$ a set of signals S_m , where $S_m \subseteq S$. Similarly with each signal $S_i \in S$ we associate a set of modules M_{s_i} , where $M_{s_i} = \{m_j | s_i \in S_{m_j}\}$. M_{s_i} is said to be a signal *net*. We are also given a set of slots or locations $L = \{L_1, L_2, \dots, L_p\}$, where $p \geq n$. The placement problem is to assign each $m_i \in M$ to a unique location L_j such that some objectives are to be optimized [1]. Generally, minimization of interconnect wire length has been widely used as the objective of VLSI placement. However, advancement in technology has resulted in reduction in gate switching delay making the interconnect delay a prominent factor in overall circuit delay [21]. Reduction of interconnect delay and layout area along with wire length are important objectives in the placement stage of the VLSI physical design automation process.

2.3 Standard Cell Layout

In standard cell design all the cells are constrained to have the same height, while width of the cell is variable and depends upon its complexity [1]. Cells are placed in horizontal rows and the cell rows are separated by horizontal routing channels.

In order to connect cells within a row or cells from two different rows, channels are used for running interconnect wires. Connecting cells from two non-adjacent rows, requires feed through cells in intermediate rows. The feed through cells allow running vertical wires from cell rows. Figure 2.1 shows block diagram of a standard cell layout.

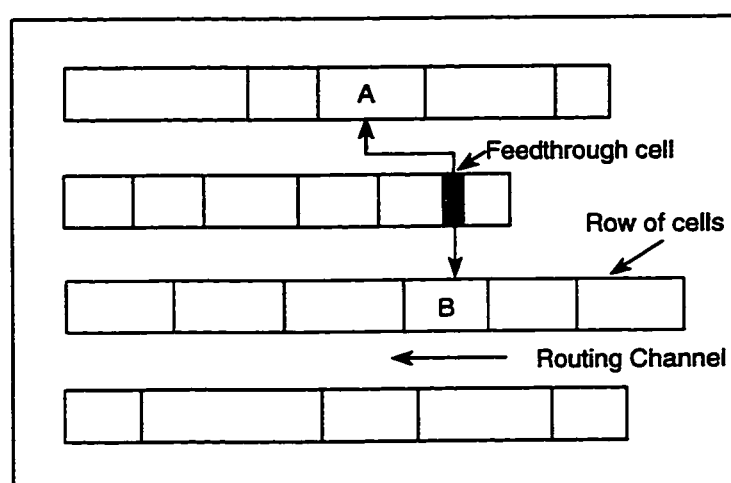


Figure 2.1: Layout of a standard cell placement.

2.4 Cost Function

In this section we will review the estimation models for finding the cost of a VLSI placement layout. As mentioned earlier, in this work we attempt to minimize three quantities: wire length, interconnect delay, and area. The following text describes the estimation schemes used in our implementations for these three parameters.

2.4.1 Wirelength Estimation of VLSI Layout

Usually, layout placement generators optimize the total interconnect wire length. Reduction in wire length results in cost reduction, reduced chip area in standard cell design and improvement in circuit delay. The cost of a solution due to wire length is determined by adding the wire length estimates for all the nets in the circuit.

Different estimation schemes for net length have been used in the placement algorithms. For example (a) semi-perimeter method, (b) complete graph, (c) minimum chain, (d) source to sink connection, (e) minimum spanning tree, and (f) minimum Steiner tree are used to estimate net length [2]. A Steiner tree is the shortest route for connecting a set of pins [1]. However, determination of minimum steiner tree is known to be NP-Complete [1]. Therefore, in this work we approximate the steiner tree using following technique.

Steiner tree approximation is a quick and accurate way to estimate net length. Figure 2.2 illustrates this approximation method. For each net, a bounding rectangle is determined. The rectangle is partitioned into two parts depending upon the smaller dimension. If the width (height) of the rectangle is more than the other dimension, then the rectangle is partitioned into two parts by a horizontal (vertical) line passing through the center of the net. The *length* of the tree will be the distance of the bisecting line and the summation of all the projections from the center of

net blocks to the bisecting line. In Figure 2.2 rectangle is partitioned horizontally because the width of the rectangle is more than the height. This scheme provides a quick and good estimation of the wire length.

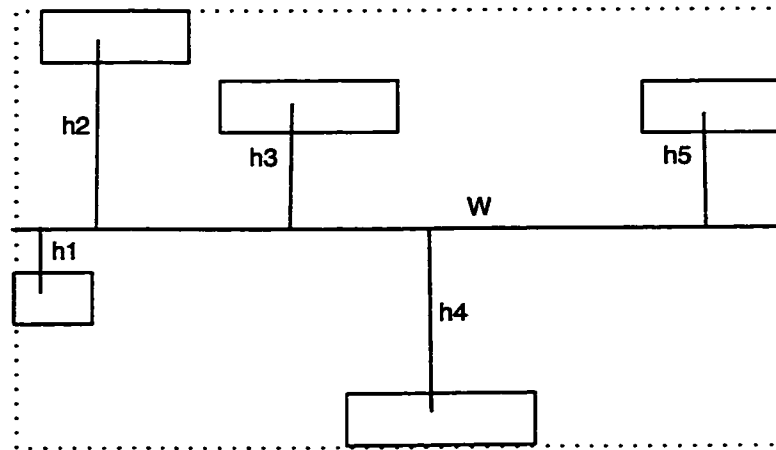


Figure 2.2: Steiner tree approximation for estimation of interconnect length. The length of the tree is distance of the bisecting line (W) and depth is $\sum_{i=1}^n h_i$.

2.4.2 Layout Area

In standard cell design, circuit blocks have fixed height and variable widths. Blocks are placed in rows with routing channels separating two adjacent rows. The overall layout area is represented by the rectangle, which bounds these rows and routing channels. In our work, the heights of routing channels are initially estimated and assumed to be fixed. This leaves only width of layout that can be minimized resulting in reduced area. If cells are placed in such a way that all the rows have approximately the same width, then the placement is considered to be good with respect to area.

2.4.3 Circuit Delay

From the perspective of a layout generator, the VLSI circuit consists of circuit blocks and interconnection between the blocks. A **net** is an equipotential interconnect of pins on different cells. One circuit block can be part of many nets. A **path** is an alternating sequence of blocks and nets from *source* net to *sink* net. A **source** net can be either input pad or output of memory element whereas the **sink** net can be output pad or input of memory element. Both source and sink of a path are controlled by the same clock.

The overall performance of the VLSI circuit depends upon how fast it can process signals i.e., its clock speed. The propagation delay of signals of a VLSI circuit consists of two elements, switching delay and interconnect delay. Due to improvement in technology, gate switching delays have considerably decreased. The effect of this decrease is that the wiring delay factor becomes prominent in the overall chip delay. The layout generator is concerned with reducing the interconnect delay.

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

$$T_\pi = \sum_{i=1}^{k-1} (CD_{v_i} + ID_{v_i}) \quad (2.1)$$

where CD_{v_i} is the switching delay of the cell driving net v_i and ID_{v_i} is the interconnect delay of net v_i . The overall circuit delay is equal to the T_{π_c} where path π_c

is the longest path in the layout (the most critical path).

For a layout generator the switching delay is constant, while the interconnect delay depends on the distance between the members of the net. Using the (lumped) *RC delay model*, this delay is due to interconnect resistance and capacitance. Since the effect of interconnect resistance is very negligible, we can ignore it. The delay due to interconnect capacitance for net v_i is given by Equation 2.2.

$$ID_{v_i} = LF_{v_i} \times C_{v_i} \quad (2.2)$$

where LF_{v_i} is load factor of the driving block and C_{v_i} is the interconnect capacitance. The load factor is independent of the layout. The interconnect capacitance depends upon the distance between the blocks of the respective net. If two layers of metal are used for routing the nets then following set of equations show how the interconnect capacitance of a net v_i is computed.

$$C_{v_i} = C_{v_i}^a + C_{v_i}^f \quad (2.3)$$

$$C_{v_i}^a = (C_{m_1} \times L_1^i + C_{m_2} \times L_2^i) \times \omega \quad (2.4)$$

$$C_{v_i}^f = 2 \times ((\omega + L_1^i) \times C_{f_1} + (\omega + L_2^i) \times C_{f_2}) \quad (2.5)$$

where

C_{v_i} = Interconnect capacitance of net v_i .

$C_{v_i}^a$ = Area Capacitance of net v_i .

$C_{v_i}^f$ = Fringe Capacitance of net v_i .

C_{m_1} = Plate capacitance per unit area of metal 1.

C_{m_2} = Plate capacitance per unit area of metal 2.

C_{f_1} = Fringe capacitance per unit length of perimeter of metal 1.

C_{f_2} = Fringe capacitance per unit length of perimeter of metal 2.

ω = Width of metal 1 or metal 2.

L_1^i = Length of metal 1 used in connecting net v_i .

L_2^i = Length of metal 2 used in connecting net v_i .

In Equations 2.3 to 2.5, ω , C_{m_1} , C_{m_2} , C_{f_1} , C_{f_2} are technology dependent parameters.

Section 2.6 gives the characteristics of the technology used in this work.

2.5 Net Delay Bounds

As we have seen in Section 2.4.3, the delay of a VLSI circuit depends on the sum of interconnect delay and switching delay of the longest path. A circuit will be free from long-path problems if, for all paths $\pi \in \Pi$, $SLACK_\pi \geq 0$. A negative slack indicates that circuit will not be able to propagate the signal through path within the required time. The $SLACK_\pi$ is computed as follows.

$$SLACK_{\pi} = LRAT_{\pi} - T_{\pi} \quad (2.6)$$

where $LRAT_{\pi}$ and T_{π} are the “latest required arrival time” and the “actual arrival time” of the signal to the sink of path π . T_{π} is computed as given in Equation 2.1 while $LRAT_{\pi}$ depends on the clock period and some technology dependent parameters like clock skew and setup time at the path sink.

Path slacks represent the upper constraints on the delay of respective paths. These constraints can be transformed to the net constraints. However, for a layout to be free from long path problems, net constraints need not be satisfied in their entirety. We have used net bounds obtained by *minimax-PERT* algorithm [22], that transforms path constraints into net bounds. These bounds are used in fuzzy logic based selection scheme in the simulated evolution algorithm (proposed in Section 4.2.3). Following is a brief description of the problem of determining net constraints and the approach used by *minimax-PERT* algorithm.

The key idea of finding net constraints is that a net v_i belongs to several paths. Thus, the net constraints for this net must satisfy the longest path traversing this net. Let π be a path in circuit represented as graph $G = (V, E)$ and V_{π} is the set of vertices it traverses. The delay along path π is given by the Equation 2.1. The switching delay is independent of the layout. Therefore, for the final layout to be free from long path problems, the interconnect delays must satisfy the following

constraint:

$$\sum_{v_i \in V_\pi} ID_{v_i} \leq LRAT_\pi - \sum_{v \in V_\pi} CD_{v_i} \quad \forall \pi \in \Pi \quad (2.7)$$

where Π is the set of all paths in circuit graph G . The net constraints must satisfy the inequality 2.7. Using *minimax* approach, the timing bound for each net v_i on path π is assigned as follows.

$$u_{v_i}^\pi = Slack_\pi \times \frac{w_{v_i}}{\sum_{u \in V_\pi} w_u} \quad (2.8)$$

Where

$$\begin{aligned} w_{v_i} &= \text{the weight of net } v_i \\ &= LF_{v_i} \times AcL_{v_i} \end{aligned}$$

Let Π_v be the set of paths going through net v_i . A consistent bound will be minimum of $u_{v_i}^\pi$ bounds for all Π_v paths. We can represent this bound as follows:

$$u_{v_i}^* = \min_{\pi \in \Pi_v} u_{v_i}^\pi \quad (2.9)$$

The problem of finding $u_{v_i}^*$ requires enumerating all the paths in the design. This is an NP-Hard [23]. *Minimax-PERT* algorithm approximates these bounds after enumerating a polynomial number of paths using a PERT-like trace of the graph. Interested reader can consult [22] for description of this algorithm.

Metal Type	w (micron)	Sheet Resistance Ω / \square	Area Capacitance $10^{-4} \text{ pF} / \mu^2$	Fringe Capacitance $10^{-4} \text{ pF} / \mu$
Metal 1	4.0	0.06	0.26	0.82
Metal2	4.0	0.033	0.15	0.85

Table 2.1: Technology parameters used in this research.

2.6 Technology Details

In our work we have adopted standard cell design using a 2μ p-well CMOS technology. The parameters for cells like timing characteristics and dimensions are given in [24]. Furthermore, it is assumed that two layers of routing are used. Metal 1 is used for routing of horizontal tracks and metal 2 is used for routing of vertical tracks. The values of capacitances and resistances of these layers are available in [25]. The values used in our computation are given in Table 2.1. Since the sheet resistance values are very negligible compared to capacitance values, we ignore the effect of resistance on delay.

In order to make interconnect delay as a prominent factor in the overall circuit delay, we have reduced the switching delay (CD_{v_i}) in Equation 2.1 by a factor of 4. It is done to emulate a high speed advanced technology in which switching delays are major portion in the overall circuit delay.

2.7 Simulated Evolution (SE) Algorithm

Simulated Evolution (SE) is a general iterative heuristic proposed in [19]. 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 [18]. This scheme combines the iterative improvement and constructive perturbation and saves itself from getting trapped in local minima by using stochastic approach. It iteratively operates a sequence of evaluation, selection and allocation (perturbation) on one solution. Using a time homogeneous irreducible Markov chain, Kling and Banerjee [26] showed that algorithm converges in the limit to a global minimum with probability one.

In this section we will review simulated evolution (SE) algorithm in detail, as given in [8].

2.7.1 Description of SE Algorithm

The Simulated Evolution is a general heuristic for solving a variety of combinatorial optimization problems. The SE proceeds as follows. It starts with a randomly generated valid initial solution. The main loop of the algorithm consists of three steps: **evaluation, selection and allocation**. These steps are carried out repetitively in a main loop until some stopping condition is satisfied. Other than these three steps, some input parameters for the algorithm are initialized in an earlier step known as

initialization. Following is the description of these procedures. The pseudo code of the algorithm is given in Figure 2.3.

Initialization

Initialization step is carried out only once. It consists of selecting a starting valid solution for the problem under consideration. This solution can be generated randomly or the output of any constructive heuristic. The other important parameters which can be initialized in this step are a stopping condition and selection bias (B). Different stopping conditions can be used. For example, the stopping condition can be a fixed number of iterations of the main loop or a function of improvement in the solution cost. Selection bias is used to compensate errors made in the estimation of the optimum cost used in the computation of goodness (see the evaluation stage below). The selection bias controls the magnitude of the perturbation of current solution. It effects the overall execution speed of the algorithm and the quality of the final solution. A carefully selected value of the bias results in a good quality solution.

Evaluation

In this step, each individual member of the solution is evaluated on the basis of problem constraints and objectives. This evaluation is represented by the **goodness** for each element of the current solution. The goodness of an element of the design

Algorithm *Simulated_Evolution*($B, \Phi_{initial}, StoppingCondition$)

NOTATION

B = Bias Value.

Φ = Complete Solution.

e_i = Individual cell in Φ .

O_i = Lower bound on cost of i^{th} cell.

C_i = Current cost of i^{th} cell in Φ . g_i = Goodness of i^{th} cell in Φ .

S = Queue to store the selected cells.

ALLOCATE(e_i, Φ_i)=Function to allocate e_i in partial solution Φ_i

Repeat

EVALUATION: **ForEach** $e_i \in \Phi$ **DO**
begin

$$g_i = \frac{O_i}{C_i}$$

end

SELECTION: **ForEach** $e_i \in \Phi$ **DO**
begin

IF *Random* > *Min*($g_i + B, 1$)

THEN begin

$S = S \cup e_i$; Remove e_i from Φ .

end

end

Sort the elements of S

ALLOCATION: **ForEach** $e_i \in S$ **DO**
begin

ALLOCATE(e_i, Φ_i)

end

Until *Stopping Condition is satisfied*

Return Best solution.

End (*Simulated_Evolution*)

Figure 2.3: Structure of the simulated evolution algorithm.

is defined as follows.

$$g_i = \frac{o_i}{a_i} \quad (2.10)$$

where o_i is the optimum value on the cost of element i and a_i is the actual cost estimate for this element in the current design. The goodness represents a measure of how near each element is to its optimum position. As is obvious from Equation 2.10, the goodness of an element is between 0 and 1. A value of goodness near 1 means that element i is near its optimum location.

Selection

The goodness is used to probabilistically select elements in the **selection** step. Elements with low goodness have a higher probability of getting selected for reposition. Selection bias (B) is used to compensate errors made in estimation of the optimum cost. Its objective is to inflate or deflate the goodness of elements. A high positive value of bias decreases the probability of selection or vice versa. A carefully tuned bias value results in good solution quality and reduced execution time [26]. The selection step results in a partial solution of only unselected elements, while selected elements are saved in a queue for allocation.

Allocation

The purpose of the allocation is to perturb the current solution in such a way that the selected elements are assigned to better design positions. The allocation

function affects the quality of solution as well as convergence of the search. Different constructive allocation schemes are proposed in [8]. One such scheme is **sorted individual best fit**, where all the selected elements are sorted in descending order in a queue with respect to their connectivity with the partial solution. The sorted elements are removed one at a time and *trial* moves are carried out for all the available empty positions at that time. The element is *finally* placed in a position where maximum reduction in cost for the partial solution is achieved. This process is continued until the selected queue is empty. Figure 2.4 illustrates this scheme for VLSI cell placement. There are four selected VLSI cells and as many empty locations in the layout. The head of line cell (cell 1) is tried in all four locations and finally placed in slot number 3 because the reduction in cost is maximum. Then second cell is tried on remaining slots and placed in slot 4. This process is repeated for all the remaining selected cells. The overall complexity of this algorithm is $O(s^2)$ where s is the number of selected elements. Other more elaborate allocation schemes are **weighted bipartite matching allocation** and **branch-and-bound search allocation** [8]. However, these schemes are more complex allocation strategies than “sorted individual best fit”, but result in comparable solution quality [8]. In this work we have used a fuzzy sorted individual best fit allocation scheme.

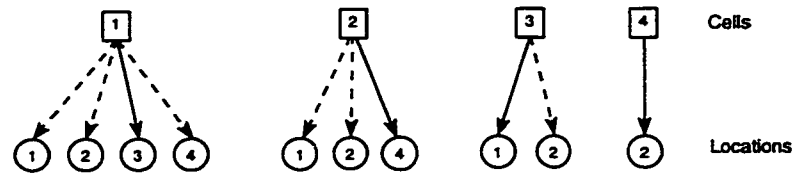


Figure 2.4: Sorted individual best fit placement.

2.8 Fuzzy Logic

2.8.1 Fuzzy Set Theory (FST)

A crisp set is normally defined as a collection of elements or objects $x \in X$ that can be finite, countable or uncountable. Each single element can either belong to a set or not. However, in real life situations objects do not have crisp [1 or 0] 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 set 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 of the set [27, 28]. A fuzzy set A of a universe of discourse X is defined as $A = \{(x, \mu_A(x)) \mid \text{all } x \in X\}$, where $\mu_A(x)$ is a membership function of $x \in X$ being an element in A [29]. Figure 2.5 shows one example of a membership function.

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

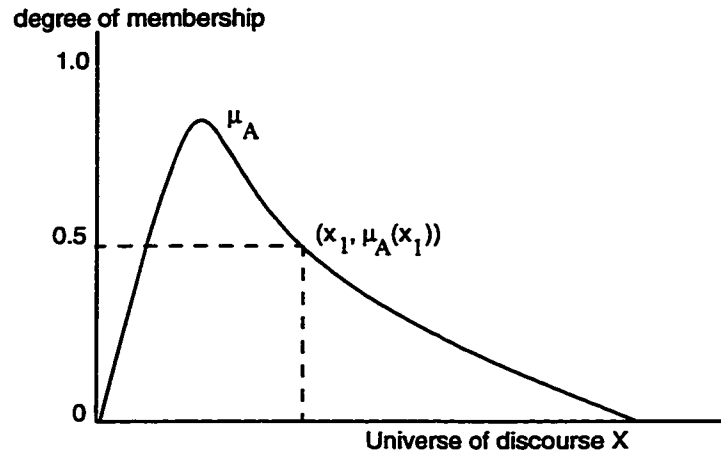


Figure 2.5: Membership function for a fuzzy set A.

(denoted as \oplus). While fuzzy intersection operators are known as **t-norm** (denoted as $*$). Some examples of **s-norm** operators are given below, (where A and B are fuzzy sets of universe of discourse X) [27].

1. Maximum. $[\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)]]$.
2. Algebraic sum. $[\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)]$.
3. Bounded sum. $[\mu_{A \cup B}(x) = \min(1, \mu_A(x) + \mu_B(x))]$.
4. Drastic sum. $[\mu_{A \cup B}(x) = \mu_A(x)$ if $\mu_B(x) = 0$, $\mu_B(x)$ if $\mu_A(x) = 0$, 1 if $\mu_A(x), \mu_B(x) > 0]$.

An **s-norm** operator satisfies commutativity, monotonicity, associativity and $\mu_{A \cup 0} = \mu_A$ properties. Following are some examples of fuzzy intersection operators known as **t-norm**.

1. Minimum. $[\mu_A \cap_B(x) = \min[\mu_A(x), \mu_B(x)]]$.
2. Algebraic product. $[\mu_A \cap_B(x) = \mu_A(x)\mu_B(x)]$.
3. Bounded product. $[\mu_A \cap_B(x) = \max(0, \mu_A(x) + \mu_B(x) - 1)]$.
4. Drastic product. $[\mu_A \cap_B(x) = \mu_A(x)$ if $\mu_B(x) = 1$, $\mu_B(x)$ if $\mu_A(x) = 1$, 0 if $\mu_A(x), \mu_B(x) < 1]$.

Like s-norms, t-norms also satisfy commutativity, monotonicity, associativity and $\mu_A \cap_1 = \mu_A$ properties. Additionally, the membership function for fuzzy complementation operator is defined as.

$$\mu_{\bar{B}}(x) = 1 - \mu_B(x)$$

Ordered Weighted Averaging Operator

Generally, 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 [30]. This operator allows easy adjustment of the degree of “anding” and “oring” embedded in the aggregation. According to [30], “orlike” and “andlike” OWA for two fuzzy sets A and B are implemented as given

in Equations 2.11 and 2.12 respectively.

$$\mu_{A \cup B}(x) = \beta \times \max(\mu_A, \mu_B) + (1 - \beta) \times \frac{1}{2}(\mu_A + \mu_B) \quad (2.11)$$

$$\mu_{A \cap B}(x) = \beta \times \min(\mu_A, \mu_B) + (1 - \beta) \times \frac{1}{2}(\mu_A + \mu_B) \quad (2.12)$$

β is a constant parameter in the range $[0,1]$. It represents the degree to which OWA operator resembles a pure “or” or pure “and” respectively.

2.8.2 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 [31] introduced the concept of linguistic variable. A **linguistic variable** is a variable whose values are words or sentences in natural or artificial language [32]. The set of values a linguistic variable can take is called a **term set**. This set is constructed by means of primary terms and by placing modifiers known as **hedges** like “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. Formally, a linguistic variable comprises five elements [33].

1. The variable name.
2. The primary term set.
3. The Universe of discourse U .
4. A set of syntactical rules that allows composition of the primary terms and hedges to generate the term set.
5. A set of semantic rules that assigns each element in the term set a linguistic meaning.

For example, *wire length* can be used as linguistic variable for VLSI cell placement problem. According to the syntactical rule, the set of linguistic values of *wire length* may be defined as *very small wire length*, *small wire length*, *somewhat small wire length*, *large wire length*. The universe of discourse for linguistic variable is possible range of wire length for some designs. For instance, universe of discourse for linguistic variable *wire length* can be the interval $[100000 \mu, 150000 \mu]$. The set of semantic rules define fuzzy sets for each linguistic value. A linguistic value is characterized by its corresponding fuzzy set. The membership in fuzzy set is controlled by membership functions like Figure 2.5. It shows the designer's knowledge of the problem.

2.9 Conclusion

In this chapter we have seen relevant material necessary to understand subsequent chapters. We have defined the VLSI cell placement problem and reviewed the cost functions with respect to interconnect wire length, layout area and circuit delay. We have also seen the transformation of path delay constraints into net constraints. We will later use these constraints in the SE based placement scheme to improve the circuit delay. The technology dependent constants and parameters are also given .

In this chapter we have also reviewed the simulated evolution heuristic and fuzzy logic. This heuristic is a powerful stochastic iterative heuristic for general combinatorial optimization problems. It is based on the analogy between optimization and evolutionary processes. Underlying principle of this heuristic is that evolutionary system be subjected to the evaluation process such that it stochastically discards inferior parts and retains superior parts of the system. The constructive perturbation in allocation stage ensures that the algorithm will converge to a sub-optimal solution. On the other hand, fuzzy logic provides a convenient algebra of combining conflicting objectives and expert human knowledge.

Chapter 3

Literature Review

3.1 Introduction

The VLSI cell placement problem is an NP-hard problem [34]. The simplified version of this problem is the optimization of wire length for one dimensional placement of cells. Even this simplified problem of optimally placing n cells requires enumeration of $n!$ options. This is impractical even for circuits having few hundred cells. Because of exponential complexity of this problem, clever techniques known as “heuristics” are used to generate near optimal solutions in reasonable amount of time.

In this chapter, we review some of the important studies carried out in this area. Section 3.2 reviews placement techniques which use crisp knowledge for decision making. Fuzzy logic allows a convenient way of combining human expert knowledge in decision making. Therefore, several studies have used fuzzy logic based decision

making in placement techniques. These studies are reported in Section 3.3. As this research is based on Simulated Evolution Algorithm (described in Section 2.7), therefore we have reviewed the applications of SE on computer-aided design (CAD) problems. This review is reported in Section 3.4.

3.2 General Placement Techniques

3.2.1 Constructive Placement

Constructive placement schemes build a placement in a piecewise manner. These schemes assign cells one by one cells are placed. The algorithm starts with an initial solution known as seed. Then other cells are selected (one at a time in order of their connectivity to the placed modules) and placed at a vacant location near to the already placed cells [2]. This process continues until all cells are placed. Such schemes are very fast but fall short of good placement. This is due to the fact that these schemes make decision about placing a cell in a location on the basis of partial placement only. Secondly, once a decision is made, no matter how bad it is, there is no mechanism to reverse it. Therefore, these schemes may remain trapped in local minima.

These algorithms are generally used to generate initial solutions for iterative algorithms. The reason of using these techniques is their fast execution time. Other examples of constructive placement techniques are numerical optimization, place-

ment by partitioning and force directed scheme. These heuristics are surveyed in [2].

3.2.2 Iterative Placement Schemes

In contrast to the constructive techniques, iterative schemes require enormous computational time to generate good placement. Iterative techniques start with an initial solution and repeatedly modify the solution in each iteration until no more improvement occurs. The modification in solution is intended to reduce the cost of the solution. Iterative schemes can be further classified on the basis of whether they can accept bad solutions probabilistically or not. Those iterative algorithms which allow acceptance of bad moves probabilistically fall in the sub-category of “stochastic iterative algorithms”. This property is called “hill climbing property”. It saves algorithms from getting trapped in local minima. However, it is required that the acceptance of bad moves be controlled to avoid random traversal of search space. Examples of such probabilistic iterative schemes are simulated annealing (SA), genetic algorithm (GA) and simulated evolution (SE).

There are iterative heuristics which do not allow acceptance of bad moves. These schemes are known as “deterministic iterative techniques”. Examples of such techniques are min-cut, force directed interchange and local search etc. These schemes are discussed in [1, 2].

Stochastic Iterative Schemes

Three stochastic iterative algorithms have been successfully applied on the VLSI cell placement problem. These algorithms are simulated annealing (SA) [6], genetic algorithm (GA) [35], and simulated evolution (SE) [19]. Following is a brief review of these implementations.

Simulated Annealing

Simulated Annealing is a popular combinatorial optimization algorithm proposed by Kirkpatrick et. al. [6]. It is derived from the analogy of the physical annealing process of metals. SA works on a single solution. The neighborhood state of the solution is generated by randomly selecting modules and interchanging their positions. All good moves are accepted. However, bad moves are stochastically accepted. The acceptance probability of bad moves is controlled by a cooling schedule. In early stages of the search, number of bad moves are accepted with high probability. However, as search progresses, cooling temperature decreases, so does the probability of accepting bad moves. In the last part of the search, SA behaves as a greedy algorithm, accepting only the good moves. For details of simulated annealing algorithm interested readers are referred to [5].

Sechen [36] used SA to solve the VLSI placement problem. Their implementation is known as Timberwolf. The algorithm performs placement and routing in three stages. The first stage is concerned with the cell placement in order to reduce

interconnect wire length. A neighbor function produces new states by making a random selection from three perturb functions i) move a single cell to a new location ii) swap two cells iii) mirror a cell about the x-axis. The cost function contains total estimated wire length, penalty for cell overlap and length of the row exceeding (or falling short of) the expected length. The annealing process is started at a very high temperature. Initially the temperature falls very rapidly. In intermediate range the temperature is reduced slowly and in the final stage it again falls rapidly. Most of the improvement in the quality of solution is achieved in the intermediate temperature range. The remaining two stages of the algorithm are concerned with routing.

In order to reduce the execution time of SA, Mallela et. al. [15] proposed a clustering based SA for the standard cell placement. In the first stage, using SA they place clusters of cells formed on the basis of interconnections. In the second stage, the clusters are broken into individual cells and then cells are placed using SA. This hierarchal SA reduces the problem size without compromising on the quality of solution.

Simulated annealing is a very successful heuristic for combinatorial optimization. However, it suffers from two major drawbacks. It requires careful tuning of its control parameters and it also needs excessive computation time [37]. Tao et. al., in [17] have shown that SA takes more execution time due to non aggressive neighborhood search. During each iteration of SA, algorithm chooses a random

neighboring solution which is not always most profitable one. The real cost improves mostly in a narrow time range.

Genetic Algorithm

Genetic Algorithm (GA) is another powerful and popular optimization algorithm which works by emulating the natural process of evolution as a means of progressing toward the optimum. It was invented by Holland [35]. Unlike SA which works repeatedly on single solution, GA works on a set of solutions in parallel. The set of solutions are known as a population. Each solution is represented by a string of symbols known as chromosome. Four genetic operators namely *selection*, *crossover*, *mutation* and *inversion* are repeatedly applied on a collection of solutions to generate new offsprings. Extensive literature is available on genetic algorithms and their use in combinatorial optimization problems. Interested reader is referred to [5, 7].

Cohon et. al. [38] used GA for VLSI cell placement. They proposed an algorithm with the name "Genie". In their algorithm, initial population is a mix of individuals generated by two methods. They have used randomly generated individuals along with individuals generated by a greedy constructive heuristic. If the entire population consists of individuals generated using a greedy constructive heuristics then genetic algorithm converges prematurely to a local minimum. The presence of random layouts in initial population provides diversification. Fitness of individual solution is determined on the basis of interconnect wire length only. The

fitness function does not account for possible cell overlaps or variation in row length. Different types of parent choice functions, crossover operators and individual survival functions were tested. They compared results of GA with that of SA for five circuits. GA was able to generate same quality solution in two cases while in other three tests quality was up to 7% worse than SA.

Shahookar et. al., [2, 9] also implemented GA for the VLSI placement and compared the performance of their algorithm using different crossover operators, like partially mapped (PMX), cycle and order crossover. They used an initial population of randomly generated individuals. The fitness function of an individual was equal to the reciprocal of its total interconnect wire length estimate. Inversion or mutation genetic operators were also used with different probabilities. After crossover the fitness of offspring is evaluated and the population for the next generation is selected from the population of offsprings and parents. According to their study, PMX or cycle crossover performed best, while order crossover was the worst. The quality of final solution generated by GASP was comparable with the results of Timberwolf [2, 9] .

Khalid [39] investigated the use of GA for multi-objective optimization of the VLSI cell placement. Not only wire length but also circuit delay and layout area was optimized. The crossover operators were geared to transfer information about satisfied paths (sequence of nets from input to output which transfer signal in same clock pulse) to next generation. This resulted in timing improvement of up to 17%

against a min-cut placer.

Holt et. al. [12] proposed GEEP, a GA based low power layout system. GEEP reduces interconnect capacitance by an average of 20% over recursive min-cut area optimizing placement. They included the low population diversity in GA to achieve fast execution time.

Genetic algorithm requires extensive computational time and high memory. Furthermore, when population has individuals with equal fitness then it is likely that GA will converge prematurely [18].

Simulated Evolution

Simulated Evolution (SE) is also an iterative heuristic proposed by Kling and Banerjee [8]. This algorithm is inspired by the simulated annealing. However, instead of carrying out a random selection and stochastic perturbation as in the case of SA, simulated evolution uses stochastic selection based on individual goodness and constructive perturbation. Due to these characteristics SE can generate near optimal solutions in reduced execution time than SA. Kling and Banerjee applied SE for VLSI placement problem and achieved solution quality comparable to SA in considerably less execution time. Simulated Evolution algorithm is described in Section 2.7.

3.3 Fuzzy Logic Based Placement Schemes

VLSI cell placement problem contains multiple conflicting objectives. As an example, if we are optimizing wire length and layout area, it is possible that one particular move in an iterative algorithm reduces area but increases wire length. It is difficult to decide whether this type of move is good or bad. Use of crisp logic for such situations will not be able to represent the expert opinion. Therefore, fuzzy logic provides an alternative reasoning mechanism which can satisfy conflicting objectives by using expert human knowledge [32]. Fuzzy logic has been used in VLSI design automation problems. Most of the schemes proposed in the literature use fuzzy logic in constructive heuristics. Following is a brief review of some of the schemes.

Lin et. al., in [32] discuss their implementation of fuzzy rule based sea of gates placement mechanism. It is a constructive cluster growth heuristic in which decisions regarding which cell to select, and where to place it, are carried out using fuzzy logic. Fuzzy logic was used to facilitate multi-objective decision making among conflicting placement objectives like reducing wire length, layout area and circuit delay. Similarly in [40], constructive placement scheme is described in which fuzzy logic is used to construct a connection matrix to represent the 'true' degree of connectivity between cells. The extension of this work is reported in [41]. The authors have proposed a hierarchical placement strategy. In the first stage, a connection matrix is formed which is used in the fuzzy similarity process for clustering. In the

second stage, final placement is carried out by using *fuzzy c-means* clustering and linear ordering process to partition the modules. It is reported that this constructive algorithm has overall complexity of $O(n^{2.5})$ where n is the number of modules.

Ball et. al., in [42] describe two methods of constructive partitioning and placement. Both schemes are based on *fuzzy c-means* algorithm. The first method minimizes wire length by force directed relaxation and fast clustering technique. The second method uses fuzzy similarity relation for specifying similarities between the cells. The set of modules are partitioned into subsets based on their characteristic features such as connectivity, size, aspect-ratios, power consumption etc.

Mackey et. al., in [43], describe a performance driven macro cell placement algorithm. Their technique optimizes inter-device delay by using a quad-partitioning algorithm with tabu search and fuzzy cost function. The algorithm repeatedly quad-partitions the layout, places the cells in regions and optimizes the regions by doing cell movements and cell swaps across the regions. This partitioning and optimization continues until the number of cells in each region is small enough to be easily placed. Three values contribute to fuzzy cost of a solution: total path length cut by region boundaries, maximum path length cut by region boundaries and total number of paths cut by region boundaries. It is claimed that fuzzy based cost function improves the placement by shortening the longest path through the circuit.

Yukimatsu et. al., in [44] give the description of hierarchical fuzzy expert system for placement of parts on printed circuit board. Their scheme starts with dividing the

set of parts into several blocks. The presence of parts in the blocks is given by fuzzy values, which represent the degree of belonging (DOB). Each block is represented by one major component. In the first stage of placement, these blocks are placed on the layout using simulated annealing, ignoring the area of blocks. Then placement areas are assigned taking into consideration the DOBs of the components. These areas can overlap. In the second stage parts are placed based on the rough placement according to their DOBs to the blocks. Each position of the device is optimized using constructive heuristics. Results show that the final solution is feasible.

Hakeem [45], used fuzzy logic based genetic algorithm (GA) for the floorplanning problem. Floorplanning problem is also a hard problem of VLSI physical design stage. It is carried out before placement and finds locations of macro blocks on a layout. His proposed fuzzy GA scheme used the fuzzy decision making approach for cost computation. He reported comparable results to a weighted sum approach.

3.4 Applications of SE algorithms

Simulated Evolution (SE) was proposed as a general iterative heuristic. It has been used in optimization of number of problems related to computer-aided design (CAD). Following is a brief review of some studies which use SE for combinatorial optimization problems in CAD.

Ly et. al. [46] have used SE for high level synthesis problem. A high level

synthesis tool maps the abstract behavior specifications (like algorithm) represented by control/data flow graphs (CDFG) into data path circuit and a finite state machine (FSM) description. The FSM specifies the controller required to sequence the data path through control steps. The synthesis task can be divided into two sub-tasks namely scheduling and allocation. The scheduling is concerned with assignment of each node of CDFG to a control step such that all conflicts due to data/control flow dependencies are resolved. The allocation task is concerned with assigning hardware cells to CDFG nodes and edges so that resulting circuit can implement the algorithm. In [46] authors have formulated scheduling and allocation steps of high level synthesis as optimum assignment problem. SE algorithm is applied for both steps. Scheduling is carried out first and then allocation is determined so that hardware (or area) is reduced. The results of experiments show that SE based synthesis requires less run times to generate designs as well as quality of final generated solution is comparable to existing schemes.

Asynchronous pipeline design requires a well-designed interconnection circuit. This design results in high speed performance by eliminating the clock skew problems of synchronous system. Kuw et. al. have reported study of partitioning and scheduling of asynchronous pipelines [47]. The scheduling of operations into stages is carried out by using simulated evolution algorithm. For two test cases, their implementation was able to achieve sub-optimal solutions.

In [48] SE algorithm is used to solve gate-matrix layout problem. The gate-

matrix layout problem is solved as a one-dimensional transistor gates placement problem. After generating a constructive solution, the algorithm repeatedly works on it by removing ill fit designs based on their individual goodness and re-allocating them to empty slots. As reported in [48] authors obtained good results.

Routing of VLSI circuits is another NP-complete problem [1]. There are several studies in recent years which use SE in one form or the other to solve this problem. For example Wang et. al. have used SE for rip-up and re-route steps in order to resolve the channel capacity violations of initial routing step [49]. In order to decrease path delays of circuit, their implementation works on critical path based routing rather than net based routing. Similarly Yuh et. al. have used SE based performance and routability driven router algorithm for field programmable gate arrays (FPGAs) [20]. In the first stage of the algorithm, nets are routed sequentially according to their criticality. In the second stage, violating nets/paths with respect to routing resource and timing constraints are ripped up and re-routed. For this stage SE heuristic is used to find near optimal solution. Experimental results show that their implementation gives better results with respect to reduction in delay and improvement in routability over many existing routing algorithm. Similarly in [50, 51, 52] SE has been used for VLSI routing problem.

3.5 Conclusion

In this chapter we reviewed VLSI placement schemes. One can broadly classify these schemes on the basis of how they build the solution. Constructive schemes use a piece wise placement strategy in which individual cells are placed one by one until complete layout is ready. Iterative schemes work on a complete solution and repeatedly try to improve the quality of layout. If the selection of a move is greedy i.e., always accepting good moves, then such a scheme is sub-classified as deterministic iterative heuristic. Constructive schemes and deterministic iterative heuristics are likely to be trapped in local minima because of their narrow view and limited traversal of search space.

Stochastic iterative heuristics are iterative heuristics which allow probabilistic acceptance of bad moves. Although these heuristics are very time consuming, they guarantee generation of sub-optimal solution. One characteristic of these schemes is that they carry out a controlled walk in the search space. Examples of such heuristics are simulated annealing, genetic algorithm, simulated evolution etc. Detail reviews of these and other algorithms are found in [5].

In this chapter we have also reviewed applications of simulated evolution on number of CAD problems.

Chapter 4

Fuzzy Logic Based Simulated Evolution Algorithm For VLSI Cell Placement

4.1 Introduction

In [8], Simulated Evolution (SE) algorithm was applied for VLSI cell placement problem. The objective function in that study was to reduce interconnect wire length. However, with the advancements in technology, gate sizes have decreased resulting in low switching delays and driving strength. This made the propagation delays of interconnects relatively more dominant. For standard cell design, where modules have variable widths, placement is also concerned with minimizing layout area. This

makes VLSI placement a multiobjective optimization problem. In our scheme, we minimize three cost parameters of the layout: interconnection wire length, circuit delay, and layout width.

During placement process, all desirable objectives can only be imprecisely estimated. Fuzzy logic provides a rigorous algebra for dealing with imprecise information. Furthermore, it is a convenient method of combining conflicting objectives and expert human knowledge. From the pseudo code of the SE algorithm given in Figure 2.3, it is clear that there are two stages of the algorithm, which can be modeled to include multiple objectives. These stages are **evaluation** and **allocation**. We have used fuzzy logic based reasoning in these two stages. The proposed schemes are compared with weighted sum allocation based SE (FSE_WA) and single objective original SE (SE) as proposed by Kling and Banerjee [8]. The FSE_WA uses a wire length based evaluation stage and multiobjective weighted average allocation stage.

In order to identify the best solution for a multiobjective optimization problem, one has to tradeoff between various objectives. In this work we adopt a goal directed search approach, where the best placement is the one that satisfies as much as possible a user specified vector of fuzzy goals. This is achieved by using a *fuzzy goal-based cost measure*.

In this chapter we describe our proposed schemes, details of implementation and results of experiments. This chapter is organized as follows. In Section 4.2, we give our proposed schemes and step by step details of implementation of different

stages of the SE algorithm. After describing the generation of initial solution in Section 4.2.1, we describe our approach of *fuzzy goal-based cost computation* in Section 4.2.2. In Section 4.2.3 we describe the proposed *fuzzy evaluation scheme* as well as classical evaluation scheme. After covering the selection process in Section 4.2.4, we describe the allocation stage of the SE algorithm in Section 4.2.5. In this section, we describe our proposed *fuzzy controlled stochastic sorted individual best fit allocation* strategy. A weighted sum sorted individual best fit allocation scheme is also described. Section 4.3 reports and discusses the results of different experiments. The chapter ends with conclusion (Section 4.4).

4.2 Proposed Scheme and Implementation Details

In this section we give our proposals of fuzzification of different stages of the SE algorithm. This description is combined with details of implementation of the SE algorithm for VLSI cell placement problem.

4.2.1 Initial Solution

Initial layout is generated randomly in which fixed number of cell rows are considered, with no limitation on the width of the cell row. The intermediate routing channels have fixed heights equal to the average height of routing channels generated

by a min-cut based placer.

4.2.2 Fuzzy Goal-based Cost Measure

VLSI placement is a multiobjective combinatorial optimization problem. A placement is evaluated against several objective criteria such as, wire length, delay and layout width. The best placement is the one which scores lowest with respect to all objectives. However, such a solution most likely does not exist. One usually has to tradeoff these various objectives. In such a case, the concept of optimum is not clear. Traditional approach consists of combining all objectives in a weighted sum cost function, and the placement with lowest weighted sum is reported as the best solution [21, 53]. This approach is at best controversial. Furthermore, the individual placement objectives are very imprecise. In this work we adopt a goal directed search approach, where the best placement is the one that satisfies as much as possible a user specified vector of fuzzy goals. This scheme is inspired by the work reported in [54, 55].

Let there be Γ solutions generated by the SE algorithm. Assume that we are optimizing a p -valued cost vector given by $C(x) = (C_1(x), C_2(x), \dots, C_p(x))$ where $x \in \Gamma$. Assume that a vector $O = (O_1, O_2, \dots, O_p)$ gives lower bound estimates on individual objectives such that $O_i \leq C_i(x) \forall i, \forall x \in \Gamma$. These are lower bounds on each objective which usually are not achievable in practice. Further, assume that there is a user specified goal vector $G = (g_1, g_2, \dots, g_p)$ which indicates the relative

acceptable limits for each objective. It means that x will be an acceptable solution if $C_i(x) \leq g_i \times O_i$ where $\forall i, g_i \geq 1.0$. For a two dimensional optimization problem, Figure 4.1 shows the region of acceptable solutions.

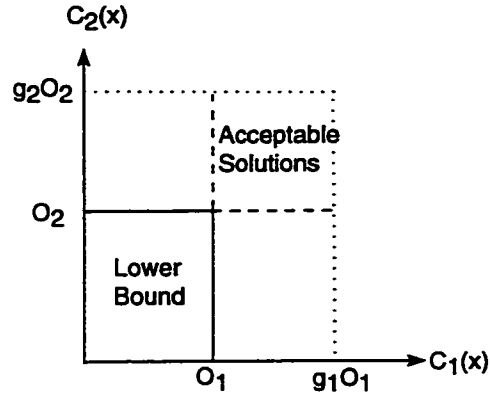


Figure 4.1: Range of acceptable solution set.

In our proposed scheme, the *acceptable solution* set is a fuzzy set. For VLSI cell placement problem of minimizing three parameters, we propose the following rule to determine membership in the fuzzy set *acceptable solution*. This rule is implemented by Equation 4.1.

Rule 4.1: **IF** a solution is *within acceptable wire length*

AND within acceptable circuit delay AND within acceptable width

THEN it is an *acceptable solution*.

$$\mu^c(x) = \beta^c \times \min(\mu_1^c(x), \mu_2^c(x), \mu_3^c(x)) + (1 - \beta^c) \times \frac{1}{3} \sum_{i=1}^3 \mu_i^c(x) \quad (4.1)$$

where $\mu^c(x)$ is the membership value for solution x in the fuzzy set *acceptable solution*. While μ_i^c for $i = \{1, 2, 3\}$ represents the membership values of solution x

in the fuzzy sets *within acceptable wire length*, *within acceptable circuit delay* and *within acceptable width* respectively. The solution which results in the maximum value for Equation 4.1 is reported as the best solution found by the SE algorithm. In Equation 4.1 β^c is an averaging constant in the range [0,1]. The superscript c stands for “cost” and it is used to differentiate similar variables used in the allocation and evaluation stages. Membership function for a general criterion i is shown in Figure 4.2. Mathematically we can write it as follows.

$$\mu_i^c(x) = \begin{cases} 1 & \text{if } \frac{C_i(x)}{O_i} \leq 1.0 \\ 1.0 - \frac{C_i(x) - O_i}{(g_i - 1.0) \times O_i} & \text{if } 1.0 < \frac{C_i(x)}{O_i} < g_i \\ 0 & \text{if } \frac{C_i(x)}{O_i} \geq g_i \end{cases} \quad (4.2)$$

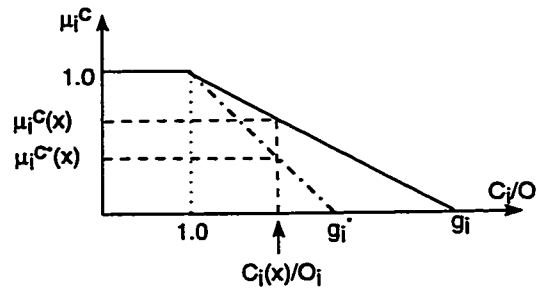


Figure 4.2: Membership function *within acceptable range*. By lowering the goal g_i to g_i^* the preference for objective “ i ” has been increased.

In this work, the lower bounds on objectives are computed at initialization by the placement program. We have used the following expressions to compute the

lower bounds for wire length (O_1), circuit delay (O_2), and layout width (O_3).

$$O_1 = \sum_{i=1}^n L_{v_i}^* \quad \forall v_i \in \{v_1, v_2, \dots, v_n\} \quad (4.3)$$

$$O_2 = \sum_{j=1}^k CD_{v_j} \quad \forall \{v_1, v_2, \dots, v_k\} \text{ nets in path } \pi \quad (4.4)$$

$$O_3 = \left[\frac{\sum_{i=1}^n W_{v_i}}{\# \text{ of cell rows in layout}} \right] \quad (4.5)$$

where

O_1 = Lower bound for interconnect wire length.

O_2 = Lower bound for circuit delay.

O_3 = Lower bound for layout width.

n = Number of nets in the layout.

$L_{v_i}^*$ = Optimum wire length for net v_i .

CD_{v_i} = Switching delay for the cell driving net v_i .

k = Number of nets in the longest path π .

π = Longest path in the layout with respect to switching delay.

W_{v_i} = Width of individual cell driving the net v_i .

The goal vector $G = (g_1, g_2, g_3)$ specifies relative acceptable ranges for wire length, delay and width criteria. Since it is relative information with respect to the lower bounds, it is easy for the user to enter these limits. User preferences can be

easily expressed in the goal vector (G). For example, by decreasing the goal value g_i to g_i^* in Figure 4.2, the subsequent membership value $\mu_i^*(x)$ for objective i will decrease. This might dictate the acceptance or rejection of solutions.

4.2.3 Proposed Evaluation Scheme

In this stage of the algorithm, individual cell **goodness** is computed. The *classical goodness* measure proposed in [8] computes the cell goodness on the basis of one parameter i.e., wire length. For such a measure, goodness of cell c_i which is a part of $\{v_1, v_2, \dots, v_k\}$ nets is computed as follows.

$$g_{c_i} = \frac{1}{k} \sum_{j=1}^k \min \left(\frac{L_{v_j}^*}{L_{v_j}}, 1.0 \right) \quad (4.6)$$

where $L_{v_j}^*$ and L_{v_j} are respectively optimum and actual wire length of net v_j . The $L_{v_j}^*$ is computed by placing the cells of a net next to each other on the layout surface and then estimating the wire length.

With the classical goodness measure, it is possible that a cell satisfying wire length might be violating net delay bound. This will negatively affect the overall circuit delay. Thus, inclusion of a penalty for not satisfying the corresponding net bound in goodness measure will increase the chances of selection of such cells. It is expected that during re-allocation stage, the selected cell will satisfy corresponding net delay bound. This will result in an improvement in the circuit delay. In order to

achieve this objective, we modified the goodness computation method. In the modified *fuzzy evaluation scheme*, satisfaction of the optimum wire length and the net delay bounds are combined using fuzzy logic. The following rule and the subsequent equation are used for this purpose.

Rule 4.2: **IF** a solution is *near optimum wire length*

AND *near net delay bound*

THEN it has *high goodness*.

$$g_{c_i} = \mu^e(x) = \beta^e \times \min(\mu_1^e(x), \mu_2^e(x)) + (1 - \beta^e) \times \frac{1}{2} \sum_{i=1}^2 \mu_i^e(x) \quad (4.7)$$

The superscript e stands for **evaluation** and it is used to distinguish similar notation in other fuzzy rules. In Equation 4.7, $\mu^e(x)$ is the fuzzy set of *high goodness*, g_{c_i} is goodness value and β^e is a constant. The $\mu_1^e(x)$ and $\mu_2^e(x)$ represent the memberships in the fuzzy sets *near optimum wire length* and *near net delay bound*. The membership functions for these sets are given in Figure 4.3. The base values are computed as follows. If a cell c_i drives a net identified by v_i which is a part of $\{v_1, v_2, \dots, v_k\}$ nets, then base values $X_1^e(x)$ for the fuzzy set *near optimum wire length* and $X_2^e(x)$ for the fuzzy set *near net bound* are computed as given in Equations 4.8-4.9.

$$X_1^e = \frac{1}{k} \sum_{j=1}^k \min\left(\frac{L_{v_j}^*}{L_{v_j}}\right) \quad (4.8)$$

$$X_2^e = \frac{u_{v_i}^*}{ID_{v_i}} \quad (4.9)$$

where $u_{v_i}^*$ is the net constraint for net v_i . Net constraints are compute using minimax-PERT algorithm [22].

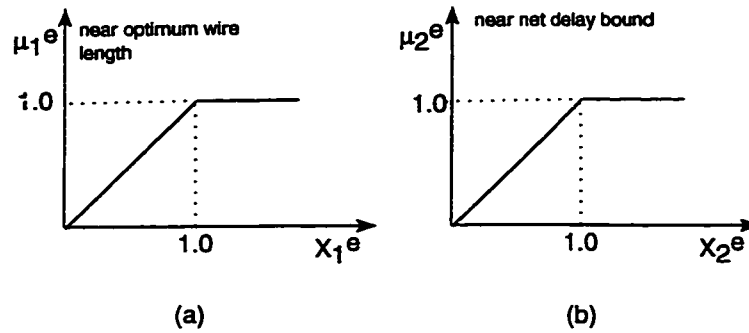


Figure 4.3: Membership functions used in fuzzy evaluation stage. (a) function for near optimum wire length (μ_1^e) (b) function for near net delay bound (μ_2^e).

4.2.4 Selection

In this stage of the algorithm, for each cell $c_i = \{c_1, c_2, \dots, c_n\}$ a random number in the range $[0,1]$ is generated and compared with $g_{c_i} + B$, where B is selection bias. If the generated random number is greater than the $g_{c_i} + B$ then cell c_i is selected for the allocation and removed from the layout. The location of cell c_i is marked as *empty*.

4.2.5 Allocation

During the **allocation** stage of the algorithm, the selected cells are repositioned on empty locations in such a way that they result in overall better solution. As described in Section 2.7, "sorted individual best fit" is an allocation scheme for the

SE algorithm. In this scheme, we identify the best location for head-of-line cell in selected queue which results in the maximum reduction in cost. The best location is removed from the list of empty locations. For single objective minimization, it is straightforward to identify such location. The original SE (SE) proposal [8] places the cell on a location which results in the maximum reduction in wire length cost. However, for multiple and conflicting objectives, the decision to identify the best location for selected cell will require many tradeoff. One solution to this problem is to use **weighted average** of multiple objectives. For each trial placement, reduction in cost due to individual objective is estimated, normalized and summed together using weights assigned for each objective. The selected cell is finally placed in a position which results in the highest value for the overall reduction in cost.

Assuming a cell c_i is temporarily placed in a location l during the m^{th} iteration of the SE algorithm. This cell is part of $\{v_1, \dots, v_k\}$ nets. Let r be the row number of the cell location l . The quality of this trial placement can be measured as follows.

$$\text{gain}_{c_i,l}^m = w_1^a \times \Delta L_{c_i,l} + w_2^a \times \Delta D_{c_i,l} + w_3^a \times \Delta W_{c_i,l} \quad ; \quad \sum_{h=1}^3 w_h^a = 1.0 \quad (4.10)$$

where $\Delta L_{c_i,l}$, $\Delta D_{c_i,l}$, and $\Delta W_{c_i,l}$ are respectively measure of gains in wire length, delay and width of the layout for the trial placement of cell c_i on location l . These

gains are computed as follows:

$$\Delta L_{c_i,l} = \frac{\sum_{j=1}^k (L_{v_j}^{m-1} - L_{v_j}^m)}{\sum_{j=1}^k L_{v_j}^{m-1}} \quad (4.11)$$

$$\Delta D_{c_i,l} = \frac{\sum_{j=1}^k (D_{v_j}^{m-1} - D_{v_j}^m)}{\sum_{j=1}^k D_{v_j}^{m-1}} \quad (4.12)$$

$$\Delta W_{c_i,l} = \frac{W_{opt} - W_r^m}{W_{opt}} \quad (4.13)$$

In Equation 4.11, $L_{v_j}^{m-1}$ and $L_{v_j}^m$ are respectively wire length estimates for net v_j in $m - 1^{st}$ and m^{th} iterations of the SE algorithm. Similarly $D_{v_j}^{m-1}$ and $D_{v_j}^m$ in Equation 4.12 are propagation delays for net v_j in $m - 1^{st}$ and m^{th} iterations respectively. The W_{opt} in Equation 4.13 represents the optimum row length (lower bound on maximum row length) for the layout. It is computed by adding the widths of all the cells and dividing it by the number of cell rows. The W_r^m is the row length of row r during the trial placement of cell c_i .

The above weight based sorted individual allocation scheme has two problems.

1. It is difficult to come up with appropriate weights for Equation 4.10.
2. It is possible that head-of-line cell will block optimum positions for remaining selected cells.

The use of proper fuzzy rules and membership functions can overcome the first problem. One solution to the second problem is to allow creation of empty space by shifting other cells. However, this will disturb many well placed cells resulting in reduction in overall quality of solution. We propose a **fuzzy allocation** scheme which is characterized by the following two properties.

1. It uses fuzzy rules and membership functions to combine multiple objectives.
2. It adds a *controlled randomness* in placing a cell on an empty location. In this scheme, it is possible that a cell is placed anywhere on a set of empty locations within a *fuzzy window*. The fuzzy window contains locations which result in near identical reductions in cost.

The logic behind the *controlled randomness* is that it decreases the chances of blocking an optimal location for the rest of the selected cells by the head-of-line cell. In the following text we describe the proposed *fuzzy allocation* scheme.

Fuzzy Allocation Scheme

Following the sorting of selected cells in descending order with respect to their connectivity with partial placement, the head-of-line cell is trial placed on all the available empty locations at that time and the membership of these locations in a fuzzy set of *good locations* is computed. The fuzzy subset of *good locations* falling within a *fuzzy window* is identified and is called *favorable locations*. The cell will

be randomly placed on any location within this fuzzy subset. Following is the description of how these sets are formed and the rules that govern them.

Good Locations

Assume E is the set of empty locations and S is the set of selected cells. A location $l \in E$ will be a member in the fuzzy set *good locations* for cell $c_i \in S$ with membership function $\mu_{c_i}^a(l)$. The determination of this membership function is carried out through the following rule and it evaluated by using the Equation 4.14

Rule 4.3: **IF** a location results in *small length AND reduced timing*

AND small layout width THEN it is a *good location*.

$$\mu_{c_i}^a(l) = \beta^a \times \min(\mu_1^a(l), \mu_2^a(l), \mu_3^a(l)) + (1 - \beta^a) \times \frac{1}{3} \sum_{i=1}^3 \mu_i^a(l) \quad (4.14)$$

where $\mu_{c_i}^a(l)$ is the fuzzy set of *good locations* and β^a is a constant parameter in the range $[0,1]$. The values $\mu_1^a(l)$, $\mu_2^a(l)$ and $\mu_3^a(l)$ represent the membership values of location l in the fuzzy sets *small length*, *reduced timing* and *small layout width* respectively.

The base values $X_1^a(l)$, $X_2^a(l)$ and $X_3^a(l)$ for corresponding membership functions $\mu_1^a(l)$, $\mu_2^a(l)$ and $\mu_3^a(l)$, are computed below using the notation of Equations 4.11-4.13.

$$X_1^a(l) = \frac{\sum_{j=1}^k L_{v_j}^m}{\sum_{j=1}^k L_{v_j}^{m-1}} \quad (4.15)$$

$$X_2^a(l) = \frac{\sum_{j=1}^k D_{v_j}^m}{\sum_{j=1}^k D_{v_j}^{m-1}} \quad (4.16)$$

$$X_3^a(l) = \frac{W_r^m}{W_{opt}^m} \quad (4.17)$$

After computing the base values, memberships in respective fuzzy sets are determined using the functions given in Figure 4.4.

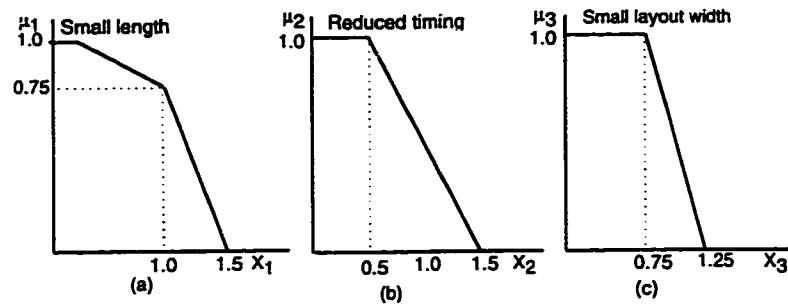


Figure 4.4: Membership functions for three fuzzy variables used in *Fuzzy Allocation Scheme* of the SE algorithm.

Favorable Locations

From the fuzzy set *good locations* a fuzzy subset of *favorable locations* is defined as follows. The upper and lower boundaries for the *favorable locations* are determined by a *fuzzy window*. For a cell $c_i \in S$, location $l \in E$ will fall within the *fuzzy window* if it satisfies the following inequality.

$$\max_{Ve \in E}(\mu_{c_i}^a(e)) \times \left\{ 1.0 - w \times \frac{\# \text{ of unplaced cells}}{\# \text{ of selected cells}} \right\} \leq \mu_{c_i}^a(l) \leq \max_{Ve \in E}(\mu_{c_i}^a(e)) \quad (4.18)$$

where E is the set of empty locations, S is the set of selected cells and w is a small positive value which determines the lower limit of $\mu_{c_i}^a(l)$ for presence in the *fuzzy window*. It controls the randomness in the fuzzy allocation scheme. All locations falling in the *fuzzy window* will be identified as *favorable locations*. The cell will be placed randomly in any of the locations within this set. The presence of the ratio of number of unplaced cells to selected cells will make sure that the size of *favorable locations* set will decrease as allocation progresses. Since the head-of-line cell is strongly connected with the partial layout, there are many locations with near identical gains. We can place the cell on any of these locations without adversely affecting the quality of solution. Thus, a bigger subset is identified for leading cells. The cells at the end of the selected queue are sparsely connected. For this reason, a narrow *favorable locations* set is used for them.

We experimented with different values for w in our proposed fuzzy allocation scheme. It is observed that for small values of w in the range (0.05, 0.1), the quality of solutions improves. However, as the value of w is increased the quality decreases due to increased randomness.

4.2.6 Stopping Criterion

In our experiments, we have used a fixed number of iterations as a stopping criterion. We experimented with different values of iterations and found that for all the available test circuits, the SE algorithm converges within 5000 iterations or less.

A major reduction in cost occurs in the first 2000 iterations while refinements are achieved in the rest of the iterations.

4.3 Experiments and Results

This section summarizes experimental study of the varieties of SE algorithms implemented in this work, namely

1. Wire length based simulated evolution algorithm implementation similar to one proposed by Kling and Banerjee [8]. This implementation is labeled as SE.
2. Fuzzy goal based simulated evolution algorithm using weighted sum allocation scheme. This algorithm is identified with code FSE_WA.
3. Fuzzy goal based simulated evolution algorithm using fuzzy controlled stochastic allocation scheme. This algorithm is identified as FSE_FA.
4. Fuzzy goal based simulated evolution algorithm using fuzzy evaluation scheme. The algorithm is labeled as FSE_FE.

The characteristics of above listed algorithm variations are summarized in Table 4.1. The parameter values for different stages of the SE algorithm are summarized in Table 4.2. For example, weighted sum allocation scheme in FSE_WA was tuned after several combinations of allocation averaging weights (w_1^a, w_2^a, w_3^a) were tested on one

circuit (c499). The best solution reported by the *fuzzy goal-based cost measure* for each weight combination is given in Table 4.4. The weight arrangement of row number 6 in Table 4.4 results in the maximum membership value for the fuzzy goal-based set: *acceptable solution* (Equation 4.2). As a result, the weights of $w_1^a = 0.6$, $w_2^a = 0.1$ and $w_3^a = 0.3$ are used in subsequent runs of FSE_WA for all circuits.

We tested our implementations of the simulated evolution algorithms on eight ISCAS-89 circuits. The characteristics of these circuits and layout parameters are listed in Table 4.3. The smallest circuit has 56 cells and the largest has 2243 cells. We are using 2μ p-well CMOS technology (Table 2.1) and place all the cells of circuit except input output pads. Following sets of experiments are carried out.

1. Understanding the effect of the size of the *fuzzy window* on FSE_FA.
2. Comparing SE, FSE_WA and FSE_FA.
3. Comparing FSE_WA and FSE_FE.

Algorithm	Evaluation	Allocation	Cost
SE	Wire length	Wire length	Wire length
FSE_WA	Wire length	Weighted sum (length,delay,width)	Fuzzy-goal based (length, delay, width)
FSE_FA	Wire length	Fuzzy (length,delay,width)	Fuzzy-goal based (length,delay,width)
FSE_FE	Fuzzy (length,net bounds)	Weighted sum (length,delay,width)	Fuzzy-goal based (length,delay,width)

Table 4.1: Classification of our SE implementations

Stage	Parameters
Fuzzy-goal based Cost	$\beta^c = 0.6, (g_1 = 2.0, g_2 = 3.0, g_3 = 1.1)$
Weighted Allocation	$(w_1^a = 0.6, w_2^a = 0.1, w_3^a = 0.3)$
Fuzzy Allocation	$\beta^a = 0.7, w = 0.1$
Fuzzy Evaluation	$\beta^e = 0.7$
Stopping Condition	Fixed 5000 iterations
Initial Solution	Random
Bias (B)	Fixed

Table 4.2: Parameter values for different stages of the SE algorithm

Circuit			Layout					
Name	# of Cells	# of IO	# of Rs	LH	Avg. RCH	O_1	O_2	O_3
highway	56	11	3	284	55.0	7156	3.91	512
fract	149	24	5	556	66.5	28207	7.97	784
c499	283	73	6	750	80.4	43583	8.17	1176
c532	395	43	7	703	49.5	64674	17.83	1152
c880	784	86	9	1034	64.0	123616	16.8	1824
c1355	1451	73	13	1557	66.9	273138	13.62	2304
struct	1952	64	15	2102	88.0	432096	13.54	3280
c3540	2243	72	17	2480	93.4	500157	23.26	3096

Table 4.3: The characteristics of circuits and layouts used in our experiments. (LH = layout heights in micron, $Avg. RCH$ = average routing channel height in micron, O_1 = optimum wire length in micron, O_2 = Sum of the switching delay of the longest path in $nsec$, O_3 = optimum row length of the layout in micron.)

Allocation Weights			Cost of best Layout		
w_1^a	w_2^a	w_3^a	L (μ)	D (ns)	W (μ)
0.2	0.1	0.7	64035	16.5	1200
0.3	0.2	0.5	58949	15.8	1208
0.3	0.1	0.6	61692	18.3	1200
0.4	0.1	0.5	59544	15.9	1200
0.4	0.2	0.4	60659	15.2	1200
0.6	0.1	0.3	59278	14.5	1200

Table 4.4: Results of FSE_WA for different weights for circuit c499

4.3.1 Effect of the Size of the Fuzzy Window on FSE_FA

In our proposed *fuzzy allocation scheme*, the presence in the fuzzy set *favorable locations* is bounded by Equation 4.18. The size of this window is controlled by a parameter w . For example if $w = 0.1$ and $\max(\mu_s^a(e)) = 0.7$ then the lower range for the fuzzy window will start from 0.63. It means that all the locations with membership functions in fuzzy set *good locations* in the range $[0.63 - 0.7]$ will be member of *favorable locations*. For higher values of w , the size of fuzzy window will increase or vice versa. In order to understand the effect of the size of this window on quality of solution, we executed FSE_FA algorithm on benchmark circuits with different values of w . The three cost parameters: length (L), delay (D), width (W) of the best solution found by FSE_FA and corresponding execution times (T) are given in Table 4.5. From this table, it is clear that by increasing the value of w , the algorithm spends more time. This is because of additional computation carried out to maintain the fuzzy set of *favorable locations*. For many circuits, the quality of final solution for small values of w is better than the solution generated by $w = 0$. This improvement is due to the fact that we probabilistically allow few best locations to remain empty for remaining cells in the selected queue. This allows some of the remaining cells to be better placed, resulting in improvement in the quality. At the same time, the window size for leading cells is small in the range $[0.05, 0.1]$. It makes sure that these cells will be placed in near identical quality positions with respect

to the best position. However, for high values of w , quality of the best reported solution decreases. This decrease is due to the size of the *fuzzy window*. For large *fuzzy window* allows the presence of bad locations in the set *favorable locations*. This will make the allocation more random, resulting in near random walk in the search space. From extensive experiments, we observed that, a small value of w (0.1) gives better results for most test circuits. This value has been used in subsequent runs of the fuzzy allocation based simulated evolution algorithm.

4.3.2 Comparison of SE, FSE_WA and FSE_FA

In order to see the performance of our proposed *fuzzy allocation scheme* (FSE_FA), we compare it with SE as originally reported in [8]. This is a single objective (optimization) simulated evolution algorithm. SE tries to optimize wire length only and reports the solution with the minimum wire length cost. We also compare FSE_FA with FSE_WA which is a multiobjective optimization algorithm and uses weighted average allocation scheme. Both FSE_FA and FSE_WA use a uniform *fuzzy-goal based cost computation* to identify the best generated solution.

The cost parameters of the best layout generated by all three schemes are reported in Table 4.6. It is clear that except for few cases (like **c532** and **c880**) the proposed fuzzy allocation scheme (FSE_FA) is able to generate a better quality solution in terms of reduced wire length cost (L) than FSE_WA. The wire length increase for **c532** and **c880** can be attributed to the fact that the weight combination in the

Circuit	FSE_FA ($w = 0.0$)				FSE_FA ($w = 0.05$)			
	L	D	W	T	L	D	W	T
highway	7665	5.80	520	0.36	7678	5.74	520	0.51
fract	32625	13.55	784	3.5	30732	12.98	792	2.86
c499	56970	14.67	1184	7.5	542020	15.24	1192	13.5
c532	76182	36.71	1160	11.0	73983	37.58	1160	36.28
c880	139632	32.05	1856	33.76	142736	31.09	1856	33.38
c1355	304094	26.04	2320	135.0	291250	26.65	2320	117.0
struct	704889	31.07	3344	167	700370	28.57	3344	176.0
c3540	775272	49.33	3152	436.0	801476	55.15	3176.0	660.0

(a)

Circuit	FSE_FA ($w = 0.1$)				FSE_FA ($w = 0.2$)			
	L	D	W	T	L	D	W	T
highway	7735	5.56	520	0.5	7863	5.69	520	0.65
fract	31528	13.62	784	4.0	34596	13.53	792	5.71
c499	56506	14.13	1200	7.5	63597	15.54	1208	15.3
c532	80779	37.96	1160	16.0	81468	39.57	1160	20.5
c880	137309	29.46	1872	45.7	139021	32.7	1840	54.0
c1355	290221	27.05	2320	144.0	322039	27.6	2352	207.0
struct	667850	28.8	3336	161	689286	28.87	3368	221.0
c3540	750153	46.01	3152	674	858277	53.71	3168	990.0

(b)

Circuit	FSE ($w = 0.3$)			
	L	D	W	T
highway	7991	5.7	520	0.7
fract	36851	13.5	792	10.7
c499	62284	15.7	1208	19.0
c532	90042	39.7	1184	26.5
c880	151689	31.1	1896	56.5
c1355	385238	32.2	2416	366.0
struct	687552	27.7	3376	281.0
c3540	924542	57.0	3176	1369.0

(c)

Table 4.5: Results of FSE_FA for different sizes of *fuzzy window* (w). Where wire length (L) and layout width (W) are in micron. The circuit delay (D) and execution time of the algorithm (T) are in ns and minutes respectively.

FSE_WA allocation strategy is suitable for medium range circuits only. For smaller circuits (like **highway**) or bigger circuits (like **c3540**) these weight combinations do not perform well. Furthermore, for **c532** and **c880**, the reduction in the circuit delay compensates the loss in wire length. When we compare the best generated solution for the delay (D) objective, FSE_FA surpasses FSE_WA for almost all circuits. The **struct** circuit is an exception because it is a structured multiplier where all the paths are of similar complexity; therefore, reduction in the net delays of all nets forming those paths is required to achieve overall reduction in the circuit delay. The layout width of both schemes is comparable for all circuits. This is due to the fact that the widths of generated layouts are very close to the respective lower bound on the maximum width.

We observed that both FSE_FA and FSE_WA were able to generate better quality solutions than SE. It is clear from the Table 4.6 that FSE_FA generates solutions with reduced wire length and delay costs than SE. This trend is valid for all of the test cases except **c532**, where SE is able to generate solution with less wire length cost. In spite of this, the circuit delay cost of FSE_FA generated solution for **c532** is less than SE and it compensates the loss in wire length. The difference in the delay cost between both schemes increases as the complexity of the test circuit increases. It means that FSE_FA is more adaptable to the size of the circuit. This improvement in solution quality is due to the use of problem specific intelligence added in FSE_FA algorithm. On the other hand, SE relies only on minimization

of wire length with an expectation that it will also reduce delay and area. SE generates less wire length cost solutions for most of the cases than FSE_WA. This is due to the fact that SE uses only wire length reduction as a measure in allocation stage and identifies the best solution as the one with the minimum wire length. While FSE_WA does tradeoff between multiple objectives in allocation stage as well as identifying the best solution. Therefore, we see that the wire length cost of FSE_WA generated best solutions is more than SE generated solutions. However, we observe that FSE_WA is able to give reduced circuit delay solutions than SE for all circuits except for **highway** and **c532**. The difference in the delay cost for **c532** is negligible therefore we ignore it. The **highway** is the smallest circuit and we have observed that the allocation weights used in our experiment were not suitable for small size circuit. Therefore, in this case, FSE_WA does not perform well.

Circuit	FSE_FA			FSE_WA			SE		
	L (μ)	D (ns)	W (μ)	L (μ)	D (ns)	W (μ)	L (μ)	D (ns)	W (μ)
highway	7735	5.56	520	9919	6.15	520	8109	5.85	520
fract	31528	13.62	784	37285	14.58	800	35945	14.76	792
c499	56506	14.13	1200	59278	14.51	1200	57194	15.85	1216
c532	80779	37.96	1160	72789	38.55	1184	75872	38.46	1168
c880	137309	29.46	1872	135509	30.92	1848	144501	35.36	1848
c1355	290221	27.05	2320	335589	28.41	2344	297677	32.05	2336
struct	667850	28.8	3336	685328	26.65	3312	672735	32.70	3344
c3540	750153	46.01	3152	844069	54.03	3152	787199	58.72	3136

Table 4.6: Best layout found by SE, FSE_WA and FSE_FA.

In order to compare the quality of search space between FSE_FA, FSE_WA and SE, we plot different cost parameters versus iteration count of the algorithm for the

circuit **c3540**. Figure 4.5 shows the wire length cost of solutions found after every 20 iterations. From these plots, it is clear that FSE_FA is able to generate reduced wire length cost solutions than SE and FSE_WA. On the other hand, SE produces solutions with less wire length than FSE_WA. This is due to the fact that it only optimizes wire length while FSE_WA tries to minimize multiple cost parameters. Figure 4.6 plots circuit delay cost for these schemes. Again, FSE_FA generates less delay cost solutions than SE and FSE_WA. FSE_WA performs better than SE because it is a multiobjective optimization algorithm, minimizing length, delay and width. Figure 4.7 plots respective layout width for these schemes. From this plot we conclude that all three schemes result in comparable layout width. After comparing individual cost parameters, overall solution quality (cost) is compared in Figure 4.8. This figure gives the membership values of solution in the fuzzy set *acceptable solutions*. We have used the *fuzzy-goal based cost measure* to quantify the quality of generated solutions. We are maximizing the membership in the fuzzy set *acceptable solution*. The graph in Figure 4.8 shows that FSE_FA is able to generate overall better quality layouts than other two schemes, and FSE_WA generates slightly better quality layouts than SE. Figures 4.9(a) and (b) plot the behavior of current average goodness and best average goodness for these schemes. The average goodness of FSE_FA plot is better than FSE_WA and SE because the former scheme is able to find solutions with less wire length. Since SE is able to find better wire length solutions than FSE_WA, its average cell goodness is better than FSE_WA. This behavior

is due to the wire length based goodness measure in these algorithms. This figure also shows that quality of solution is improving, since, as the algorithm progresses, more and more cells are approaching their respective near optimal positions in the layout. It also shows that the algorithm converges because less number of cells are selected for perturbation.

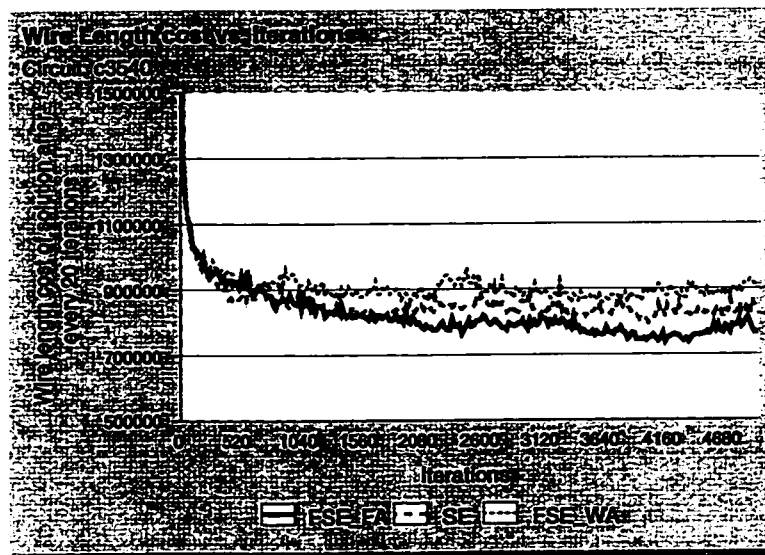
In order to show the stability of our implementations, we plot the cardinality of selection set against the number of iterations of the algorithm. Figure 4.10 shows such a graph for the circuit **c3540**. From the graph, it is clear that for all implementations of the simulated evolution algorithm i.e., SE, FSE_FA, and FSE_WA, the size of the selection set is decreasing with the increase in the number of iterations. This shows that the goodness of cells is increasing resulting in improved cost and smaller number of perturbations.

4.3.3 Comparing FSE_WA and FSE_FE

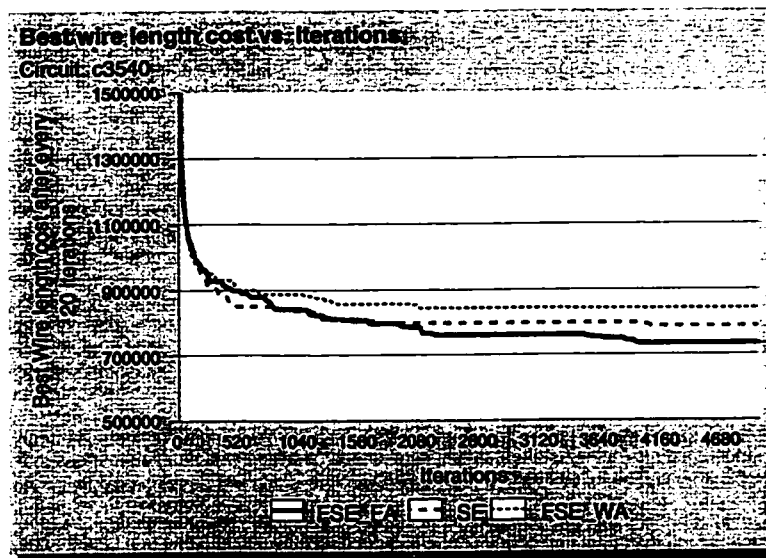
The objectives of the third set of experiments are to see the effect of:

1. Using net timing constraints in VLSI cell placement process.
2. Fuzzification of evaluation stage on the simulated evolution algorithm.

In order to achieve the above objectives we compare the results FSE_FE and FSE_WA variations of simulated evolution algorithm. Table 4.1 gives the characteristics of these algorithms. FSE_FE uses a fuzzy evaluation scheme in which goodness of an

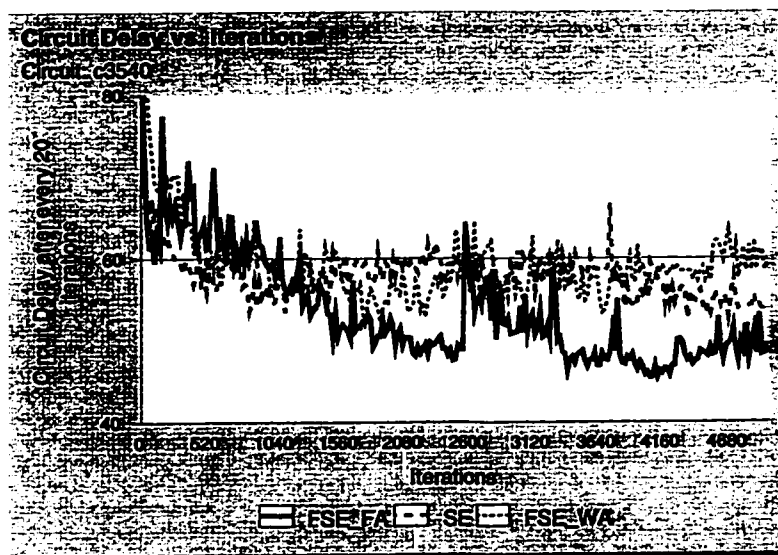


(a)

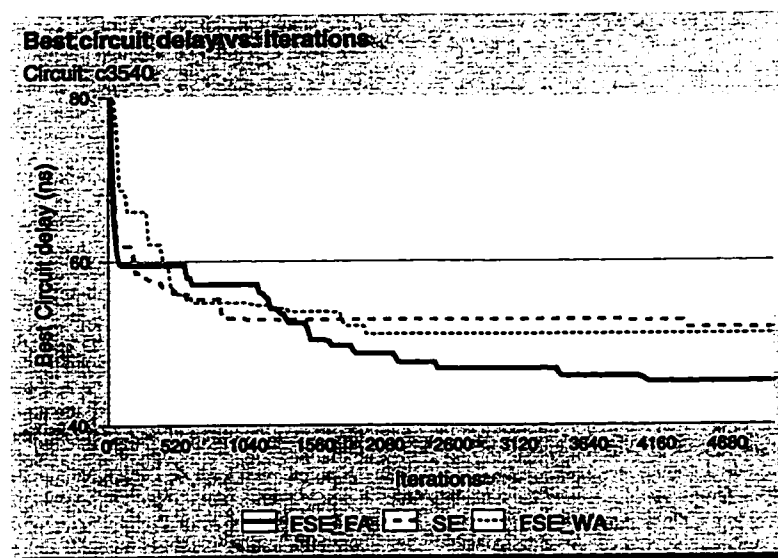


(b)

Figure 4.5: Wire length costs of solutions against iteration count: (a) length of the current solution (after every 20 iterations) (b) length of the best solution

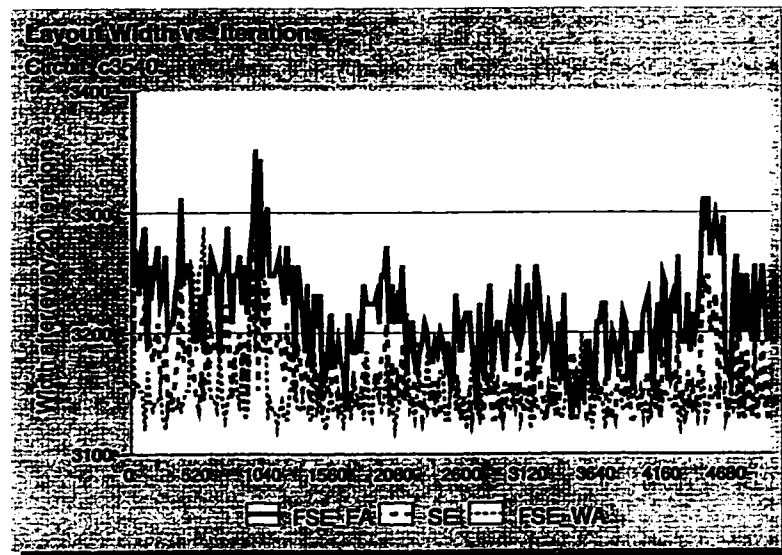


(a)

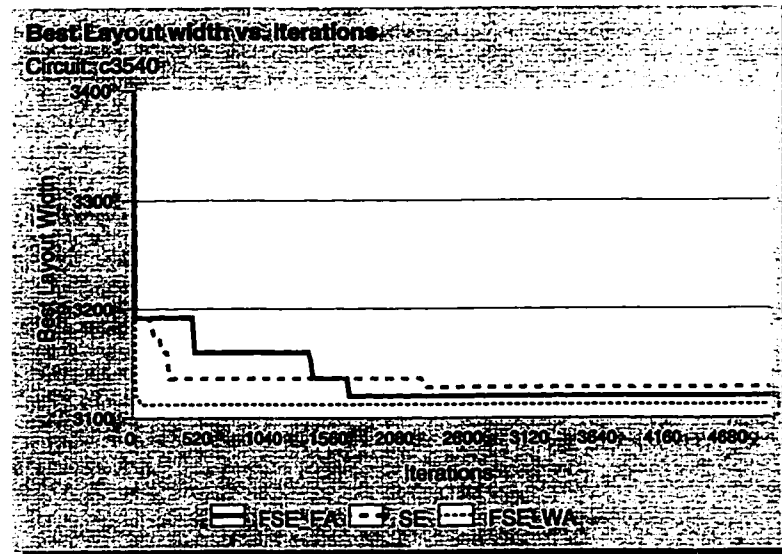


(b)

Figure 4.6: Circuit delay of solutions: (a) delay of the current solution (after every 20 iterations) (b) delay of the best solution.

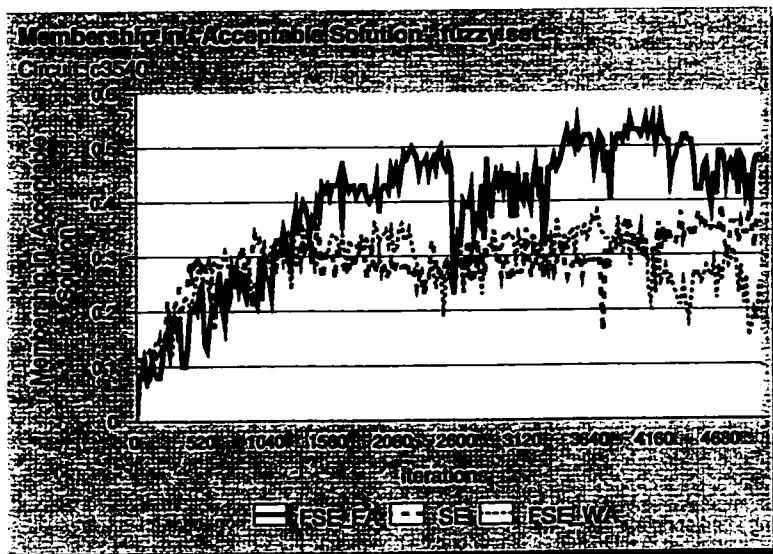


(a)

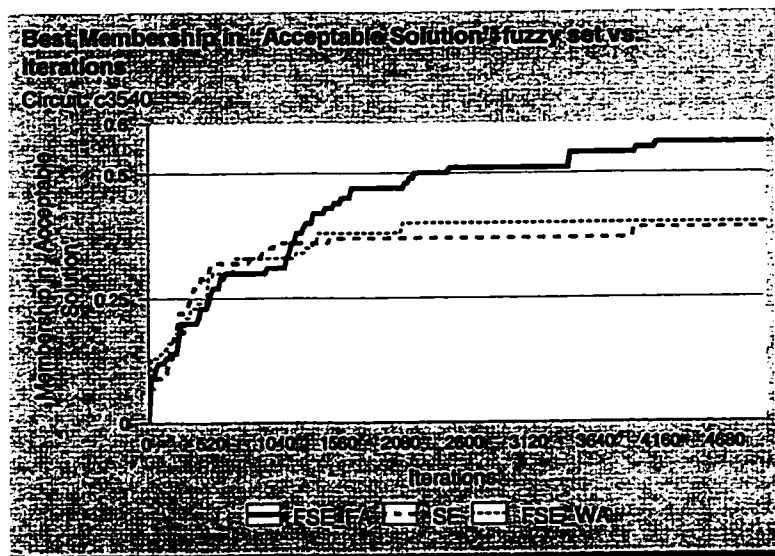


(b)

Figure 4.7: Layout width against iteration count: (a) width of the current solution (after every 20 iterations) (b) width of the best solution.

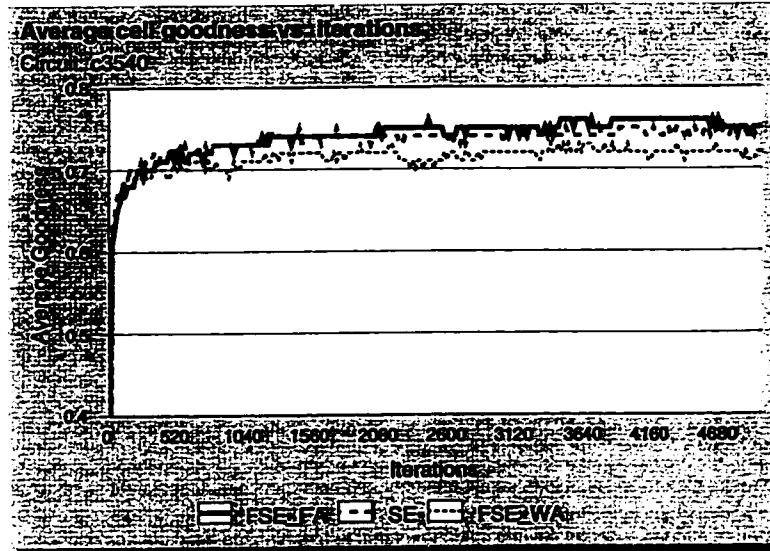


(a)

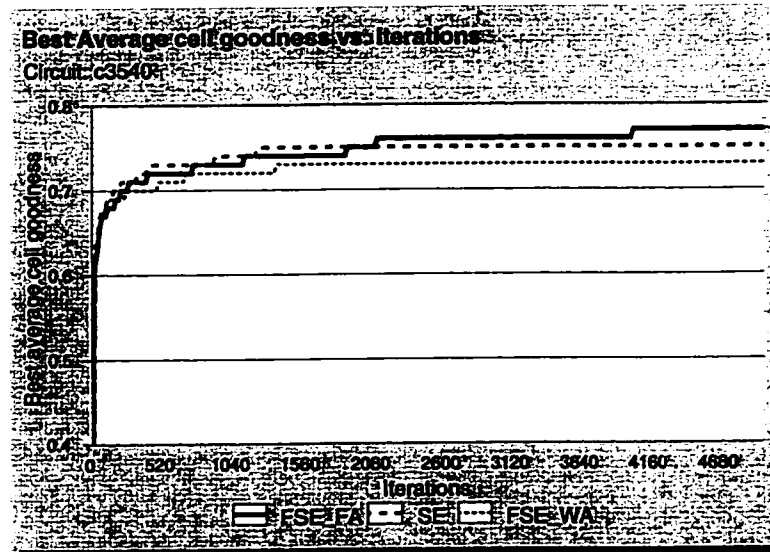


(b)

Figure 4.8: Overall cost of the solutions represented as membership values of solutions in fuzzy set *acceptable solution*: (a) current membership values (after every 20 iterations) (b) best membership values.



(a)



(b)

Figure 4.9: Average cell goodness against iteration count: (a) cell goodness of the current solution (after every 20 iterations) (b) goodness of the best solution.

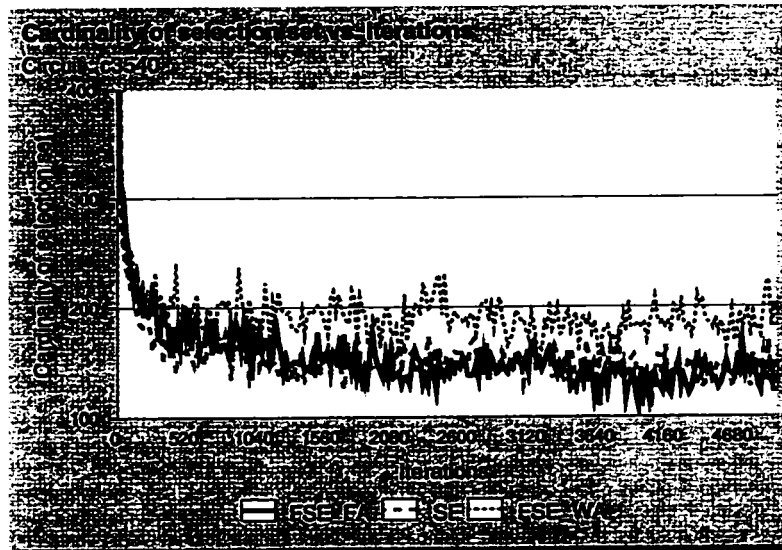


Figure 4.10: Cardinality of selection sets of FSE_FA, SE and FSE_WA.

individual cell has two components. One component is satisfaction of wire length and the second component represents the satisfaction of the net timing bounds (constraints). These two components are combined using the fuzzy rule and membership functions, as described in Section 4.2.3. For Equation 4.7 we used $\beta^e = 0.7$. While FSE_WA uses only one component, that is, satisfaction of optimum wire length. Both these algorithms use a weighted average allocation scheme described in Section 4.2.5. The values of allocation weights are $w_1^a = 0.6$, $w_2^a = 0.1$ and $w_3^a = 0.3$. The results of our experiment are compared and summarized in Table 4.7.

Generally, Fuzzy Evaluation (FSE_FE) gives inferior results than FSE_WA with respect to wire length. However, if we compare these algorithms with respect to circuit delay, FSE_FE is able to reduce the circuit delay for most of the circuits. We

have following explanation for this behavior.

1. The wire length cost of FSE_FE increases because cells are not selected on the basis of wire length only. Having a component for the net constraints satisfaction in cell goodness results in non-selection of many wire length violating nets. On the other hand, many nets satisfying wire length constraints were selected because they were violating the net constraints. These factors resulted in the increase in wire length.
2. The delay of a circuit is due to overall path delay of the longest path. While in our scheme, we are trying to select nets violating timing constraints and expect that the net might be placed in a location which improves the timing delay. We have got an improvement in the timing delay of most of the circuits. For big circuits this improvement is significant.

From this experiment we conclude that net constraints when used with wire length constraints in selection stage of the simulated evolution algorithm result in marginal improvement in circuit delay. On the other hand, they cause increase in wire length. Thus, we have not used the fuzzy evaluation scheme in our final version of Fuzzy Simulated Evolution Algorithm (Section 5.5).

Circuit	Fuzzy Evaluation (FSE_FE)			Weighted sum (FSE_WA)			% Gain		
	L (μ)	D (ns)	W (μ)	L (μ)	D (ns)	W (μ)	L	D	W
highway	9503	6.74	528	9919	6.15	520	4.2	-9.6	-1.5
fract	35635	13.50	808	37285	14.58	800	4.4	-2.2	-1.0
c499	60964	14.83	1192	59278	14.51	1200	-2.8	-2.2	0.6
c532	75216	35.80	1200	72789	38.55	1184	-3.3	7.1	-1.3
c880	137838	29.03	1852	135509	30.92	1848	-1.71	6.1	-0.2
c1355	349953	28.23	2368	335589	28.41	2344	-4.2	0.6	-1.0
c3540	883503	52.59	3128	844069	54.03	3152	-4.6	2.6	0.76

Table 4.7: Best layout found by FSE_FE and FSE_WA.

4.4 Conclusion

The VLSI placement is a hard and ill-defined problem with many conflicting objectives. In order to identify the best solution generated by the placement algorithm, we proposed a novel approach of *fuzzy goal-based cost measure*. This approach avoids the problems associated with the controversial weighted sum approach. It also allows easy incorporation of user preferences for different objectives. Furthermore, we proposed fuzzification of **evaluation** and **allocation** stages of the simulated evolution algorithm.

The fuzzy evaluation scheme uses a parameter for satisfaction of net timing constraints in the goodness computation of cells along with classical wire length measure. It results in improvements in circuit delay but slight increase in the wire length cost.

The proposed fuzzy allocation scheme combines controlled random move in a

purely constructive sorted individual best fit allocation technique. The identification of favorable locations for a cell is carried out through the use of fuzzy rule and membership functions. Being the most important stage of the SE algorithm, fuzzy allocation results in noticeable overall improvement in the quality of final solution.

Chapter 5

Effect of Selection Bias on Algorithm Efficiency

In the simulated evolution (SE) algorithm, Kling and Banerjee have proposed the use of **fixed selection bias** [8]. The selection bias provides a mechanism of compensating any errors made in the estimation of the optimum cost used in the computation of goodness. Furthermore, it is used to limit the size of the selection set i.e. controlling the magnitude of the perturbation of current solution. In case of a **fixed selection bias**, the user provides the bias value before the execution of the algorithm. The bias effects the execution time as well as the quality of final solution. According to Kling and Banerjee, a low bias value takes more execution time than a high bias value. This is due to the large selection sets in case of low bias values [26]. Moreover, the quality of solution also fluctuates with changes in bias values. For

very high and very low bias values, the quality of solution is bad [26]. This poses one question. How to find out the most suitable bias value in advance, which results in best solution quality and minimum execution time? One solution to this problem is to do several trial runs of the algorithm with different bias values. However, these trial runs will result in finding the best bias value for only one instance of the problem. As the problem changes or even instance of the problem changes, another set of trial runs is required to find the best bias value. Therefore, it can't be used as a general value for any set of problems. Furthermore, it also causes excessive execution time for all trial runs. Other researchers have proposed a **normalized goodness** scheme with zero bias value [20, 47, 51]. The normalization technique spreads the individual goodnesses. However, as we will see, even this scheme fails in our problem of VLSI cell placement. As the size of the circuit increases, normalized goodness takes excessive execution time and deteriorates quality of the best solution.

In order to overcome these issues, we propose *variable bias* scheme. In our proposed scheme, bias value is not arbitrarily set, it depends on the state of the solution and is a function of the goodness of individual cells.

The variable bias scheme is compared with fixed bias and normalized goodness schemes by executing the SE algorithm on benchmark circuits of Table 4.3. Except for the experiments of Section 5.5, following SE algorithm parameters are used for all other experiments reported in this chapter. The stopping condition was a fixed number of iterations. From our experience of Chapter 4, 2000 iterations are enough

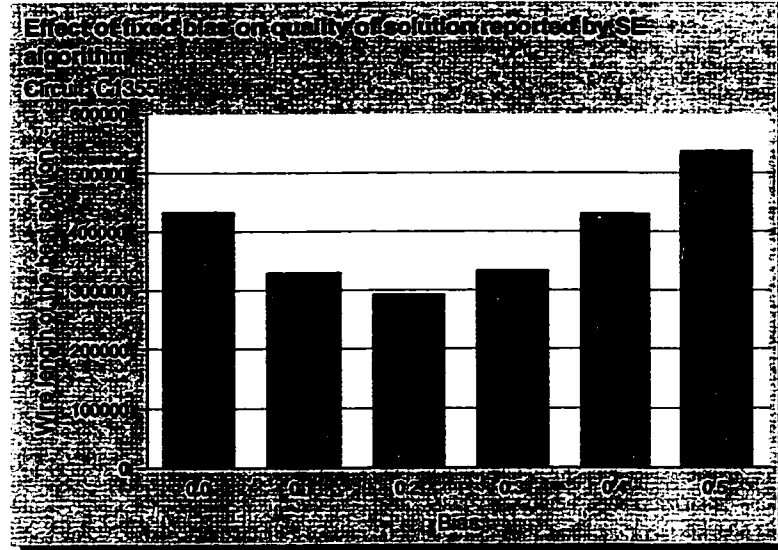
to understand the behavior of the algorithm. The wire length based classical evaluation and weighted sum based allocation schemes are used. Similarly, to simplify the analysis, only wire length cost of the best solution is considered as cost measure. However, in Section 5.5, where proposed fuzzy allocation scheme (Chapter 4) is combined with the variable bias measure, the SE algorithm parameters are the same as used for experiments in Chapter 4. The description of these parameters is repeated in Section 5.5.

This chapter is organized as follows. In Section 5.1, the effect of fixed bias on quality of solution and execution time is investigated. In Section 5.2, the goodness of individual cells is normalized as reported in [20] and its effect on performance of SE algorithm is examined. Section 5.3 describes our proposal of *variable bias*. The results of our experiments are discussed and compared in Section 5.4. In Section 5.5, we combine two schemes of Chapter 4, namely *fuzzy goal-based cost measure* and *fuzzy controlled random allocation* with *variable bias* scheme in one SE implementation. This implementation is labeled as FSE_VB. In this section, the results of FSE_VB are compared with another implementation of SE called as FSE_WA, which uses a fuzzy goal-based cost, weighted average allocation and fixed selection bias. We conclude this chapter in Section 5.6.

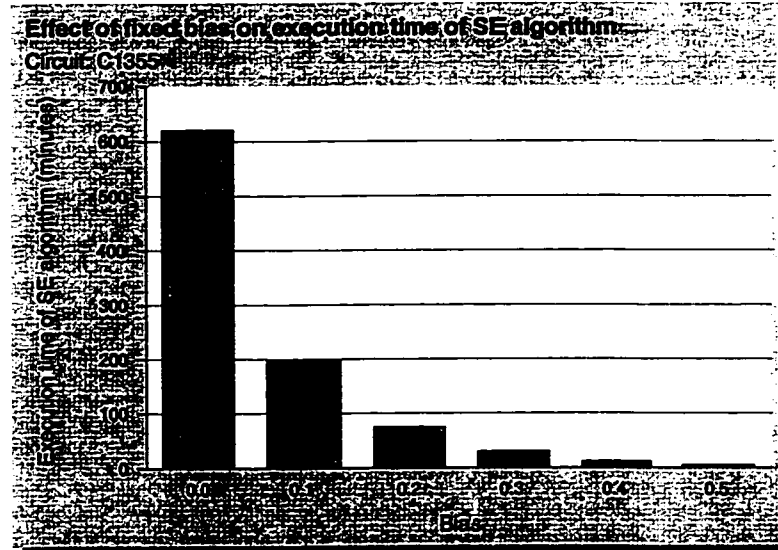
5.1 Effects of Bias on Quality of Solution

In this section, the effects of different values of fixed bias on the quality of solution are described. The SE algorithm was executed on all the circuits of Table 4.3 with different values of fixed bias. For this experiment we used bias values of 0, 0.1, 0.2, 0.3, 0.4, and 0.5. Table 5.1 shows the results of this experiment. In this table the wire length cost of the best solution and the corresponding execution time of the algorithm are given.

Figure 5.1(a) shows the effect of different bias values on the quality of solution for circuit **c1355**. In this figure, the interconnection length of the best solution reported is shown against bias values. It is clear that out of many bias values tested, the value of 0.2 gives best solution. Figure 5.1(b) shows the execution time for different values of bias. The execution time is high when the bias value is low and as bias value increases the execution time decreases. It is due to the fact that by increasing the bias value, a smaller number of cells are selected resulting in less amount of time spent in trial placement carried out in the allocation stage of the algorithm. This is also supported by Figure 5.2. In this figure, the cardinality of selection set is shown against the frequency of the solutions generated by the SE algorithm for different bias values. For clarity purposes, results for bias values 0.4 and 0.5 are not shown. For low bias values like 0.0 and 0.1, more solutions have large selection sets. While for higher bias values like 0.3, the frequency of solutions



(a)



(b)

Figure 5.1: The effect of bias on the Simulated Evolution algorithm. (a) quality of solution generated by SE (b) execution time of the algorithm.

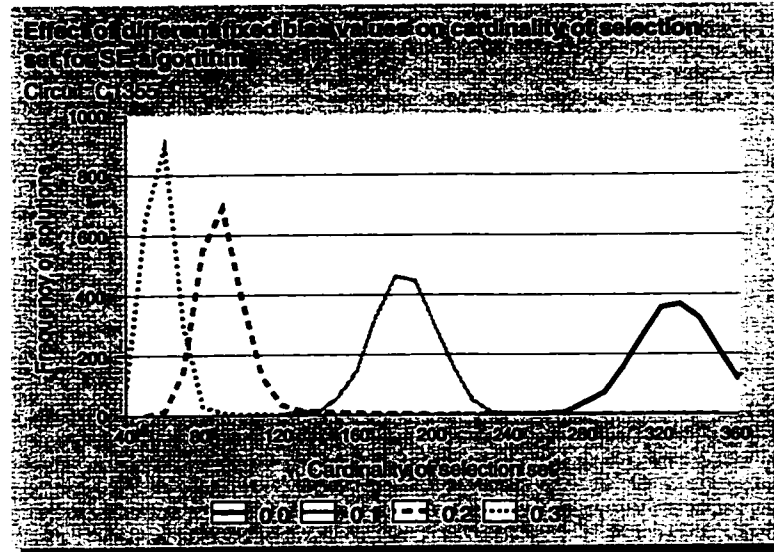


Figure 5.2: Cardinality of selection set against against frequency of solutions for different fixed bias SE algorithm.

with smaller selection sets increases.

The effect of bias on the quality of final solution can be described as follows. When we have very low bias value, a large number of cells are selected in each iteration. This will decrease the algorithm's ability to replace cells optimally due to uncertainty about unplaced cells. It will also increase the execution time of the algorithm. In contrast, when bias is very high, the algorithm ability to get out of local minima is restricted because badly placed cells have reduced chances of selection. Thus, the size of the selection set, execution time as well as the quality of solution space reached are reduced [26].

This gives rise to the following concern about fixed bias scheme. It is not practical for the user to determine the best bias value in advance by carrying out

Circuit	Parameter	Bias					
		0.0	0.1	0.2	0.3	0.4	0.5
highway	L	9093	8702	8042	9898	12816	13431
	T	0.2	0.12	0.1	0.09	0.08	0.08
fract	L	32140	32641	34407	39266	43449	51793
	T	3.0	1.5	0.6	0.4	0.32	0.32
c499	L	55523	54187	54100	62147	77624	96112
	T	14.6	7.2	3.5	1.7	1.6	1.1
c532	L	71495	70564	71344	77413	96379	120084
	T	16.8	10.4	4.9	2.6	1.4	1.2
c880	L	133742	136451	129089	150388	196548	268172
	T	12.4	28.4	12.4	6.4	4.0	2.8
c1355	L	432919	329872	292861	297592	429725	536780
	T	620.0	198.0	74.4	30.0	12.4	5.6
struct	L	991684	750862	675253	683596	745110	936332
	T	3045.0	1190.0	356.0	87.0	39.6	18.8
c3540	L	1387513	1505863	882139	857280	92666	1372237
	T	6774.0	3369.0	1213.0	360.0	108.0	38.0

Table 5.1: Effect of bias on quality of solution generated and execution time of the SE algorithm. The wire length cost of the solution (L) is in micron and the execution time (T) is in minutes.

several trial runs of the algorithm. Furthermore, these runs will be required for each instance of the problem. For example, for VLSI cell placement problem, different best bias values are required for different circuit sizes. The results of Table 5.1 confirm this assumption. The best bias values for circuits **c1355** and **c3540** are 0.2 and 0.3 respectively.

In the following section we will see a scheme, reported in the literature, to overcome the problems related to the fixed bias scheme. In Section 5.3, we propose an alternative approach to chose the bias value.

5.2 Normalized Goodness Values

In the previous section we have seen the effect of fixed bias on quality of solution and execution time of the SE algorithm. One solution to this problem is to normalize individual goodness. In [20, 47, 51], researchers have used normalized goodness measure with zero bias value in their simulated evolution implementations. The objective of the normalization is to spread the goodness within *Upper* and *Lower* bounds where $Upper + Lower \leq 1.0$. In one such implementation, Yuh et. al. [20] have normalized individual *score* in the range 0.05 and 0.95. Using their normalization method for our problem, we modified individual cell goodness as follows.

$$g'_{c_i} = 0.05 + 0.95 \times \frac{g_{c_i} - g_{c_i}^{\min}}{g_{c_i}^{\max} - g_{c_i}^{\min}} \quad (5.1)$$

Where

g_{c_i} = actual goodness of cell c_i as computed by Equation 4.6.

$g_{c_i}^{\min}$ = $\min (g_{c_i}) \quad \forall i$

$g_{c_i}^{\max}$ = $\max (g_{c_i}) \quad \forall i$

g'_{c_i} = normalized goodness of cell c_i .

Using the above equation, the best cell in a solution will have a goodness of 0.95. On the other hand, the worst cell will be identified with the goodness of 0.05.

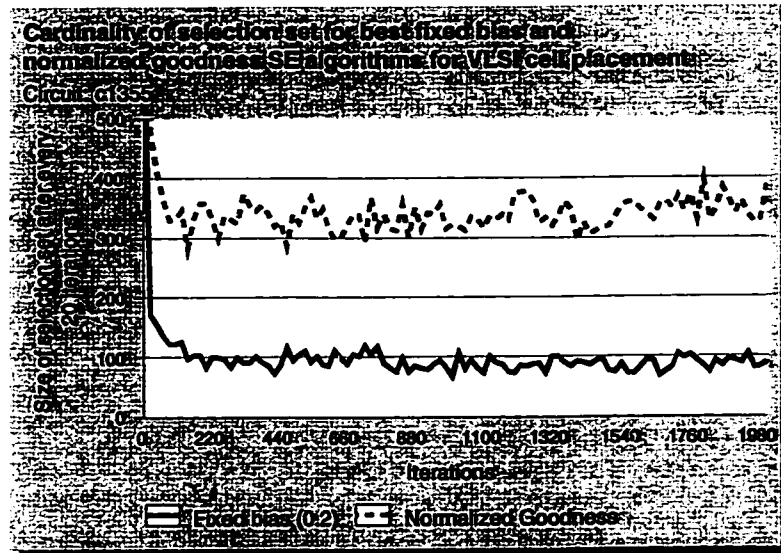
We tested this normalized goodness on benchmark circuits. In this experiment, zero selection bias was used. The quality of solution produced and execution time of this experiment are given in Table 5.2. The results of the best fixed bias SE algorithm are also reproduced for comparison purposes. From this table, it is clear that for small circuits the quality of solution of normalized goodness is comparable to that of the best fixed bias SE algorithm. However, as circuit size increases (e.g., circuit c880 onwards), cost of the (best reported) solution increases. For all cases, normalized goodness based algorithm takes more execution time than fixed bias SE, one order of magnitude more for big circuits. The reason for the increased execution time and bad quality solutions may be attributed for the following.

In case of normalized goodness SE, all cells are exposed for selection. Even the best located cell has a non-zero probability of selection. This increases the size of the selection set, forcing the algorithm to spend more time in each iteration. For small circuits, where layout dimensions are limited, individual cells have higher goodness values. Therefore, even after normalization, the selection set does not become huge. However, for bigger circuits, actual wire lengths can be large. Therefore more and more cells will have low goodness values. When these values are normalized between the range 0.05 and 0.95, the size of the selection set becomes huge. This reduces the algorithm's ability to place cells in correct positions. Therefore, solution quality deteriorates. The fixed bias algorithm saves itself from this situation by protecting all the c_i cells which satisfy the condition $g_{c_i} + B \geq 1.0$ from selection and subsequent

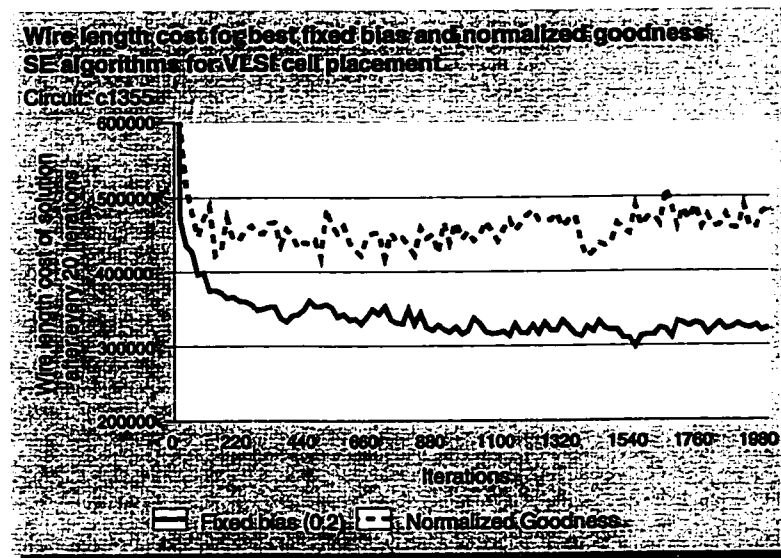
Circuit	Fixed Bias			Normalized Goodness	
	B	L	T	L	T
highway	0.2	8042	0.1	8891	1.0
fract	0.0	32140	3.0	32187	5.4
c499	0.2	54100	3.5	53643	20.4
c532	0.1	69193	10.4	65393	24.4
c880	0.2	128645	12.4	138467	72.0
c1355	0.2	292861	74.4	402939	648
struct	0.2	675253	356.0	953478	3180
c3540	0.3	857280	360.0	1403252	5760

Table 5.2: Comparison of solution quality and execution times of best fixed bias and normalized goodness SE algorithm. The wire length cost of the solution (L) is in micron and the execution time (T) is in minutes. B is best fixed bias value.

perturbation. Therefore, we see as the complexity of the circuit increases, higher values of fixed bias are required to give better quality solutions. In order to support the above analysis, we have shown two graphs. Figure 5.3(a) shows the size of selection set in the normalized goodness and the best fixed bias SE algorithms. Figure 5.3 (b) gives the cost of the best solution found by both schemes. We see that the size of selection set for the normalized goodness is bigger than the fixed bias SE resulting in bad quality of solution.



(a)



(b)

Figure 5.3: Comparison of best fixed bias and normalized goodness SE algorithm.
(a) size of the selection set (b) solution quality.

5.3 Variable Bias: Function of Average Cell Goodness ($B = \mathcal{F}(G)$)

As a modification to the concept of fixed bias value in the SE algorithm, we propose a *variable bias* which will be a function of *quality of solution*. When the overall solution quality is bad, a high value of bias will be used, otherwise a low value is used. The bias value will change from iteration to iteration and will depend on the quality of solution. Average cell goodness of a solution is a measure of how all the cells are near to their optimum positions. Therefore, we made the bias value function of average goodness i.e., for k^{th} iteration of the algorithm, bias B_k will be computed as follows.

$$B_k = 1 - G_{k-1} \quad (5.2)$$

where G_{k-1} is the average goodness of all the cells at the end of $(k - 1)^{st}$ iteration of the SE algorithm.

The advantages of this approach are following.

1. Bias value is not arbitrarily selected and no trial runs are required to find the best bias value. The variable bias automatically adjusts according to the problem state.
2. For bad quality solutions, the average goodness is low, resulting in a high bias value. This will make sure that the size of selection set is not excessively large.

It will save the algorithm from making big moves.

3. For good quality solutions, the average goodness is high. Therefore, a low bias value is used. It will result in the selection of sufficient number of cells and protect the algorithm from early convergence.

The modified algorithm was tried on eight ISCAS-89 benchmark test circuits. The stopping criteria was a fixed number of iterations. The results of our experiments are summarized in Table 5.3. Best results of fixed bias SE algorithm (given in Table 5.1) are reproduced here for comparison purpose. The quality of solution of the modified algorithm is comparable to the best fixed bias SE algorithm. The execution time of the variable bias SE algorithm is slightly more than the best fixed bias SE. However, if we consider the time spent in trial runs of the SE algorithm to find the best bias fixed value, then variable bias outperforms the fixed bias SE algorithm. There were at least 3 to 4 trial runs with different bias values to identify the best bias value for each circuit. The other advantage of variable bias is that bias is no longer a user specified value and there is no need to run several trials.

5.4 Comparison of Bias Schemes

In this chapter we discussed different bias schemes. Two performance metrics i.e., best wire length reported by the algorithm and execution time are compared in Tables 5.1- 5.3. Apart from these two parameters, we also observed other parameters

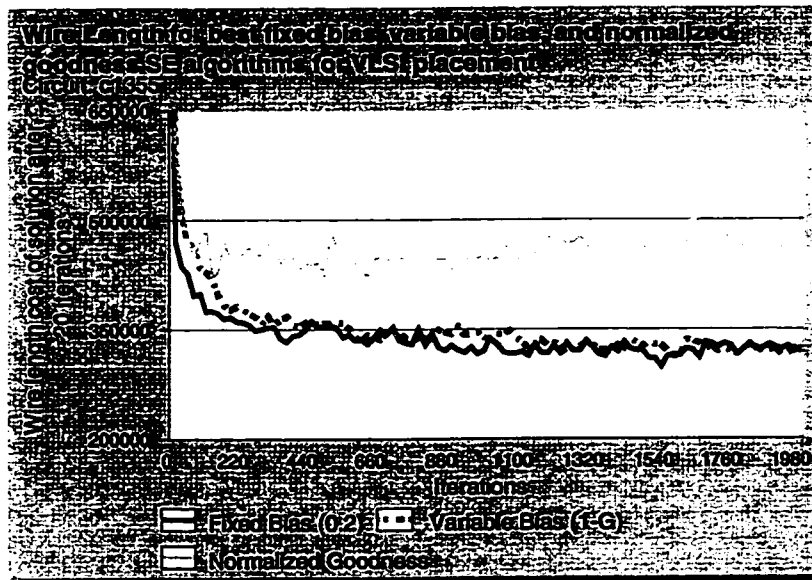
Circuit	Best Fixed Bias			$B = \mathcal{F}(G)$	
	B	L	T	L	T
highway	0.2	8042	0.1	8939	0.1
fract	0.0	32140	3.0	32907	1.0
c499	0.2	54100	3.5	55169	2.8
c532	0.1	69193	10.4	73477	5.6
c880	0.2	128645	12.4	129794	17.4
c1355	0.2	292861	74.4	290203	93.6
struct	0.2	675253	356.0	666822	222.0
c3540	0.3	857280	360.0	877130	502.0

Table 5.3: Comparison of solution quality and execution times of best fixed bias and variable bias ($B = \mathcal{F}(G)$) SE algorithms. The wire length cost of the solution (L) is in micron and the execution time (T) is in minutes. B is best fixed bias value.

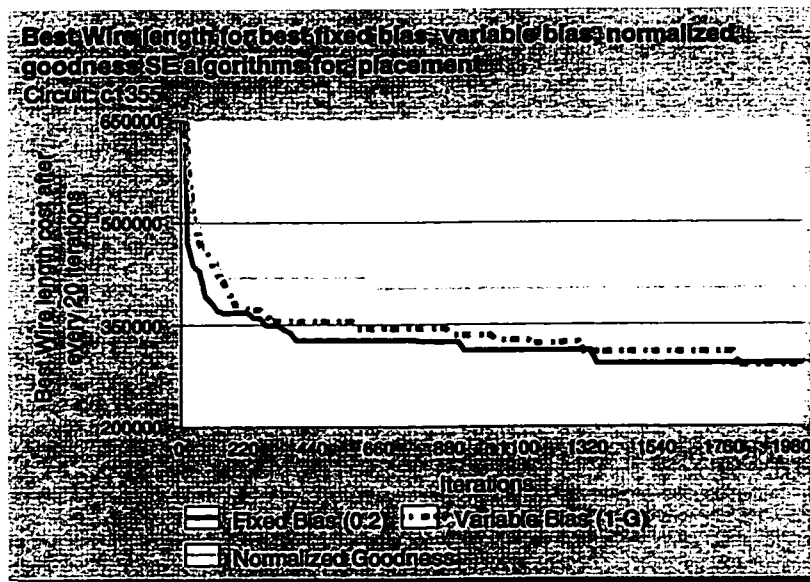
which allow us to compare the quality of search space investigated by each scheme. In this section we compare the **best fixed bias**, **normalized goodness**, and *variable bias* schemes This comparison is based on the following parameters:

1. Current and best wire length cost of solution.
2. Current and best average cell goodness.
3. Cardinality of selection set.

Figures 5.4(a) and (b) compare the search pattern for different bias schemes in the SE algorithm. Figure 5.4(a) and (b) respectively plots the current wire length and the best wire length determined by the algorithms. From this figure, it is clear that the quality of search for best fixed bias (bias=0.2) and variable bias ($1 - G$) are almost identical. These schemes decrease the cost sharply and then carry out



(a)



(b)

Figure 5.4: Comparison of different bias schemes (a) current wire length cost (b) best wire length cost

gradual refinements in the rest of the iterations. We have observed that the fixed bias value spends more time in earlier iterations because of the large selection sets. On the other hand, variable bias $(1 - G)$ scheme selects less number of cells than fixed bias and achieves the same improvement in solution quality. This considerably saves the execution time of the variable bias SE algorithm. Therefore, variable bias SE algorithm is able to get comparable quality solutions in reduced amount of time for most of the test circuits (Table 5.3). In order to support our assumption about the size of the selection set, Figure 5.5 plots the cardinality of the selection set for different bias schemes. From this figure it is clear that for variable bias $(1 - G)$, the selection set remains in a narrow band and slightly more than the selection set size of best fixed bias scheme. The reason for narrow band is that when average cell goodness is low, high bias maintains a limited size of selection set. Similarly when average goodness improves, a low bias does not allow the size of selection set to considerably decrease. This results in a narrow band for selection set. Due to this the algorithm does not spend extra time in early iterations.

The normalized goodness scheme tries to make the SE algorithm independent of the bias value. For small circuits, it results in comparable solutions with slightly more execution time (Table 5.2). However, for bigger circuits like **c1355**, **c3540** etc., it results in bad quality solution with excessive execution time. The pattern of search for this scheme is shown in Figure 5.4(a)-(b). The cost decreases in the early few iterations. This decrease is far less than the decrease obtained with best fixed

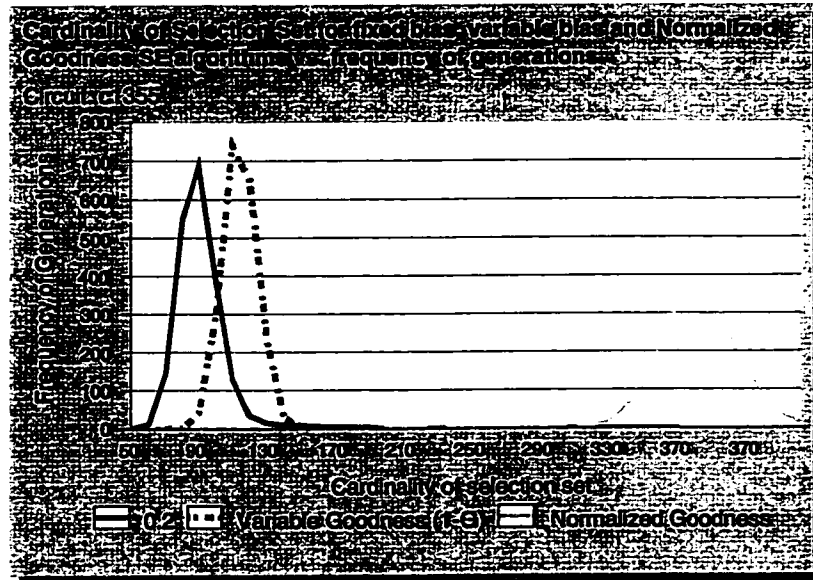
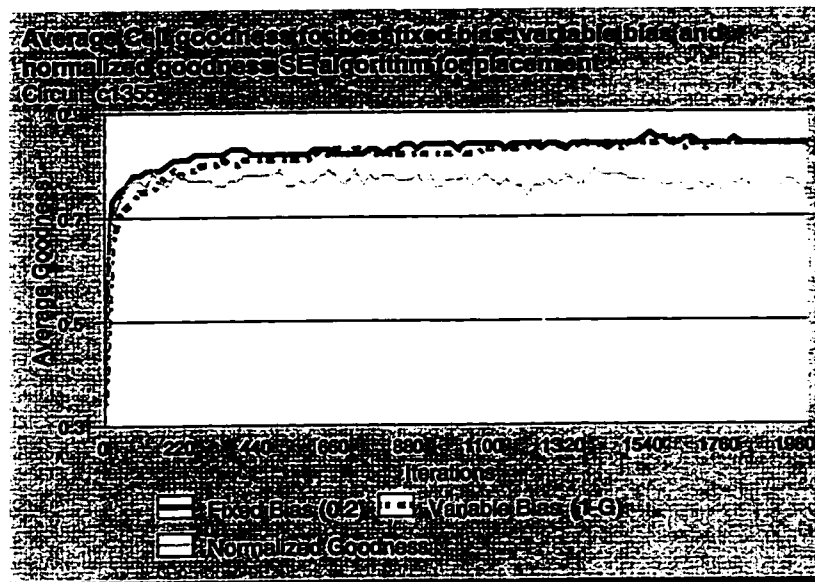


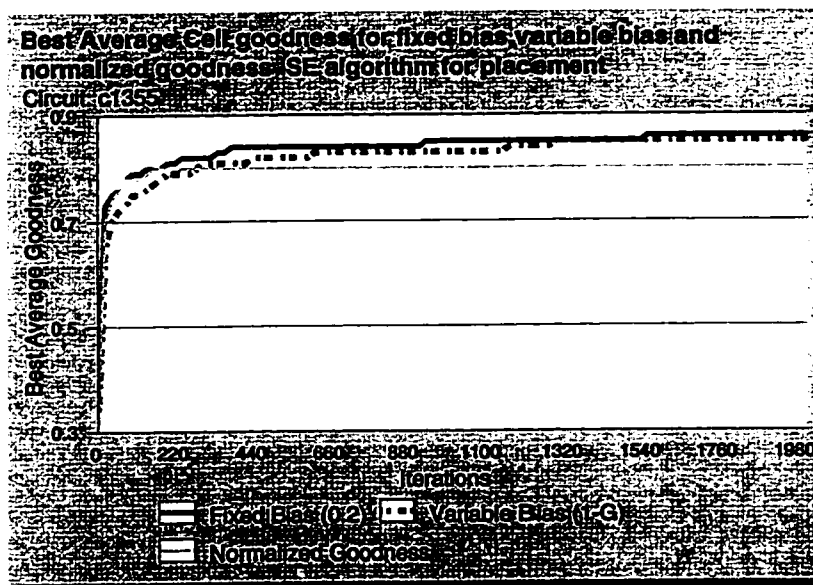
Figure 5.5: Cardinality of selection set against frequency of solutions for different bias schemes for the SE algorithm.

bias and variable bias ($1 - G$) schemes. Even thereafter, the normalized goodness SE algorithm converges to a much higher cost value. This behavior is attributed to the cardinality of the selection set as shown in Figure 5.5. The normalized goodness SE algorithm always has a bigger selection set than any of the other schemes. It is due to the fact that even the best placed cell has a non-zero probability of selection. This results in unnecessary perturbations of the well placed cells, reducing the quality of solution and increasing the execution time.

For all bias schemes discussed in this chapter, the current and the best average cell goodness are compared in Figure 5.6(a) and (b) respectively. For normalized goodness scheme, the current and best average goodness plots are for goodness



(a)



(b)

Figure 5.6: Comparison of different bias SE algorithms (a) average cell goodness (b) best average cell goodness

parameter before normalization. The average cell goodness represents the quality of the solution. A high average cell goodness means that many cells are able to achieve optimal positions in the solution. For all schemes, individual cell goodness represents the fitness of individual cells with respect to wire length cost. Therefore, we observe that schemes, which are able to find reduced wire length cost solutions, have higher average goodness values. This is true for best fixed bias and variable bias schemes. Both results in comparable behavior of average cell goodness. In contrast, normalized goodness based scheme has very low average goodness values. This is due to the fact that this scheme could not reduce wire length cost of the solutions as compared to other two schemes.

5.5 Fuzzy Allocation and Variable Bias SE algorithm (FSE_VB)

In Chapter 4, we proposed *fuzzy goal-based cost measure* and *fuzzy controlled random allocation scheme* which results in good quality solution. In this chapter, we proposed a variable bias scheme in which bias is a function of average cell goodness ($B = \mathcal{F}(G)$). In this section, we combine *fuzzy allocation* and *variable bias* concepts in one algorithm and label is as FSE_VB. In this section we will compare the results of FSE_VB with a weighted sum best fixed bias SE (FSE_WA) algorithm described in Chapter 4.

Circuit	FSE_VB				FSE_WA				
	L	D	W	T	B	L	D	W	T
highway	7978	5.8	520	0.3	0.2	9919	6.15	520	0.3
fract	32629	14.3	784	1.9	0.0	37285	14.58	800	5.5
c499	56913	14.6	1192	7.4	0.2	59278	14.51	1200	9.0
c532	77380	35.7	1168	16.1	0.1	72789	38.55	1184	23.0
c880	133729	30.0	1872	50.3	0.2	135509	30.92	1848	31.0
c1355	300419	25.6	2336	220.0	0.2	335589	28.41	2344	180.0
struct	646727	27.6	3408	500.0	0.3	685328	26.65	3312	240.0
c3540	779995	48.7	3240	907.0	0.3	844069	54.03	3152	1022.0

Table 5.4: Best layout found by FSE_VB and FSE_WA. The wire length cost (L) and layout width (W) are in micron. The circuit delay cost (D) is in ns and execution time of the algorithm (T) is in minutes.

The results of FSE_VB and FSE_WA are compared in Table 5.4. The FSE_VB is able to preserve all the good characteristics of fuzzy allocation scheme as well as variable bias scheme. As expected, FSE_VB is able to give good quality solution with respect to wire length and circuit delay costs. The layout width for both schemes is comparable. This is in line with our results of Section 4.3.2. Furthermore, the variable bias value also helps to improve the situation and saves the algorithm from early convergence. The execution time of the FSE_VB is also comparable to the FSE_WA algorithm. Also in FSE_VB, the best bias value does not have to be known in advance and it is automatically set by the algorithm.

5.6 Conclusion

In this chapter we investigated the effect of selection bias on the quality of solution produced by the simulated evolution (SE) algorithm. We examined two schemes reported in the literature. One uses a fixed bias value [26] and another uses normalized goodness scheme with zero bias value [20, 47, 51]. The best fixed bias value results in reduced solution cost as well as reduced execution time. However, finding out the best bias value is not straightforward. It requires several trials with different bias values. On the other hand, the normalized goodness avoids the use of bias value altogether. This scheme performs well for small circuits. As the size of the circuit increases, it causes excessive execution time and deteriorates quality of solution.

We proposed variable bias concept instead of static bias of [8]. In the proposed variable bias scheme, bias value is a function of average cell goodness. For each iteration of the SE algorithm, bias is computed as $1 - G$ where G is the average goodness of previous iteration. This makes the algorithm more adaptable to the overall quality of solution. It also reduces the size of the selection set in early iterations leading to a considerable saving in the execution time. It also saves the algorithm from getting trapped in local minima by decreasing the bias value for higher average goodness solutions. The results of this scheme are comparable to the best fixed bias scheme. The most important advantage of this scheme is that the bias is not a preset value. Here, bias adapts itself according to the state of the

solution.

In the end, we combined our proposals of *fuzzy goal-based cost measure*, *fuzzy controlled random allocation* and *variable bias* in one SE algorithm and label it as FSE_VB. The results of this implementation are compared with another implementation called FSE_WA, which uses fuzzy goal based cost measure, weighted sum allocation scheme, and best fixed bias values. The results show that FSE_VB gives better solutions than FSE_WA for wire length and circuit delay cost parameter. For layout width and execution time both algorithms are comparable.

Chapter 6

Conclusion

The *placement* problem of Very Large Scale Integrated Circuit (VLSI) consists of placing circuit modules on a layout in order to reduce cost parameters. Being NP-Hard, this problem can't be solved using full enumeration methods. Therefore intelligent heuristics are used to get sub-optimal solutions for this problem. Simulated Evolution (SE) is a general purpose iterative stochastic heuristic to solve such problems. It combines the iterative improvement and constructive perturbation. We have used SE for optimization of multiple objectives in standard cell VLSI placement problem. We are optimizing *wire length*, *circuit delay*, and *layout width*. We have used fuzzy logic based reasoning to combine conflicting objectives. In this thesis, we have fuzzified *evaluation* and *allocation* stages of the SE algorithms. Schemes for *Fuzzy allocation* and *fuzzy evaluation* are proposed. In order to compare layouts with conflicting cost values for interconnect length, delay and width, we have pro-

posed a *fuzzy goal-based cost* computation measure. We have also investigated the effect of selection bias on quality of solution and execution time of the SE algorithm. We have proposed the concept of variable bias value where bias is not static but a function of some parameter of layout. Our variable bias scheme makes the SE algorithm independent from any preset fixed bias value. In this scheme, bias is a function of average cell goodness and it changes from iteration to iteration.

Following are the conclusions of our research.

- *Fuzzy goal-based cost* measure allows us an easy to use, multiobjective adaptable cost measure. User preferences for any objective will be expressed in the form of goal vector.
- *Fuzzy Evaluation* scheme combines satisfaction of net bounds along with classical wire length only scheme. This results in reduction in circuit delay. However, this improvement is achieved at a cost of increased wire length.
- *Fuzzy Allocation* scheme combines controlled random move in a purely constructive sorted individual best fit allocation technique. The identification of favorable locations for a cell is carried out through the use of fuzzy rule and membership functions. Being the most important stage of the SE algorithm, fuzzy allocation results in noticeable in the quality of final solution.
- *Variable bias* proposal allows the value of bias to be a function of some layout dependent parameter. This results in making the bias more adaptable param-

eter to the quality of solution. The performance of *variable bias* is the best in terms of solution quality and execution time.

- Combining different proposals in a single version of the SE algorithm. For this we combined *fuzzy goal-based cost measure*, *fuzzy controlled random allocation* and *variable bias* schemes in one SE algorithm (FSE_VB). It is compared with a multiobjective weighted sum allocation based best fixed bias SE algorithm (FSE_WA). FSE_VB generates good quality results and its execution time is comparable in most of the cases.

Following is the summary of each chapter of this thesis.

In Chapter 1, we have briefly described the problem of VLSI cell placement and different solutions of this problem. A brief organization of the thesis is also given in this chapter.

In Chapter 2, we have included all relevant preliminary information, necessary to understand subsequent chapters. In this chapter we have formally defined the VLSI placement problem and reviews the standard cell design style. Cost functions to estimate the interconnect wire length, circuit delay and layout area are also given. Net bound computation method and details of technology are listed in this chapter. In this chapter, we have also reviewed the simulated evolution algorithm and fuzzy logic.

In Chapter 3, we have reviewed studies relevant to VLSI cell placement. This

review is classified as “general placement techniques” and “fuzzy logic based placement studies”. Review of applications of the simulated evolution algorithm for different computer-aided design problems is also given.

In Chapter 4, we propose fuzzy logic based SE algorithm for multiobjective optimization of VLSI cell placement. A *fuzzy goal based cost* computation scheme is proposed. For the *evaluation* stage of the SE algorithm, we have proposed a *fuzzy evaluation* scheme which tries to identify cells violating wire length as well as net constraints. This scheme is compared with the classical evaluation scheme in which only violation of wire length is identified. The results show an improvement in circuit delay at a cost of increased wire length. For the *allocation* stage of the SE algorithm, a *fuzzy allocation* scheme is proposed. This scheme tries to optimize wire length, circuit delay and layout width. We have also added an additional characteristic of controlled stochastic move with the help of a *fuzzy window*. The *fuzzy window* allows placement of cell on any of the near identical locations. The results are compared with a multiobjective weighted average allocation (FSE_WA) and single objective simulated evolution (SE) algorithm as proposed in [8]. According to the results, the proposed allocation scheme is able to generate solutions with reduced wire length circuit delay and comparable layout widths than other two schemes.

In Chapter 5, we investigated the use of selection bias in the SE algorithm. We studied existing methods and proposed variable bias concept. With our proposed modification, bias is treated as a dynamic parameter instead of fixed parameter as

in the case of the classical SE algorithm. Our proposed modification treat bias as a function *average goodness*. The results show that variable bias as a function of *average goodness* results in comparable execution time and quality of solution with respect to best fixed bias of classical algorithm. In this chapter we have combined our proposals of fuzzy allocation scheme, fuzzy goal-based cost measure and variable bias into one SE algorithm known as FSE_VB. This scheme was tested on benchmark circuits and compared with another implementation FSE_WA, which uses weighted sum allocation, fuzzy goal-based cost and best fixed bias. This algorithm is able to preserve the advantages of fuzzy allocation as well variable bias and results in reduction in solution cost.

In Chapter 6, we have summarized our thesis work. Here we have also mentioned some directions for future research.

6.1 Future Research

We think that our work can be extended to cover following issues.

- Path based minimization of circuit delay can be included in our proposed Fuzzy Simulated Evolution Algorithm. In that case, the allocation and evaluation stages of the proposed algorithm will be modified.
- Effect of dynamic bias on convergence of the SE algorithm.

- Implementation of a graphical user interface for our program. This interface should allow the users to interactively change search parameters online as well as view the layout characteristic from time to time.

Bibliography

- [1] Sadiq M. Sait and Habib Youssef. *VLSI Physical Design Automation: Theory and Practice*. McGraw-Hill Book Company, Europe (also co-published by IEEE Press), 1995.
- [2] K. Shahookar and P. Mazumder. VLSI Cell Placement Techniques. *ACM Computing Surveys*, pages 143–220, June 1991.
- [3] S. Sahni and A. Bhatt. The Complexity of design automation problems. In *17th Design Automation Conf.*, pages 402–411, June 1980.
- [4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, W. H. Freeman and Company, 1979.
- [5] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms and their Application to Engineering*. IEEE Computer Society Press (to appear in 1999).
- [6] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):498–516, May 1983.

- [7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, INC., 1989.
- [8] Ralph M. Kling and Prithviraj Banerjee. ESP: Placement by Simulated Evolution. *IEEE Transactions on Computer-Aided Design*, 8(3):245–255, March 1989.
- [9] K. Shahookar and P. Mazumder. A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization. *IEEE Transactions on Computer-Aided Design*, 9(5):500–511, May 1990.
- [10] K. Shahookar and P. Mazumder. GASP-A genetic algorithm for standard cell placement. *IEEE Design Automation Conference*, 9:660–664, March 1990.
- [11] S.M. Sait, H. Youssef, K. Nassar, and M. S. T. Benten. Timing driven genetic algorithm for standard-cell placement. In *Proceedings International Phoenix Conference on Computers and Communications*, pages 403–409, March 1995.
- [12] G. Holt and A. Tyagi. GEEP:A low power Genetic Algorithm layout system. In *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, pages 1337–1340, Aug. 1996.
- [13] K. Handa and S. Kuga. Polycell Placement for Analog LSI Chip Designs by Genetic Algorithms and Tabu Search. In *Proceedings of the Int. Conf. on Evolutionary Computations*, 1995.

- [14] C. Sechen and A. L. Sangiovanni-Vincentelli. Timberwolf3.2: A new standard cell placement and global routing package. *Proceedings of 23rd Design Automation Conference*, pages 432–439, 1986.
- [15] S. Mallela and L. K. Grover. Clustering based simulated annealing for standard cell placement. In *Proceedings of the 25th ACM/IEEE Design Automation Conference*, pages 312–317, June 1988.
- [16] R. Putatunda, D. Smith, M. Stebnisky, C. Puschak, and P. Patent. VITAL: fully automatic placement strategies for very large semicustom designs. *Proc. of Int. Conf. of Computer Design (ICCD)*, pages 434–439, Oct. 1988.
- [17] L. Tao and Y. Zhao. Multi-way graph partition by stochastic probe. *Computers Ops Res*, 20(3):321–347, 1993.
- [18] Fogel D.B. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, Jan 1994.
- [19] Ralph Michael Kling. *Optimization by Simulated Evolution and its Application to cell placement*. Ph.D. Thesis, University of Illinois, Urbana, 1990.
- [20] Yuh-Sheng Lee and A.C.-H. Wu. A performance and routability-driven router for FPGAs considering path delays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16:179–185, February 1997.

- [21] Sadiq M. Sait and Habib Youssef. Timing-influenced general-cell genetic floor-planner. *Microelectronics Journal*, 28(2):151–166, 1997.
- [22] H. Youssef, R. Lin, and E. Shragowitz. Bounds on Net Delays for VLSI Circuits. *IEEE Transactions on Circuit and Systems-II: Analog and Digital Signal Processing*, 39(11):815–824, Nov. 1992.
- [23] Habib Youssef. *Timing Analysis of Cell Based VLSI Designs*. Ph.D. Thesis, University of Minnesota, 1990.
- [24] MCNC Group. *Oasis 2.0 Reference Manual*, 1990.
- [25] Orbit Semiconductor Inc. *Foresight Manual*, 1992.
- [26] Ralph M. Kling and Prithviraj Banerjee. Empirical and Theoretical Studies of the Simulated Evolution Method Applied to Standard Cell Placement. *IEEE Transactions on Computer-Aided Design*, 10(10):1303–1315, October 1991.
- [27] J. M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proceeding of the IEEE*, 83(3):345–377, March 1995.
- [28] H. J. Zimmerman. *Fuzzy Set Theory and Its Applications*. Kluwer Academic Publishers, third edition, 1996.
- [29] L. A. Zadeh. Fuzzy Sets. *Information Contr.*, 8:338–353, 1965.

- [30] Ronald R. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Transaction on Systems, MAN, and Cybernetics*, 18(1):183–190, January 1988.
- [31] 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.
- [32] Q. E. Kang, R. Lin, and E. Shragowitz. Fuzzy Logic Approach to VLSI Placement. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):489–501, Dec. 1994.
- [33] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8:199–249, 1975.
- [34] W.E. Donath. Complexity theory and design automation. In *17th IEEE/ACM Design Automation Conference*, pages 412–419, 1980.
- [35] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Mich., 1975.
- [36] C. Sechen. *VLSI Placement and Global Routing using Simulated Annealing*. Kluwer Academic Publishers, Boston, 1988.

- [37] Youssef G. Saab and Vasant B. Rao. Combinatorial Optimization by Stochastic Evolution. *IEEE Transaction of Computer-Aided Design*, 10(4):525–535, April 1991.
- [38] J. P. Cohoon and W.D. Paris. Genetic Placement. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 422–425, 1986.
- [39] Khaled Muhammad Walid Nassar. *Timing Driven Placement Algorithm for Standard Cell Design*. M.S. Thesis, King Fahd Univ. of Pet. and Minerals, Saudi Arabia, June 1994.
- [40] M. Razaz and J. Gan. Fuzzy Set Based Initial Placement For IC Layout. *Proc. of the IEEE European Design Automation Conf. EDAC*, 9:655–659, 1990.
- [41] M. Razaz. A Fuzzy C-Means Clustering Placement Algorithm. *IEEE Int. Symposium on Circuits and Systems, ISCAS*, 3:2051–2054, May 1993.
- [42] C. F. Ball, P. V. Kraus, and D. A. Mlynski. Fuzzy partitioning applied to VLSI-floorplanning and placement. *IEEE Int. Symposium on Circuits and Systems, ISCAS*, 1:177–180, May-June 1994.
- [43] C. A. Mackey and J. D. Carothers. Performance-driven macro cell placement. In *15th Annual Int. Phoenix Conf. on Computers and Communications*, pages 427–433, March 1996.

- [44] T. Yukimatsu, T. Furuhashi, and Y. Uchikawa. A fuzzy expert system for hierarchical placement of parts on printed circuit board. In *Proc. of Second NZ Int. Conf. on Artificial Neural Networks and Expert System*, pages 342–345, Nov. 1995.
- [45] Hakeem Adiche. *Fuzzy Genetic Algorithm for VLSI FloorPlan Design*. M.S. Thesis, Computer Engineering department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, November 1997.
- [46] T.A. Ly and J. T. Mowchecko. Applying simulated evolution to high level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12:389–409, March 1993.
- [47] Yau-Hwang Kuo, Shaw-Pyng Lo, P. Dewilde, and J. Vandewalle. Partitioning and scheduling of asynchronous pipelines. In *CompEuro '92, Computer Systems and Software Engineering*, pages 574–579, May 1992.
- [48] Y. H. Hu and C.-Y. Mao. Solving gate-matrix layout problems by simulated evolution. In *IEEE Int. Symposium on Circuits and Systems*, volume 3, pages 1873–1876, May 1993.
- [49] L. Y. Wang, B. D. Liu, Y. T. Lai, and M. Y. Yeh. Performance-driven global routing based on simulated evolution. In *Proceedings of Computer, Commu-*

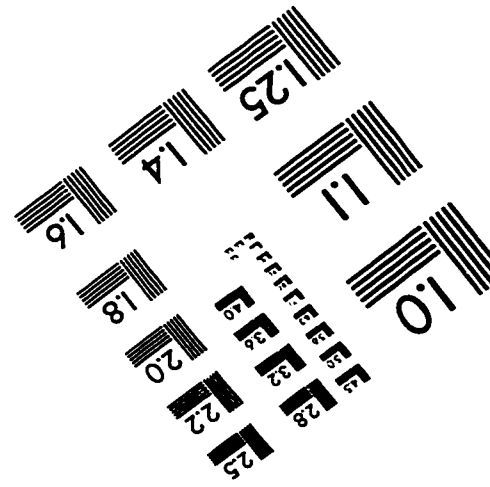
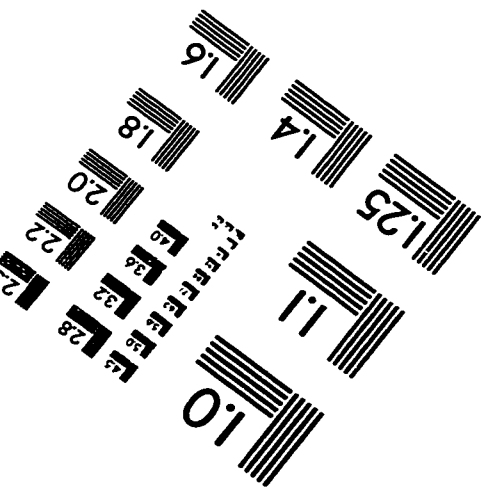
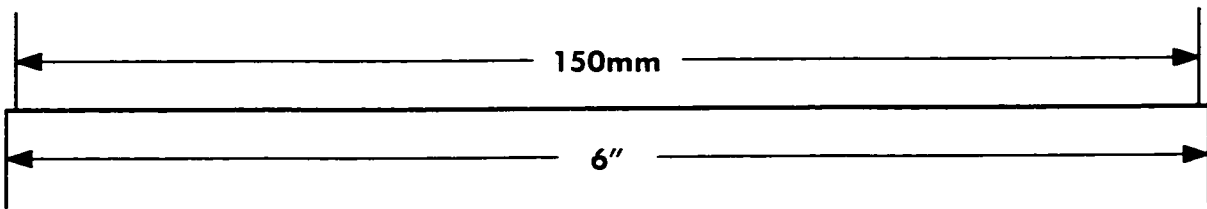
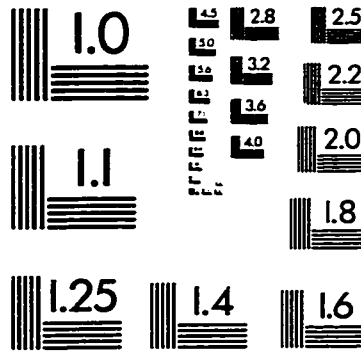
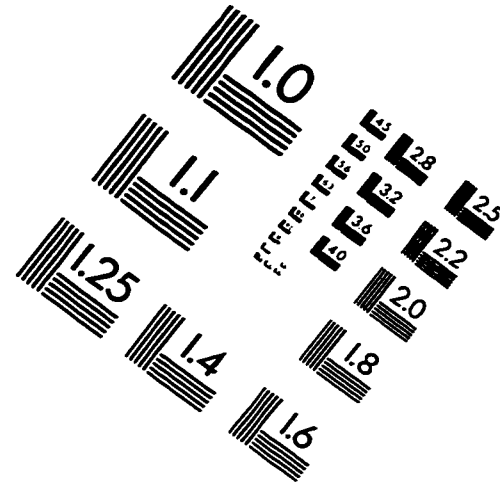
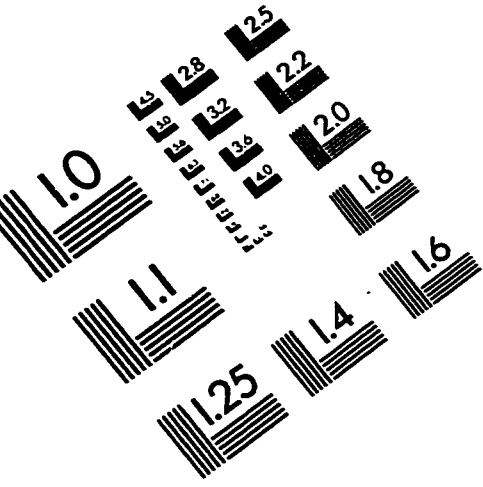
- nication, Control and Power Engineering, Conference TENCON '93* , pages 511–514, October 1993.
- [50] Y. L. Lin, Y. C. Hsu, and F. H. S. Tsai. Solving gate-matrix layout problems by simulated evolution. In *IEEE Int. Conf. on Computer-Aided Design, ICCAD-88*, pages 38–41, November 1988.
- [51] Y. L. Lin, Y. C. Hsu, and F. H. S. Tsai. SILK: A Simulated Evolution Router. *IEEE Transactions on Computer-Aided Design*, 8(10):1108–1114, October 1989.
- [52] Ching-Dong Chen, Yuh-Sheng Lee, A. C.-H. Wu, and Young-Long Lin. TRACER-fpga: a router for RAM-based FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:371–374, March 1995.
- [53] S. M. Sait, H. Youssef, K. Nassar, and M. S. T. Benten. Timing driven genetic placement. *Computer Systems Sciences & Engineering, CRL Publishing Ltd*, 13(6):125–133, Nov. 1998.
- [54] Henrik Esbensen and Ernest S. Kuh. Design Space Exploration Using The Genetic Algorithm. In *ISCAS '96. IEEE Int. Symposium on Circuits and Systems*, pages 500–503, 1996.

- [55] Eugene Shragowitz, Habib Youssef, Sadiq M. Sait, and Hakim Adiche. Fuzzy Genetic Algorithm for Floorplanning. In *Proc. of SPIE*, pages 36–47, July 1997.

Vita

- Syed Hussain Ali
- Born in Karachi, Pakistan.
- Received Bachelor of Engineering (B.E.) degree in Computer System Engineering from N.E.D. University of Engineering and Technology, Karachi, Pakistan in 1993.
- Joined the Department of Computer Engineering at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia as a Research Assistant in January 1997.
- Completed Master of Science (M.S.) in Computer Engineering at KFUPM in December 1998.

IMAGE EVALUATION TEST TARGET (QA-3)



APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved