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.

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.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

ITERATIVE ALGORITHMS FOR TIMING AND LOW POWER DRIVEN VLSI STANDARD-CELL PLACEMENT

BY

MAHMOOD-UR-REHMAN MINHAS

A Thesis Presented to the

DEANSHIP 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

JUNE 2001

UMI Number: 1404202

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by

MAHMOOD-UR-REHMAN MINHAS

under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean 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. Aiman H. El-Maleh (Member)

Dr. Habib Youssef (Member)

Department Chairman

Dean of Graduate Studies

12/6/2001
Date

Heartily dedicated to my family and especially to

my dear mother

whose prayers, love, and guidance
led to this accomplishment.

Acknowledgements

All praise to Allah, the most Beneficent and the most Merciful, who enabled me to complete my thesis work. I make a humble effort to thank Allah for his endless blessings on me, as His infinite blessings cannot be thanked for. Then, I pray Allah to bestow peace on his last prophet Muhammad (*Sal-allah-'Alaihe-Wa-Sallam*) and on all his righteous followers till the day of judgement.

I pay a heartily tribute to all of my family members and especially to my parents, who guided me during all my life endeavors. Their love and support motivated me to continue my education and achieve higher academic goals. Without their moral support and sincere prayers, I would have been unable to accomplish this task.

Next, I am deeply grateful to my thesis advisor Dr. Sadiq M. Sait for his valuable guidance throughout my thesis work. At the same time, gratitude is due to my thesis committee members Dr. Habib Youssef and Dr. Aiman H. El-Maleh. I express my thanks to all of them for their valuable time and support.

I acknowledge the academic and computing facilities provided by the Computer Engineering Department of King Fahd University of Petroleum and Minerals (KFUPM).

Finally, I appreciate the friendly support from all my colleagues at KFUPM. In particular, I want to thank Junaid, Salman, Shazli, Yassir, Atif, Ahmer, Raslan, and Aamir.

Contents

Acknowledgements iv

List of Tables x

List of Figures xi

Abstract (English) xiv

Abstract (Arabic) xv

1 Introduction 1

1.1 Background 1

1.2 VLSI Cell Placement 3

1.3 Iterative Algorithms 5

1.4 Organization of Thesis 11

2 Literature Review 12

2.1 Introduction 12

2.2 Techniques for Low Power Design 13

2.2.1 Techniques at Higher Levels 13

2.2.2 Physical Level 17

2.3 Techniques for Performance-Driven Placement 21

3 Problem and Cost Function Formulation **23**

3.1 Introduction 23

3.2 Problem Statement 24

3.2.1 Assumptions 24

3.2.2 Inputs and Outputs 25

3.3 Cost Functions Modeling 26

3.3.1 Cost Function for Interconnect wirelength 26

3.3.2 Cost Function for Power Consumption 28

3.3.3 Cost Function for Delay 30

3.3.4 Cost Function for Layout Width 32

3.4 Interconnect Capacitance and Resistance 32

3.5 Fuzzy Logic 34

3.5.1 Fuzzy Reasoning 35

3.5.2 Fuzzy Operators 37

3.5.3 Ordered Weighted Averaging (OWA) Operator 38

3.6 Fuzzy Cost Function for VLSI standard-cell Placement Problem . . . 39

3.7 Technology Parameters 43

4 Iterative Algorithms for VLSI standard-cell Placement 44

4.1 Introduction 44

4.2 GA for Timing and Low Power Driven VLSI Cell Placement 45

4.2.1 Chromosome Encoding and Initial Solution 46

4.2.2 Fitness Evaluation 48

4.2.3 Parent Choice 50

4.2.4 Crossover 51

4.2.5 Selection 54

4.2.6 Mutation 56

4.2.7 Stopping Criterion 57

4.3 TS for Timing and Low Power Driven VLSI Cell Placement 58

4.3.1 Initialization and Cost Evaluation 58

4.3.2 Neighborhood Generation 58

4.3.3 Tabu List and Aspiration Criterion 59

4.3.4 Stopping Criterion 60

5 GATS: A Hybrid Algorithm for VLSI standard-cell Placement 61

5.1 Introduction 61

5.2 Hybridization 62

5.3 GATS: Hybrid of GA and TS 62

5.3.1 Parameters of GATS 65

6 Experiments and Results **67**

6.1 Introduction 67

6.2 Circuits Details 68

6.3 Comparison of Proposed Techniques 68

6.3.1 GA versus TS 69

6.3.2 Comparison between TS and GATS 77

6.4 Sensitivity Analysis 78

6.5 Comparison of Various Crossover Operators and Selection Schemes for GA 83

6.6 Single Objective Versus Multiobjective Optimization 86

6.6.1 wirelength only optimization versus MOP 87

6.6.2 Power only optimization vs MOP 88

6.6.3 Delay only optimization vs MOP 92

7 Conclusions and Future Directions **95**

BIBLIOGRAPHY **98**

List of Tables

3.1 0.25 μ technology parameters. 43

6.1 Circuit and layout details. 69

6.2 Comparison between costs of the best solutions generated by GA and TS. 70

6.3 A comparison between the quality of the best solutions generated by TS and GATS. 78

6.4 The effect of varying the crossover probability in GA. 79

6.5 A comparison between using fixed and dynamic mutation probability in GA. 81

6.6 Effect of TS population size (N_T) on the performance of GATS. . . . 83

6.7 A Comparison between the quality of the best solutions obtained from GA by performing SOP for wirelength and MOP. 89

6.8 A Comparison between the quality of the best solutions obtained from TS by performing SOP for wirelength and MOP. 89

6.9 A Comparison between the quality of the best solutions obtained from GATS by performing SOP for wirelength and MOP. 90

6.10 A Comparison between the quality of the best solutions obtained from GA by performing SOP for power consumption and MOP. 90

6.11 A Comparison between the quality of the best solutions obtained from TS by performing SOP for power consumption and MOP. 91

6.12 A Comparison between the quality of the best solutions obtained from GATS by performing SOP for power consumption and MOP. 91

6.13 A Comparison between the quality of the best solutions obtained by performing optimization for delay only and multiobjective optimization. 93

6.14 A Comparison between the quality of the best solutions obtained from TS by performing SOP for delay and MOP. 93

6.15 A Comparison between the quality of the best solutions obtained from GATS by performing SOP for delay and MOP. 94

List of Figures

1.1 Various steps in VLSI design process. 4

1.2 Layout of a standard-cell placement. 5

1.3 Outline of simple Genetic Algorithm [1]. 7

1.4 Outline of Tabu Search algorithm [1]. 10

3.1 Steiner tree approximation to calculate the wirelength of a net. 27

3.2 Membership function of a fuzzy set A. 36

3.3 Range of acceptable solution set. 40

3.4 Membership functions within acceptable range. 41

4.1 An example of PMX operation. 52

4.2 An example of possible problems in using PMX when dummy cells are non-distinct negative integers. 53

5.1 GATS: A Proposed Hybrid of Genetic Algorithm (GA) and Tabu Search (TS) for Timing and Low Power Driven VLSI standard-cell Placement. 64

6.1 A comparison between GA and TS for circuit s832. (a) and (b) show the fuzzy membership of the best solution against run time for GA and TS respectively. 72

6.2 A comparison between GA and TS for circuit s832. (a) and (b) show actual costs for wirelength of the best solution against run time in case of GA and TS respectively. 73

6.3 A comparison between GA and TS for circuit s832. (a) and (b) show actual costs for power consumption of the best solution against run time in case of GA and TS respectively. 74

6.4 A comparison between GA and TS for circuit s832. (a) and (b) show actual costs for delay of the best solution against run time in case of GA and TS respectively. 75

6.5 A histogram of GA search for circuit s832. 76

6.6 A histogram of TS search for circuit s832. 76

6.7 Effect of population size on the quality of the best solution generated by GA for circuit s832. 79

6.8 A comparison between using dynamic and fixed mutation probability for circuit s832. 80

6.9 The effect of varying the neighborhood size on the performance of TS for circuit s832. 82

6.10 A comparison of Order, PMX, and CDX for circuit s832. 84

6.11 A comparison of three different selection schemes of GA for s832. . . 86

THESIS ABSTRACT

Name: MAHMOOD-UR-REHMAN MINHAS

Title: ITERATIVE ALGORITHMS FOR TIMING AND LOW POWER DRIVEN VLSI STANDARD CELL PLACEMENT

Major Field: COMPUTER ENGINEERING

Date of Degree: June 2001

In this thesis, the VLSI standard-cell placement problem is addressed with the objective of optimizing power consumption, timing performance, and interconnect wirelength, while layout width is taken as a constraint. This is known to be a hard optimization problem. Two iterative algorithms namely Genetic Algorithm (GA) and Tabu Search (TS) are presented for finding a good quality solution to the above said problem. In addition, a novel hybrid of GA and TS is proposed. Since the problem involves multiple possibly conflicting objectives, fuzzy rules have been incorporated in designing the overall cost function that integrates the costs of individual objectives in a single value. The proposed techniques are applied to the ISCAS-85/89 benchmark circuits and the results are promising.

MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

June 2001

ملخص الرّسالة

ا لا سم: محمود الرحمن منها س

عنوا ن الدراسة: خوارزميات تكرارية لتوزيع الخلية المعيارية للدوائر المتكاملة دات الكثافة العالية جدا (VLSI) بمراعاة التوقيت و الطاقة المستنفدة

التخصص: هندسة الحا سب اللآلي

تا ر يخ التخر ج: يونيو ٢٠٠١

في هذه الرّسالة، نركز على توزيع الخلية المعيارية للدوائر المتكاملة دات الكثافة العالية جدا (VLSI) مع الأهداف من تقليل الاستهلاك الطّاقة، أداء التوقيت، و طول الموصل، بينما يؤخذ عرض التّصميم كقيد. هذا عرف لكون مشكلة تفاؤل صعبة. أستعملنا قاعدتان تكرارتان، قاعدة جينيّة (Genetic Algorithm) والبحث تابو (Tabu Search) لإيجاد حلّ جيّد إلى هده المشكلة. بالإضافة لذلك، أقترحنا استعمال هجين مبتكر. تستلزم المشكلة مجموعة من الأهداف المتعارضة، وقد أدرجت القواعد المبهمة في التّصميم التّكلفة الكلّيّة التي تدمج تكاليف الأهداف المفردة في قيمة واحدة. أستعملت التكنيكات المقترحة إلى دوائر معيار ISCAS-85/89 و النتائج مبشّرا .

درجة الماجستير في العلوم

جامعة الملك فهد للبترول و المعادن

الظهران ، المملكة العربيّة السّعوديّة

يونيو ٢٠٠١

Chapter 1

Introduction

1.1 Background

VLSI circuit design can be performed with a view of achieving different objectives. Until the beginning of this decade, two main objectives of VLSI circuit design were focused. One was the minimization of interconnect wirelength and the other was the improvement of timing performance. A large number of efforts targeting either one or both of the above objectives is reported in the literature [2, 3]. The power consumption of the circuit was not of concern while trying to optimize the above two objectives.

Recently, the need for low power design has emerged. The optimization for power consumption can be performed at various levels of VLSI design including behavioral level, architectural level, logic level, and physical level. Some power

optimization techniques that have been proposed in the literature are reviewed in the next chapter. The optimization at each level can be performed subject to the degree of actual realization of the circuit. For example, it is not possible to optimize power consumption due to the interconnect capacitance at logic level, because a real wirelength estimation cannot be done at this stage. This fact enhances the need to perform the interconnect capacitance optimization at physical level. However, not much work is reported at the physical level.

This thesis addresses the multi-objective problem of simultaneously optimizing power consumption, timing performance, and wirelength of VLSI circuits at the placement step in physical level. standard-cell layout is considered in which all the cells in a circuit have the same height, but varying widths [2]. General iterative heuristics are suggested for solving this optimization problem.

Motivation

Since the mid 1990's, there has been a rapid increase in the use of mobile devices such as laptop computers, mobile phones and many others. Such kind of devices rely on a battery for their power needs. The battery power is limited by the factors of its size and weight, both of which are desired to be as small as possible. Also, the battery technology is expected to improve by only 30% in the near future [4].

Another compelling reason for the desire of low power consumption is the increasing density of VLSI circuits. With the rapid advancement in technology, VLSI

circuits are reducing in size resulting in higher transistor density on a chip. The present technology allows to integrate millions of transistors on a single chip and the still advancing technology is allowing further high integration. The excessive power consumption of the circuit results in heating and thus becoming a hindrance towards high integration and hence the feasible packaging of circuits [5, 6].

Also, the circuits are operating at much higher clock frequency than before. Therefore, the power dissipation which is a function of the clock frequency, is getting significantly prominent. This phenomenon is offering an obstacle in further increase of clock frequency. Due to these reasons, there is an emerging need for minimizing the power requirement of VLSI circuits.

1.2 VLSI Cell Placement

In this section, VLSI cell placement is described in brief. VLSI design is a complex process and is therefore broken down into a number of intermediate steps [2]. The design cycle starts from an abstract idea, and then each intermediate step continues refining the design and the process ends with the fabrication of a new chip as illustrated in Figure 1.1.

Placement is a phase in physical design responsible for the arrangement of cells on a layout surface while optimizing the costs of certain objectives such as the total wirelength or power consumption. The placement problem can be stated as follows.

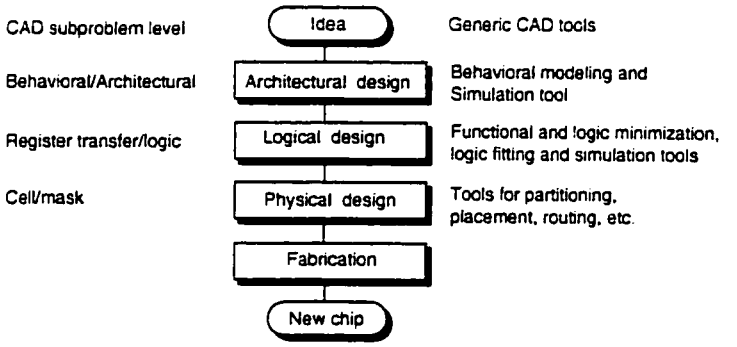


Figure 1.1: Various steps in VLSI design process.

Given a collection of cells or modules with pins (inputs, outputs, power and ground pins) on the boundaries, the dimensions of these cells, and a collection of nets (which are sets of pins that are to be wired together), the process of *placement* consists of finding suitable physical locations for each cell on the entire layout. By suitable we mean those locations that minimize given objective functions, subject to certain constraints imposed by the designer, the implementation process, or layout strategy and the design style [2]. The cells may be standard-cells, macro blocks, etc., but in this work, we shall be concerned with standard-cell design layout methodology.

In standard-cell placement, all the cells in the cell library are of the same height with variable widths (depending upon the complexity of the cell). Each cell has its ports at the top and at the bottom. Figure 1.2 shows the diagram of a standard-cell layout.

The placement of cells in order to optimize even a single objective e.g. interconnect wirelength is an NP-complete problem [2]. The simplest case of the problem, namely one-dimensional placement, is hard to solve due to the large size of search

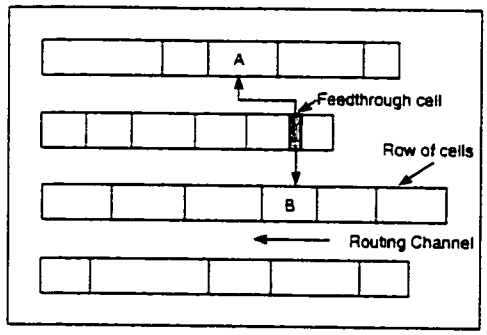


Figure 1.2: Layout of a standard-cell placement.

space. For n given cells, there are as many as $\frac{n!}{2}$ possible linear arrangements. Therefore, it is not possible to check all the arrangements in polynomial time. This fact favors the application of non-deterministic iterative algorithms over deterministic ones. The following section describes the suggested iterative algorithms to be employed for the present multi-objective placement problem.

1.3 Iterative Algorithms

A number of iterative algorithms are proposed in the literature. The motivation for using iterative algorithms becomes clear when recalling the hard nature of the VLSI cell placement problem as mentioned above. These algorithms are capable of efficiently searching for a near optimal solution in a large solution space and have been very successful in solving a number of combinatorial optimization problems in various disciplines of science and engineering including standard-cell placement [1]. In the following, a brief description of genetic algorithms (GAs) , tabu search (TS),

and simulated annealing (SA) is presented.

Genetic Algorithm (GA)

GA is an elegant search technique that emulates the process of natural evolution as a means of progressing towards the optimal solution. A high level algorithmic description of GA is given in Figure 1.3 [1]. GA uses an encoded representation of a solution in the form of a string made up of symbols called *genes*. The string of genes is called *chromosome*. The algorithm starts with a set of initial solutions called *population* that may be generated randomly or taken from the results of a constructive algorithm. Then, in each iteration (*known as generation in GA terminology*), all the individual chromosomes in the population are evaluated using a *fitness function*. Then, in the *selection* step, two of the above chromosomes at a time are selected from the population. The individuals having higher fitness values are more likely to be selected. After the selection step, different operators namely *crossover, mutation,* and *inversion* act on the selected individuals for evolving new individuals called *offsprings*. These genetic operators are described below.

Crossover is an important genetic operator. It is applied on two individuals that are selected in the selection step to generate an offspring. The generated offspring inherits some characteristics from both its parents in a way similar to natural evolution. There are different crossover operators namely *simple, order, partially mapped, and cycle*. The simple crossover operation for instance, works by

```
Algorithm (Genetic_Algorithm)
 (N_p = Population Size)
 (N_g = Number of Generations)
 (N_o = Number of Offsprings)
 (P_i = Inversion Probability)
 (P_μ = Mutation Probabilty)
 Begin
  (Construct initial population)
  Construct_Population(N_p);
  For j = 1 to N_p
   Evaluate_Fitness (Population[j])
  EndFor;
  For i = 1 to N_g
   For j = 1 to N_o
(Choose parents with probability proportional to fitness value)
     (x,y) ← Choose_parents;
     (Perform crossover to generate offsprings)
     offspring[j] ← Crossover(x,y)
     For k = 1 to N_p
      With probability P_μ apply Mutation (Population[k])
      With probability P_i apply Inversion (Population[k])
     EndFor;
     Evaluate Fitness(offspring[j])
  EndFor;
  Population ← Select(Population, offspring, N_p)
 EndFor;
 Return highest scoring configuration in population
End. (Genetic Algorithm)
```

Figure 1.3: Outline of simple Genetic Algorithm [1].

choosing a random cut point in both parent chromosomes (the cut point should be the same in both parents) and generating the offspring by combining the segment of one parent to the left of the cut point with the segment of the other parent to the right of the cut [1]. For description of other crossover operators, see [3, 1, 7].

The *mutation* operator is used to introduce new random information in the population. It helps to prevent the search process from trapping in local minima. An example of mutation operation is the swapping of two randomly selected genes of a chromosome. The importance of this operation is that it can introduce a desired characteristic in the solution that could not be introduced by the application of the crossover operator alone. However, mutation is applied with a low rate so that GA does not turn into a memory-less search process [3].

The quality of the solution obtained from GA is dependent on the choice of certain parameters such as population size, number of generations, crossover and mutation rates and also the type of crossover used. The selection of values for these parameters is problem specific and so there are no hard and fast rules for this purpose. The choice of these parameters is left to the conception and intuition of the person applying GA to a specific problem.

Tabu Search (TS)

Tabu search is an iterative heuristic that has been applied for solving a range of combinatorial optimization problems in different fields [1]. Tabu search starts from

an initial feasible solution and carries out its search by making a sequence of random moves or perturbations. A *tabu list* is maintained that stores the attributes of a number of previous moves. This list prevents bringing the search process back to already visited states. In each iteration, a subset of *neighbor* solutions is generated by making a certain number of moves and the best move (the move that resulted in the best solution) is accepted, provided it is not in the tabu list. Otherwise, if the said move is in the tabu list, the best solution is checked against an *aspiration criterion* and if satisfied, the move is accepted. Thus, the aspiration criterion can override the tabu list restrictions. It is desirable in certain conditions to accept a move even it is in the tabu list, because it may take the search into a new region due to the effect of intermediate moves. The behavior of tabu search heavily depends on the size of tabu list as well as on the chosen aspiration criterion. Different sizes of tabu list result in short-term, intermediate term, and long-term memory components that can be used for intensifying or diversifying the search. The aspiration criterion determines the extent to which the tabu list can restrict the possible moves. If a tabu move satisfies aspiration criterion, then the move is accepted and tabu restriction is overridden. The structure of TS is given in Figure 1.4. The detailed description of tabu search can be found in [1].

Algorithm *Tabu_Search*

Ω : Set of feasible solutions
S : Current solution
S* : Best solution
Cost: Objective function
N(S): Neighborhood of S ∈ Ω
V* : Sample of neighborhood solutions
T : Tabu list
AL : Aspirartion level

Begin

Start with an initial feasible solution S ∈ Ω
Initialize tabu list and aspiration level
For fixed number of iterations **Do**
    Generate neighbor solutions V* ⊂ N(S)
    Find best S* ∈ V*
    If move S to S* is not in T **Then**
        Accept move and update best solution
        Update T and AL
    Else
        If Cost(S*) < AL **Then**
            Accept move and update best solution
            Update T and AL
        End If
    End If
End For

End.

Figure 1.4: Outline of Tabu Search algorithm [1].

1.4 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 presents a survey of techniques reported in literature for VLSI cell placement. In particular, the application of iterative algorithms to the placement problem is reviewed. Also, a brief review of performance driven placement techniques is given.

In Chapter 3, the multi-objective placement problem is formulated. The cost functions for objectives i.e., interconnect wirelength, timing performance, power consumption as well as for width constraint are designed. An overview of fuzzy logic is also presented. Finally, the overall cost function used in multi-objective optimization is designed using fuzzy operators.

Chapter 4 discusses the implementation details of the proposed GA and TS, for multi-objective VLSI standard-cell placement. The setting of various GA and TS parameters is discussed. Then, Chapter 5 presents the implementation aspects of the proposed novel hybrid algorithm.

Experimental results for the application of the proposed techniques to ISCAS-85/89 benchmarks are discussed and compared in Chapter 6. The effect of various parameters on the quality of the final placement solution is studied. Some conclusions and possible future directions of work are given in Chapter 7.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews some recent techniques reported for optimizing the power consumption in VLSI circuits. As various techniques have been reported at different levels of the VLSI design process (see Figure. 1.1), we first review the power minimization techniques at the higher levels and then we review the techniques at the physical level. Later in the chapter, we also review some approaches that are targeted for optimization of the timing performance only.

2.2 Techniques for Low Power Design

In standard CMOS VLSI circuits, switching activity of circuit nodes is responsible for most of the power dissipation. It is reported in [8] that this switching activity contributes 90% to the total power dissipation in the circuit. Therefore, most of the reported techniques focus on minimizing the power consumption due to this switching activity at various levels of abstraction [9].

2.2.1 Techniques at Higher Levels

A brief review of the techniques for low power design at the higher levels is presented here.

Behavioral Level

Behavioral synthesis is the process of transforming high-level specification of a circuit into a register-transfer level design. There are a number of techniques reported that can be used to optimize power at this design step [10, 11]. The first technique is to modify the input high-level specifications through applying specific transformations that lead to power reduction. One of the most important transformation for fixed throughput systems is that which can reduce the number of control steps. Another class of transformations results from extension of the transformations that are employed to reduce the amount of resources required for the implementation of

a given control-graph. These transformations are then used to reduce the amount of capacitance that switches [9]. Some of such transformations that can be used in automated systems are described in [12]. Some specific transformations for DSP circuits can be found in [13].

Another technique for reducing power consumption at this level is to choose appropriate modules at the time of allocation in behavioral synthesis. But this choice of modules is limited by the availability of modules with a range of power consumption values, that can be used to implement a given operation type [9]. Although there is a trade-off between the power and the delay values for modules, even then appropriate choice of modules can optimize the power cost without increasing the delay [14]. Moreover there are a number of decisions made during the allocation step, including the sequence of mapping of operations in the control/data flow graphs to functional units, and extent of hardware sharing that affect the switched capacitance in the data path. Some techniques that have been used to control these critical decisions in order to minimize the resulting capacitance are described in [15, 16]

Logic Level

Different power optimization techniques have been reported for combinational and sequential circuits. We review techniques for both in brief.

For combinational circuits, logic synthesis is performed in two steps, namely technology independent and technology dependent. Power optimization can be achieved

by logic equations manipulation in the former and by appropriate mapping to a library in the latter. A technique to reduce the switching activity and hence the power consumption of a gate is through the use of don't care sets [17]. A method to reduce switching activity by utilizing don't care optimization is reported in [18].

Another technique for power optimization attempts to reduce spurious transitions that may contribute as much as 40% to the switching activity in the circuit [19]. This is achieved by path balancing in which the delays of paths that converge at each gate in the circuit are made roughly equal by inserting unit-delay buffers to the inputs of a gate. An application of such path balancing method is described in [20]. Another power optimization approach targets at reducing the number of transistors required in the implementation of a logic function. This is accomplished by *Kernel extraction* method that can be used to find and reuse common sub-expressions across multiple functions. Although the above method directly aims at reducing the number of literals and thus area, it has been modified to also reduce the switching activity and thus the dynamic power consumption [21].

Better technology mapping can also help in reducing power requirements of the resulting circuit. Typical cell libraries contain a large number of gates with different transistor sizes. Various low power oriented technology mapping techniques based on a graph covering method have been proposed [22, 23, 24].

For sequential circuits, power optimization techniques can be applied at the State Transition Graph (STG) level as well as at the logic-gate and flip-flop level.

One technique for power optimization is related with state encoding in the STG of a sequential circuit. It works by assigning uni-distant codes to two states that involve a large number of transitions between them. This will minimize the switching activity at the flip-flop outputs. Two such optimization techniques for assignment of codes to states are presented in [21, 25].

Another technique for power optimization is retiming that repositions the flip-flops to reduce the required clock period [26]. The switching activity at flip-flop inputs is much higher than that at flip-flop outputs. This is due to the filtration of many spurious transitions occurring at the inputs of flip-flops by the clock signal. This fact has been utilized for power optimization in a retiming technique reported in [27].

Gating of clocks is also a well-known technique for reducing power consumption in a sequential circuit. The Basic idea is to determine the conditions under which a particular set of registers do not change their state, and then to gate the clocks of these particular registers with those conditions [28]. The purpose is to save power by reducing the switching activity in these registers under certain values of the above conditions. Gating technique is also useful in the case of arithmetic units. When a particular unit is idle in a particular clock cycle, the power can be saved by turning off that unit. The gating idea is extended further in the form of another approach for power optimization called *pre-computation* [29]. In this technique, certain modules within a circuit are identified that remain idle during a particular computation and

those can be turned off to save the power consumption.

2.2.2 Physical Level

Reviewing the efforts for power optimization at the physical design step is important in the scope of this work because we are interested in this particular step of the VLSI design. Relatively, a small number of low-power-driven techniques at this design step has been proposed so far. The physical design process is carried out in a number of phases, including circuit partitioning, floorplanning, placement, and routing. Here, we review the power optimization techniques for all these phases.

For the partitioning phase, two low-power oriented techniques based on the simulated annealing algorithm have recently been presented in [30]. One algorithm uses the Shannon expansion-based scheme and the other uses the Kernel-based scheme. These algorithms partition the circuit into a number of sub-circuits such that a single sub-circuit needs to be active at a particular time. In this way, the unnecessary signal transitions are prevented. The circuit partitioning is performed by using an adaptive SA algorithm. The cost function is modeled for low-power consumption under given area constraints. A partitioning solution is obtained by recursive bi-partitioning of the circuit and the solution space is represented as a binary tree. The stopping criteria used is non-improvement in the solution for a constant number of moves. The performance of the algorithm is evaluated by its application to MCNC benchmark circuits and compared with the results of Synopsis design Analyzer to

show 8.7% power reduction over the latter without allowing any increase in the layout area.

An approach for optimizing power consumption at the floorplanning step has been proposed in [31]. This technique proceeds by selecting and placing circuit modules with the objective of reducing total power consumption as well as area. The proposed technique is based on the simulated annealing algorithm and works by bottom-up calculation of power consumption in interconnected wires in the slicing floorplan tree. The results presented show marginal improvements but with slight increase in layout area. The low power placement and routing problem of FPGA layout style under timing constraints is addressed in [32].

An automated layout synthesis system namely VPNR has been extended to incorporate the objective of reducing power consumption by decreasing switched capacitances. The modified system is named as EPNR. The authors used the classical physics notion of minimum energy systems and viewed the placement problem from the perspective of a system of interconnected springs. Consequently, EPNR worked by reducing wirelength of nets driven by the cells having high switching probability values [33]. The performance of EPNR was evaluated by using the MCNC benchmark circuits and results exhibited approximately 18% reduction in power consumption but not without an increase in the circuit area.

A couple of placement approaches based on Genetic algorithm have been reported. One such approach namely PCUBE is proposed in [34]. In this approach,

the placement problem is first formulated as a constrained programming problem and is then addressed in two subsequent steps named as *global optimization* and *slot assignment*. The total weighted net length is taken as an objective function where the weight of a net corresponds to the switching probability of the gates driving that net. The authors used a quadratic objective function in which the sum of the squares of power consumption values of each net is taken over all the nets to obtain the value of objective function. Then, they combined quadratic optimization with iterative circuit partitioning and proceeded by incorporating the partitioning information in each subsequent global quadratic optimization step. The iterative partitioning continues until they remained with 5 to 10 number of gates in each partition. It was assumed that each partition has at least as many number of slots as the number of cells and hence the problem reduced to the *linear assignment* problem. The circuit performance degradation is prevented throughout the above power optimization process by putting constraints on total path delays. The experimental results of PCUBE were compared with an older performance driven placement approach RITUAL [35] and an improvement of 7% in terms of power consumption was shown with an increase of 8% in total wirelength and an increase of 2% in circuit delay. The authors also considered the low-power placement problem under *real delay* model.

Another such technique namely GEEP has been proposed in [36]. Like PCUBE, it is also considering the standard cell placement problem. This approach differs

from the previous one in the sense that the potential search here is limited based on the information of the search space obtained from another approach namely EPNR that was described above. In this way, the search process is made efficient at the cost of global optimality. The encoding of the solution is drawn from [37] and the initial population is constructed by making some random perturbations of an area optimization solution generated by another placement package VPNR. The *fitness function* used in GEEP is based on *bounding-box* method that approximates the total interconnect power consumption in a circuit [38]. Furthermore, PMX crossover operator has been used in GEEP with a rate of 0.3 and the offsprings generated were accepted only if their fitness values were higher than one of their corresponding parents. Two types of mutation operators namely **blind mutation** and **directed mutation** were applied, and a reversal inversion operator similar to the one reported in [37] is used. The technique was evaluated by its application to some of the MCNC benchmark circuits and an improvement of 20% over VPNR was shown in terms of power consumption. However, nothing was mentioned about the effects of GEEP on the values of circuit area and delay. Moreover, the authors did not claim global optimality of their solution rather they were more interested in investigating the trade-off between solution quality and run time requirements of GA.

2.3 Techniques for Performance-Driven Placement

Circuit performance is determined by the maximum clock rate that can result in correct function of the circuit. However, the maximum clock rate is limited by the delay along the longest path in a circuit, called critical path. The critical path delay is determined by the maximum time needed by a signal to propagate along a path between an input and an output, or from an input to a storage element or from a storage element to an output or between two storage elements. With advancement in technology, the cell switching delays have been reduced and as a result the interconnect delays have become prominent contributors towards path delays. Hence, if the path delays are to be kept under control, then the interconnect delays need to be reduced. Since the placement phase has a direct relation with interconnect delays, the objective of the placement problem can be altered to satisfy the path timing requirements. Previously, when low power consumption in circuits was not realized as a design objective, the efforts targeting low delay (performance-driven) with less wirelength (area-driven) designs were dominating the literature.

The performance-driven placement techniques usually work by imposing timing constraints on the interconnects and paths of the circuit. We can classify these techniques into three types. In the first type, timing constraints on the paths are transformed into *weights on nets*. These weights are then used to classify the nets into certain categories and in this way the placement process is affected by net

weights [39]. In the second type, the path timing constraints are rather converted into *timing bounds on the nets.* The net timing bounds are then transformed into length bounds and provided as input to the placement process that attempts to keep the net lengths under these given length bounds [40, 41]. The last type is characterized by finding the timing requirements for a set of the critical paths in the circuit and supplying this information to the placement process which then monitors these paths throughout placement. [35, 42, 43, 44].

Chapter 3

Problem and Cost Function Formulation

3.1 Introduction

In this chapter, the multi-objective VLSI standard placement problem is formulated. The objective is simultaneous optimization of power consumption, timing performance and interconnect wire length, whereas layout width is considered as a constraint. The cost functions for the estimation of costs of the above three objectives as well as width constraint are modeled. Then, the overall cost function used for multi-objective optimization (MOP) is designed using fuzzy logic. This is preceded by an overview of fuzzy sets and rules, which is helpful for understanding the details of the presented fuzzy cost function.

3.2 Problem Statement

An informal description of the VLSI cell placement problem was given in Chapter 1. Formally, the problem can be stated as follows. Assume that a set of modules $M = \{m_1, m_2, ..., m_n\}$ and a set of signals $S = \{s_1, s_2, ..., s_k\}$ is given and a set of signals S_{m_i}, where $S_{m_i} \subseteq S$, is associated with each module $m_i \in M$. Similarly, a set of modules M_{s_j} (where $M_{s_j} = \{m_i | s_j \in S_{m_i}\}$ is called a signal net) is associated with each signal $s_j \in S$. Also, a set of locations $L = \{L_1, L_2, ..., L_p\}$, where $p \geq n$, is given. The problem is to assign each $m_i \in M$ to a unique location L_j, while optimizing an objective function subject to certain design constraints [2]. In the present problem, the objectives to be optimized are power consumption, timing performance, and wirelength, whereas the constraint is the width of the layout.

3.2.1 Assumptions

Here are some assumption that are made before proceeding to the design of cost functions and solution methodologies.

- The logic level design of the circuit is available.
- All the cells have predefined input and output terminals.
- The cells are interconnected in a predefined way.
- A set of critical paths and switching probabilities of cells are available.

- The circuit is represented in the form of a directed graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ and $E \subseteq V \times V$. Each gate of the circuit can be represented by a vertex v_i and the connection from the output of gate i to input of gate j can be represented by a directed edge (v_i, v_j) or simply (i, j) [33].

3.2.2 Inputs and Outputs

The solution methodologies presented in this thesis consider the following information as input.

- A net list describing the interconnections between the terminals of cells in the circuit in VPNR netlist format [45].
- The number of rows in the layout that is obtained from a min cut placement program.
- The height of each routing channel that lies between two adjacent rows.
- The delay parameters, width, and pin locations of cells obtained from a standard-cell library, namely OASIS SCMOS library [46].
- Timing information of the K most-critical paths obtained from a pre-placement timing analysis program.

The outputs produced are as follows.

- A cell placement solution consisting of the physical location of each cell.

- The quality measure of the final placement solution in terms of costs of power consumption, circuit delay and wirelength as well as layout width.

3.3 Cost Functions Modeling

This section discusses the modeling of cost functions used for estimating values of the three objectives as well as the constraint.

3.3.1 Cost Function for Interconnect wirelength

The wirelength cost can be computed by adding the wirelength estimates for all the nets in the circuit. The complexity of a wirelength estimation technique affects the CPU time requirement of a placement algorithm. There are several estimation techniques including semi-perimeter, complete graph, minimum chain, source to sink connection, minimum spanning tree, and Steiner tree approximation [2]. A Steiner tree is the shortest route for connecting a set of pins. However, determination of minimum Steiner tree is known to be NP-Complete [2]. Therefore an estimation of Steiner tree is made as follows.

For each net, a bounding rectangle is formed, as shown in Figure 3.1. Then, the dimensions of the bounding rectangle are compared. If $dx > dy$ then, the rectangle is partitioned into two parts by a horizontal line passing through the center of the rectangle, as shown in Figure 3.1(a). In this case, wirelength l_i associated with net

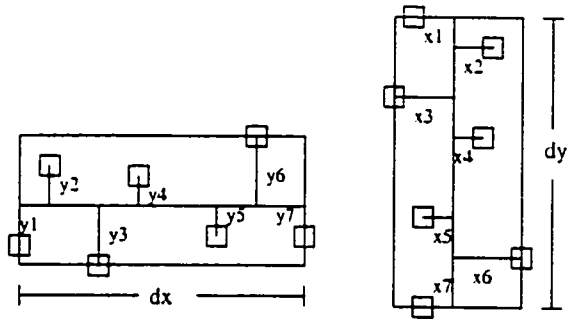


Figure 3.1: Steiner tree approximation to calculate the wirelength of a net.

v_i is given as follows.

$$l_i = dx + \sum_{j \in M_i} y_j \tag{3.1}$$

If $dy > dx$ then, the rectangle is partitioned into two parts by a vertical line passing through the center of the rectangle, as shown in Figure 3.1(b). In this case, wirelength l_i associated with net v_i is given as follows.

$$l_i = dy + \sum_{j \in M_i} x_j \tag{3.2}$$

The wirelength cost can be computed by adding the wirelength estimates for all the nets in the circuit, as shown in the following Equation.

$$Cost_{wire} = \sum_{i \in M} l_i \tag{3.3}$$

3.3.2 Cost Function for Power Consumption

In a standard CMOS circuit, the total power consumption can be given by the following Equation.

$$P_t = \sum_{i \in M} (\frac{1}{2} \cdot C_i \cdot V_{DD}^2 \cdot f \cdot S_i \cdot \beta) + \sum_{i \in V} Q_{SC_i} \cdot V_{DD} \cdot f \cdot S_i + I_{leak} \cdot V_{DD} \qquad (3.4)$$

In Equation 3.4, P_t is the total power consumption, V_{DD} is the supply voltage, S_i is the switching probability at the output node of cell i i.e., the number of transitions per clock cycle at the output of gate i, and f is the clock frequency.

The first term in the above equation gives the dynamic power consumption during charging or discharging of a node in the circuit. Here, C_i denotes the total capacitance of node i whereas β is a technology dependent constant.

The second term in Equation 3.4 gives the power consumption due to the short circuit current from V_{DD} to ground during output transition. Here, Q_{SC_i} represents the charge carried by the short circuit current per transition.

The third term represents the static power dissipation due to leakage current I_{leak}.

In the VLSI circuits with well designed logic gates, the dynamic power consumption contributes the 90% to the total power consumption [9, 47]. Hence, most of the reported work is focused on minimizing the dynamic power consumption. Also,

in the case of standard-cell placement, the cells are obtained from the technology library and nothing can be done to reduce the power consumption due to the second and the third term in Equation 3.4. Due to this fact, the emphasis in this work is on optimizing the dynamic power consumption. Since the first term is dominant, Equation 3.4 can be approximated as follows.

$$P_t \simeq \sum_{i \in M} \frac{1}{2} \cdot C_i \cdot V_{DD}^2 \cdot f \cdot S_i \cdot \beta \tag{3.5}$$

Assuming the clock frequency and input voltage to be fixed, the total power consumption of the circuit becomes a function of the total capacitance and the switching probabilities as shown below.

$$P_t \simeq \sum_{i \in M} C_i \cdot S_i \tag{3.6}$$

The total capacitance C_i of gate i is comprised of the interconnect capacitance at the output node of gate i and the sum of the capacitances of the input nodes of the gates driven by gate i.

$$C_i = C_i^r + \sum_{j \in M_i} C_j^g \tag{3.7}$$

where C_j^g is the capacitance of the input node of a gate j driven by gate i and C_i^r represents the interconnect capacitance at the output node of cell i.

In the case of standard-cell design, the cell properties are fixed for a particular library and hence the term C_j^g cannot be manipulated. Thus, the cost of the overall

power consumption in VLSI circuits can be given as follows.

$$Cost_{power} = \sum_{i \in M} S_i \cdot C_i^T \tag{3.8}$$

Further, the interconnect capacitance at a gate is related to the corresponding interconnect wirelength. Therefore, the above cost function can be written as follows.

$$Cost_{power} = \sum_{i \in M} S_i \cdot l_i \tag{3.9}$$

3.3.3 Cost Function for Delay

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 in VLSI circuit consists of two elements, switching delay of gates and interconnect delay. Due to improved technology, libraries with considerably low switching delay are available. This fact and the increased gate density in the chip make the interconnect delay the prominent factor in the overall circuit delay. Therefore, most of the work focuses on reducing the interconnect delay [48].

If a path π consist of nets $\{v_1, v_2, ..., v_n\}$, then, the delay T_π along π is expressed by the following Equation.

$$T_\pi = \sum_{i=1}^{n-1} (CD_i + ID_i) \tag{3.10}$$

Where CD_i is the switching delay of the cell driving gate v_i and ID_i is the interconnect delay of net v_i.

Since the term CD_i is independent of the placement, the above Equation can be simplified as shown below.

$$T_\pi = \sum_{i=1}^{n-1} (ID_i) \tag{3.11}$$

Using the RC delay model, ID_i depends on the load factor, interconnect resistance and load capacitance, as shown in Equation 3.12.

$$ID_i = (LF_i + R_i) \times C_i \tag{3.12}$$

Where LF_i is the load factor of the driving block (which is independent of layout), R_i is the interconnect resistance of net v_i, and C_i is the load capacitance of cell i given by Equation 3.7.

The overall circuit delay is determined by the delay along the longest path (the most critical path) in the layout. If the most critical path is denoted by π_c then, the cost function for the circuit delay can be given as follows.

$$Cost_{delay} = T_{\pi_c} = \max_j \{T_{\pi_j}\} \quad \forall j \in \{1, 2, \ldots, K\} \tag{3.13}$$

Where K represents the total number of critical paths determined by the timing analysis program.

3.3.4 Cost Function for Layout Width

In standard-cell design, cells have fixed height and variable widths. Cells are placed in rows separated by routing channels. The overall area of the layout is represented by the rectangle that bounds all the rows and routing channels. In this work, the channels heights are initially estimated using an area efficient placement tool and then assumed to be fixed. This leaves only the width of the layout that can effect the layout area. Since the available area for the placement is normally predefined, therefore the width of the layout is used as a constraint. The upper limit on the layout width is defined in Equation 3.14.

$$Width_{max} = (1 + \alpha) \times Width_{opt} \tag{3.14}$$

Where $Width_{max}$ is the maximum allowable width of the layout, $Width_{opt}$ is the minimum possible layout width obtained by adding the widths of all cells and dividing by number of rows in the layout, and α denotes the maximum allowed fractional increase in the layout width as compared to the optimal width.

3.4 Interconnect Capacitance and Resistance

Since the interconnect capacitance and resistance are to be used in the delay calculation, so it is necessary to compute these values. If two layers of metal are used for

routing in such a way that metal 1 is used for horizontal segments of the net and metal 2 for the vertical segments then, the following set of equations can be used to compute interconnect capacitance C_i^r associated with net v_i.

$$C_i^r = C_i^a + C_i^f \tag{3.15}$$

$$C_i^a = (C_{m1} \times l_1^i + C_{m2} \times l_2^i) \times \omega \tag{3.16}$$

$$C_i^f = 2 \times ((\omega + l_1^i) \times C_{f1} + (\omega + l_2^i) \times C_{f2}) \tag{3.17}$$

The following formula is used to calculate the interconnect resistance.

$$R_i^r = \frac{l_1^i \times R_{sh1} + l_2^i \times R_{sh2}}{\omega} \tag{3.18}$$

Where

C_i^r = Interconnect capacitance of net v_i.

C_i^a = Area capacitance of net v_i.

C_i^f = Fringe capacitance of net v_i.

C_{m1} = Plate capacitance per unit area of metal 1.

C_{m2} = Plate capacitance per unit area of metal 2.

C_{f1} = Fringe capacitance per unit perimeter length of metal 1.

C_{f2} = Fringe capacitance per unit perimeter length of metal 2.

R_{sh1} = Sheet resistance per square of metal 1.

R_{sh2} = Sheet resistance per square of metal 2.

ω = Width of metal 1 and metal 2.

l_1^i = Horizontal segment of l_i, associated with metal 1.

l_2^i = Vertical segment of l_i, associated with metal 2.

In Equations 3.15-3.17, ω, C_{m1}, C_{m2}, C_{f1} and C_{f2} are technology dependent parameters discussed in Section 3.7.

3.5 Fuzzy Logic

Fuzzy Logic is a mathematical tool invented to express human reasoning. In classical (crisp) reasoning a proposition is either true or false whereas in fuzzy system a proposition can be true or false with some degree.

A classical (crisp) set is normally defined as collection of elements or objects $x \in X$. Each single x element is either belong to the set X (true statement), or not belong to the set (false statement). Whereas a fuzzy set can be defined as follows.

$$A = \{(x, \mu_A(x)) | x \in X\}$$

$\mu_A(x)$ is called the membership function or grade of membership(or degree of truth) of x in A that maps X to the membership space M. The range of the

membership function is a subset of the non-negative real numbers whose supremum is finite [49]. Elements with zero degree of membership are normally not listed.

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 (denoted as $\oplus$). While fuzzy intersection operators are known as **t-norm** (denoted as *).

3.5.1 Fuzzy Reasoning

Fuzzy reasoning is a mathematical discipline to express human reasoning in vigorous mathematical notation. Unlike classical reasoning in which propositions are wither true or false, fuzzy logic establishes approximate truth value of propositions based on linguistic variables and inference rules [48]. A linguistic variable is a variable whose values are words or sentences in natural or artificial language [50]. For example, wirelength is a linguistic variable is its values are linguistic rather than numerical, i.e., very short, short, medium, long, very long and very long etc., rather than $20\mu m$, $25\mu m$, $35\mu m$, $45\mu m$, $55\mu m$ and $80\mu m$. The linguistic variables can be composed to form propositions using connectors like AND, OR and NOT. Formally, a linguistic variable comprises five elements [51].

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.

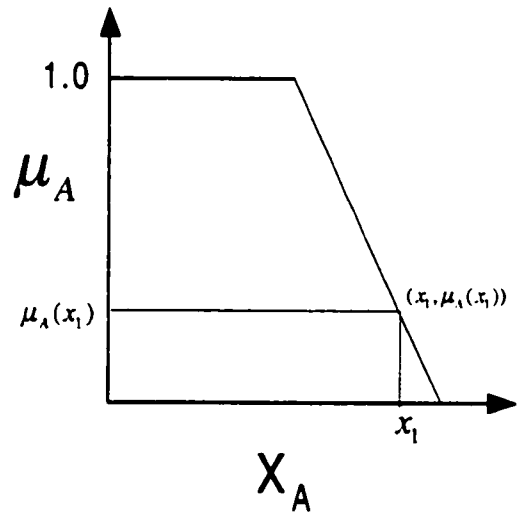


Figure 3.2: Membership function of a fuzzy set A.

For example wirelength can be used as linguistic variable for VLSI placement problem. According to the syntactical rule, the set of linguistic values of wirelength may be defined as very short, short, medium, long, very long and very long wirelength. The universe of discourse for linguistic variable is positive range of wirelength of a design, eg., [25μm, 80μm]. 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 3.2. It shows the designer knowledge of problem [48].

3.5.2 Fuzzy Operators

There are two basic types of fuzzy operators. The operators for the intersection, interpreted as the logical "and", and the operators for the union, interpreted as the logical "or" of fuzzy sets. The intersection operators are known as triangular norms (t-norms), and union operator as triangular co-norms (t-co-norms or s-norms) [49]. Some examples of s-norm operators are given below, (were A and B are the fuzzy sets of universe of discourse X).

1. Maximum. $[\mu_{A \bigcup B}(x) = max\{\mu_A(x), \mu_B(x)\}]$.

2. Algebric sum. $[\mu_{A \bigcup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)]$.

3. Bounded sum. $[\mu_{A \bigcup B}(x) = min(1, \mu_A(x) + \mu_B(x))]$.

4. Drastic sum. $[\mu_{A \bigcup B}(x) = \mu_A(x)$ 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 $\mu_{A \bigcup 0}(x) = \mu_A(x)$ properties. Following are some examples of t-norm operators.

1. Minimum. $[\mu_{A \bigcap B}(x) = min\{\mu_A(x), \mu_B(x)\}]$.

2. Algebraic product. $[\mu_{A \bigcap B}(x) = \mu_A(x)\mu_B(x)]$.

3. Bounded product. $[\mu_{A \bigcap B}(x) = max(0, \mu_A(x) + \mu_B(x) - 1)]$.

4. Drastic product. $[\mu_{A \bigcap B}(x) = \mu_A(x)$ if $\mu_B(x) = 1$, $\mu_B(x)$ if $\mu_A(x) = 1$, 0 if $\mu_A(x), \mu_B(x) < 1]$.

Like s-norm, t-norms also satisfy commutativity, monotonicity, associativity and $\mu_{A \bigcap 1}(x) = \mu_A(x)$. Also, the fuzzy complementation operator is defined as follows.

$$\bar{\mu}_B(x) = 1 - \mu_B(x) \tag{3.19}$$

3.5.3 Ordered Weighted Averaging (OWA) Operator

Generally, the formulation of multi criterion decision functions neither desires the 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 [52, 53]. This operator allows easy adjustment of the degree of "anding" and "oring" embedded in the aggregation. According to [52, 53], "orlike" and "andlike" OWA for two fuzzy sets A and B are

implemented as given in Equations 3.20 and 3.21 respectively.

$$\mu_{A\cup B}(x) = \beta \times \max(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B) \tag{3.20}$$

$$\mu_{A\cap B}(x) = \beta \times \min(\mu_A, \mu_B) + (1-\beta) \times \frac{1}{2}(\mu_A + \mu_B) \tag{3.21}$$

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.

To solve an MOP using fuzzy logic, the problem is first defined in linguistic terms then the membership of different fuzzy sets is combined using t-norm or s-norm operator (depends upon problem). Then the resulting membership is used in minimization or maximization problem.

3.6 Fuzzy Cost Function for VLSI standard-cell Placement Problem

In this method, it is assumed that there are Γ Pareto-optimal solutions. Also a p-valued cost vector $C(x) = (C_1(x), C_1(x), ..., C_p(x))$, where $x \in \Gamma$ is given. There is a vector $O = (O_1, O_2, ..., O_p)$ that gives the lower bounds on the cost for each objective such that $O_j \leq C_j(x)\ \forall j$, and $\forall x \in \Gamma$. These lower bounds are normally not reachable in practice. There is another user defined goal vector $G = (g_1, g_2, ..., g_p)$ that represents the relative acceptance limits for each objective. It means that x is

an acceptable solution if $C_j(x) \leq g_j \times O_j$, $\forall j$ where $g_j \geq 1.0$. For two dimension problem, Figure 3.3 shows the region of acceptable solution.

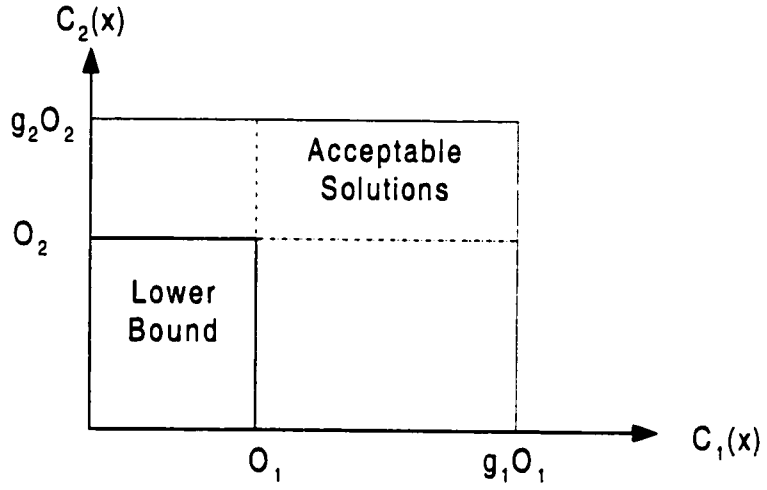


Figure 3.3: Range of acceptable solution set.

In order to solve multiobjective placement problem, three linguistic variables wirelength, power dissipation, and delay are defined. The following fuzzy rule is used to combine the conflicting objectives.

Rule R1:

IF a solution has
small wirelength
AND
low power dissipation
AND
short delay
THEN it is a good solution.

The above mentioned linguistic variables are mapped to the membership values

in fuzzy sets *small wirelength, low power dissipation,* and *short delay* respectively. These membership values are computed using the fuzzy membership functions shown in Figure 3.4.

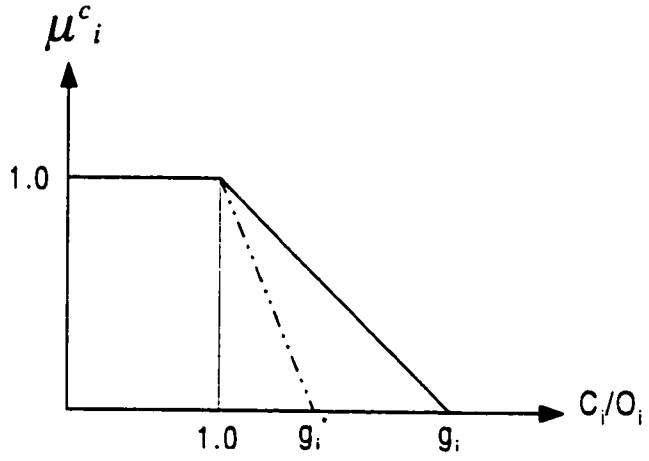


Figure 3.4: Membership functions within acceptable range.

As layout width is a constraint, therefore if a solution violates this constraint, it is not a valid solution and is hence discarded. However, for the objectives, by increasing and decreasing the value of g_i, its preference can be varied in combined membership function. The lower bounds O_j (shown in Figure 3.4) for different objectives are computed as given in Equations 3.22-3.25.

$$O_l = \sum_{i=1}^{n} l_i^* \qquad \forall v_i \in \{v_1, v_2, ..., v_n\} \tag{3.22}$$

$$O_p = \sum_{i=1}^{n} S_i l_i^* \qquad \forall v_i \in \{v_1, v_2, ..., v_n\} \tag{3.23}$$

$$O_d = \sum_{j=1}^{k} CD_j + ID_j^* \quad \forall v_j \in \{v_1, v_2, ..., v_k\} \text{ in path } \pi_c \tag{3.24}$$

$$O_{width} = \frac{\sum_{i=1}^{n} Width_i}{\#\ of\ rows\ in\ layout} \tag{3.25}$$

where O_j for $j \in \{l, p, d, width\}$ are the lower bounds on the costs for wirelength, power dissipation, delay and layout width respectively, n is the number of nets in layout, l_i^* is the lower bound on wirelength of net v_i, CD_i is the switching delay of the cell i driving net v_i, ID_i^* is the lower bound on interconnect delay of net v_i calculated with the help of l_i^*, S_i is the switching probability of net v_i, π_c is the most critical path with respect to optimal interconnect delays, k is the number of nets in π_c and $Width_i$ is the width of the individual cell driving net v_i.

Using Equation 3.21 and minimum operator, rule **R1** is interpreted as follows:

$$\mu(x) = \beta \times min(\mu_p(x), \mu_d(x), \mu_l(x)) + (1 - \beta) \times \frac{1}{3} \sum_{j=p,d,l} \mu_j(x) \tag{3.26}$$

where $\mu(x)$ is the membership of solution x in fuzzy set of acceptable solutions, whereas $\mu_j(x)$ for $j = p, d, l$, are the membership values in the fuzzy sets *within acceptable power, within acceptable delay, and within acceptable wirelength* respectively. β is the constant in the range $[0, 1]$. In this thesis, $\mu(x)$ is used as the aggregating function. The solution that results in maximum value of $\mu(x)$ is reported as the best solution found by the search heuristic.

Metal Type	ω (μm)	R_{sh} $\Omega/\square$	C_p aF/μ^2	C_f aF/μ
Metal 1	0.36	0.07	39	26
Metal 2	0.36	0.07	19	60

Table 3.1: 0.25 μ technology parameters.

3.7 Technology Parameters

In this work, all the cell are taken from the 0.25 μ MOSIS TSMC CMOS technology library. The cell parameters like the cell delay, the input capacitance, and the cell width are given in [46].

It is assumed that two layers of metal routing are used. Metal 1 is used in the routing of horizontal and metal 2 in the routing of vertical tracks. The values of capacitances and resistances for these two metal layers are given in Table 3.1.

Chapter 4

Iterative Algorithms for VLSI standard-cell Placement

4.1 Introduction

Genetic Algorithm was proposed by Holland in 1975 [54], whereas Tabu Search (TS) [55] was presented by Fred Glover. After their introduction, these have found applications in optimization problems from various areas of science and engineering. There has been many efforts involving application of GA to the VLSI placement problem. Earliest application of GA for the placement problem targeted the objective of minimizing interconnect wirelength only [7]. Also, the application of TS to the Quadratic Assignment Problem (QAP), which is a generalization of the placement problem is reported in [56].

However, with the advancement in technology, standard-cell libraries with smaller cell delays and smaller input pin capacitances became available. As a consequence, interconnect capacitance turned into prominent part in the computation of overall circuit timing and power consumption. A number of GA based techniques for timing-driven placement were presented [2, 44]. A brief review of some of the reported timing-driven placement techniques is given in Chapter 2.

Recently, for obvious reasons, power consumption in VLSI circuits appeared as an important design objective. The present work addresses this issue by developing iterative algorithms for low-power and timing-driven VLSI standard-cell placement. This chapter discusses the implementation details of Genetic algorithm and Tabu search for the above problem.

4.2 GA for Timing and Low Power Driven VLSI Cell Placement

Genetic algorithms belong to the class of general iterative heuristics that may be applicable to a variety of optimization problems. For employing GA to a certain problem, it needs to be tailored according to the specific characteristics of the underlying problem. Some significant modifications in the implementation of simple GA (as shown in Figure 1.3) are suggested for engineering it to the present multi-objective VLSI standard-cell placement problem. For instance, it is suggested to

use *mutation* operator after *selection for the next generation.* The implementation details of the GA steps are described in following sub-sections.

4.2.1 Chromosome Encoding and Initial Solution

For a solution to be processed by GA, it is required to be represented in form of a string. A placement solution is an arrangement of cells in two dimensional physical layout surface. For instance, consider a circuit comprising of 11 cells $\{1, 2, 3, \ldots, 11\}$. A possible layout is as shown below.

row 1: 3 5 8 6

row 2: 9 10

row 3: 7 11 1

row 4: 4 2

The above layout is generated after computing the average row width. Then, average row width is divided by the smallest cell width to compute the maximum possible number of slots in a row. Assume there are four slots and it is known from the input information that there are four rows in layout. The initial solution is constructed by randomly selecting a cell from eleven cells and placing it in the first row. Before placing a cell, it is checked whether adding it will violate the width constraint, and if so, it is placed at the start of next row and so on. In the above example, it is assumed that sum of widths of cells 3, 5, 8, 6 is within width constraint,

but adding cell 9 was violating it, so cell 9 is placed in second row. Similarly, all cells are placed on the layout. As a result, five slots remain empty, two in second row, one in third row, and two in last row.

To form a chromosome, it is proposed to concatenate all the rows of a solution. After concatenation, the chromosome that is to be used for the application of GA operators becomes as shown below.

3 5 8 6 9 10 7 11 1 4 2

In each iteration, the above chromosome needs to be mapped back to a two dimensional layout in order to compute the fitness value. There should be some mechanism for this mapping. Due to varying widths of the cells in a circuit, all rows may not have equal number of cells in them as can be observed in the layout example shown above. This fact introduces a problem in mapping of chromosome to the layout that at which points the above chromosome should be cut. A simple solution could be to follow the same process that was adopted above for creating initial solution i.e., start from the left of chromosome and keep on placing the encountered cells in first row until the width constraint is violated, then move to second row and so on. But this scheme enforces a deterministic width cost throughout the search process and may restrict visiting certain points in search space.

A better approach to deal with this problem is the introduction of dummy cells. Assuming there are N cells in the circuit while L slots in 2-D layout, where $L > N$, $L - N$ dummy cells represented by distinct negative integers are inserted to fill the

empty slots of 2-D layout. The reason for choosing distinct negative integers will become clear when crossover operation will be explained. After insertion of dummy cells, the resulting layout is as shown below.

row 1: 3 5 8 6

row 2: 9 10 -1 -2

row 3: 7 11 1 -3

row 4: 4 2 -4 -5

The chromosome representation corresponding to the above layout is

3 5 8 6 9 10 -1 -2 7 11 1 -3 4 2 -4 -5

In addition to random initial solution, constructive initial solution was also investigated. The latter was generated by greedy pair wise interchange approach. The comparison of two approaches in terms of the quality of final solution is presented in Chapter 6.

The results obtained from using GA depend heavily on the population size i.e., number of chromosomes in the population. Therefore, Experiments are carried out using various population sizes.

4.2.2 Fitness Evaluation

For addressing a multi-objective optimization problem to minimize three mutually conflicting objectives, a measure is needed which can quantify the overall quality of a

solution with respect to all three objectives collectively. Fuzzy membership functions and fuzzy rules are used for evaluating the fitness of a solution. A fitness value between 0 and 1 is assigned to each solution. The fitness value of a chromosome is its membership value $\mu(x)$ in the fuzzy set of acceptable solution. This membership is computed using Equation 3.26. Moreover, if a solution violates width constraint, it is not considered a valid one and is discarded.

The fitness of a solution is a measure of its proximity to the optimal solution. The higher the fitness value of a solution, the closer is it to the optimal solution. In the present implementation, initial random solution is assigned a membership value of 0 and the optimal solution is assigned a fitness value of 1. This implies that any solution may have a fitness value in range [0.0,1.0).

For evaluation of cost, the chromosome is translated into the placement structure that maps each cell to its corresponding ordered pair (X,Y). Here X represents the actual horizontal location of left edge of the cell whereas Y represents the index of the row in which the cell lies. The dummy cells introduced in chromosome representation are used primarily for facilitating encoding as well as for the appropriate application of genetic operators like crossover. They are ignored in the above translation process and hence do not contribute in cost computation of the solution.

4.2.3 Parent Choice

In each generation of GA, certain number of off-springs are created. Each offspring is the result of mating two chromosomes called parents. There are many different schemes including *roulette wheel*, and *ranking* choice of parents [1]. The former scheme was employed in the present work as it is the most widely reported scheme in the literature. It is based on the idea of *stochastic sampling with replacement.* In this scheme, an individual chromosome is selected with a probability that is proportional to its fitness value. The probability $P_{choice}(x)$ of choosing a chromosome x can be given as

$$P_{choice}(x) = \frac{\mu(x)}{\sum_{i=1}^{N_p} \mu(i)} \tag{4.1}$$

where N_p denotes population size. Although using this scheme, there is higher chance for fitter chromosomes to be chosen, but it also allows the individuals having low fitness values to be selected with a non-zero probability.

The motivation for using such scheme that favors the choice of fitter chromosomes over the weaker ones is following. The mating of two fitter chromosomes is more likely to reproduce fitter chromosomes than mating of two weaker ones. However, the choice of fitter individuals only is a greedy approach that may lead the algorithm to a local optimal. Therefore roulette wheel choice also permits the weaker chromosomes

to take part in crossover. This property ensures the diversity in population, that is known to be essential for healthy progress of GA towards better points in search space.

In addition to roulette wheel, random choice scheme was experimented for comparison purposes. In this scheme, parents are chosen randomly from the current population irrespective to their fitness values. The effect of these two schemes is discussed in Chapter 6.

At start of each generation, N_p number of parent pairs are chosen for crossover. As a result N_p offsprings are created.

4.2.4 Crossover

Crossover is the operator that causes inheritance of characteristics from one generation to the next. Many different types of crossover including are one-point or two-point simple crossover, order crossover, and partially mapped crossover (PMX) are reported in the literature [1]. In the present problem, each gene in the chromosome representation is distinct and this property must be preserved from generation to generation for a chromosome to represent a valid solution. Therefore, simple crossover operators can not be used as these may result in duplicate genes.

In this work, different types of existing crossover operator like Order crossover and Partially Mapped Crossover (PMX) are experimented. An example of PMX working is shown in Figure 4.1. The detailed description of PMX can be found

in [1]. Figure 4.2 illustrates the problem that may occur if the distinct negative integers are not used for representing the dummy cells in the layout.

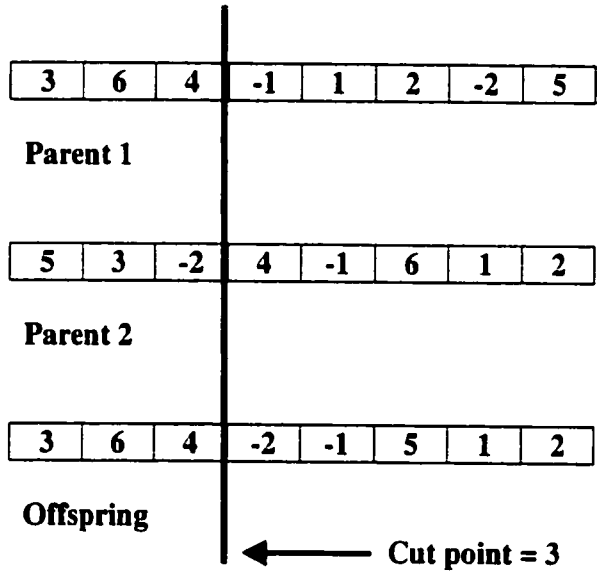


Figure 4.1: An example of PMX operation.

In addition, a modified form of PMX application is investigated, which is named Controlled Dual PMX (CDX) and it works as follows. Since PMX is not a symmetric operator, first PMX is applied between parent-1 and parent-2 and then it is applied between parent-2 and parent-1. As a result, two offsprings are generated. The better of two offsprings is chosen with a certain high probability P_{cdx} and any one of two is chosen randomly with probability $1 - P_{cdx}$.

The crossover operation is generally performed with a high probability p_c. Different high crossover probabilities are used. The implementation of crossover probability is done as follows. After choosing two parents, a random number *rand* in

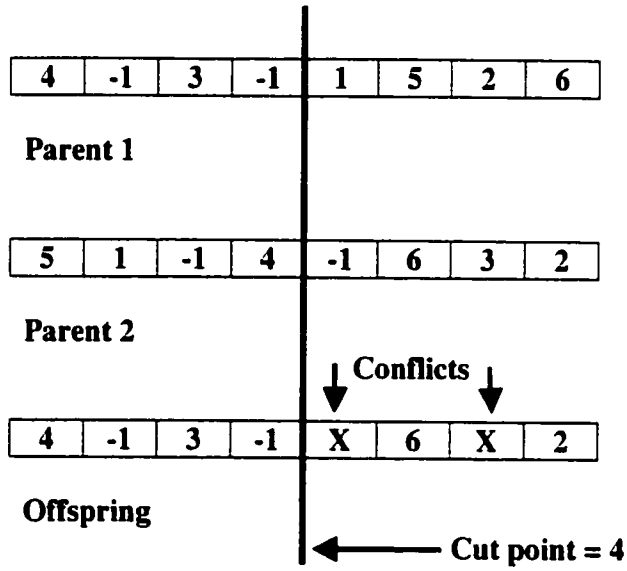


Figure 4.2: An example of possible problems in using PMX when dummy cells are non-distinct negative integers.

range [0, 1] is generated, and if $rand < p_c$, crossover is applied; otherwise, other two parents are chosen and another random number is regenerated and so on. In this way, it is ensured that the same number of offsprings are generated in each iteration. The number of offsprings generated are equal to the population size. Each time when a new offspring is generated, it is checked whether its width cost is violating the width constraint. If it is so, then the offspring is discarded and another crossover is performed after choosing other parents. This process is repeated until the desired number of offsprings are generated.

Another phenomenon is observed that sometimes an offspring resulting from crossover of two dissimilar parents is exactly similar to one of the parents. This

leads to duplicate chromosomes in the population which decreases diversity of the population. This phenomenon is not desired in GA because it offers hindrance to search process in moving to the un-explored regions of the search space. Therefore this phenomenon is restricted by discarding any such duplicate offsprings. Some other measures for maintaining diversity in the population are also proposed as described below in the following sub-sections.

4.2.5 Selection

A modification in simple GA is suggested by moving *selection for the next generation* before *mutation*. The purpose is to encourage the diversity in the population by ensuring the transfer of mutation effect into next generation. The mutation operator introduces new characteristics in the population that help in preventing the algorithm from trapping in local minima. Therefore, it is important to preserve the effect of mutation with generations.

In simple GA, selection is performed after mutation. In this case, it may happen that the mutated chromosomes are not selected for the next generation and thus the effect of mutation is lost. While in the presented approach, all the selected chromosomes are subjected to mutation.

Experiments are carried out using following seven different selection schemes.

- *roulette wheel selection* (rlt): Chromosomes for the next generation are selected

with probabilities proportional to their fitness value. In this scheme, chromosomes having lower fitness values may also propagate with small probability to the next generation.

- *random selection* (rnd): All chromosomes for the next generation are selected randomly. The average fitness of chromosomes in the next generation is unpredictable.

- *elitist-roulette selection* (erlt): The best chromosome from current population is selected and then remaining $N_p - 1$ chromosomes are selected using roulette wheel. Using this approach, the global best over generations is always remembered.

- *extended-elitist-roulette selection* (eerlt): $\frac{N_p}{2}$ best chromosomes are selected and remaining $\frac{N_p}{2}$ are selected proportional to their fitness. This scheme as well as the previous one may lead the search process to local minima, because weaker individuals are not selected and thus diversity in population is lost.

- *elitist-random selection* (ernd): The best chromosome is selected from parents and offsprings, whereas the remaining $N_p - 1$ chromosomes are selected randomly.

- *extended-elitist-random selection* (eernd): $\frac{N_p}{2}$ best chromosomes are selected and remaining $\frac{N_p}{2}$ are selected randomly. This scheme and the last one offer

balance between greediness and randomness.

- *elitist-pareto-random selection* (eprnd): The best chromosome is selected, then one best with respect to each of the objectives is selected, and remaining $N_p - 3$ are selected randomly.

4.2.6 Mutation

In this work, it is suggested to apply mutation with a dynamic probability P_μ^k, which is a function of the diversity of the population. The standard deviation $\sigma_{fit}^{(k+1)}$ of the population in $(k+1)^{th}$ generation is taken as a measure of the diversity. The idea is to increase the mutation probability when population tends to lose diversity. The mutation probability P_μ^k is varied in a range $[P_{\mu_{min}}, P_{\mu_{max}}]$. This is another effort to enhance diversity and thus GA search capability. The following set of equations specifies how mutation probability is updated dynamically.

$$P_\mu^k = P_{\mu_{min}} \quad if \quad \sigma_{fit}^k < 0.02 \tag{4.2}$$

$$P_\mu^k = P_{\mu_{max}} \quad if \quad \sigma_{fit}^k > 0.05 \tag{4.3}$$

$$P_\mu^k = 0.05 - \frac{2}{3}(\sigma_{fit}^k - 0.02) \quad otherwise \tag{4.4}$$

For each chromosome in the population selected for the next generation, a ran-

dom number *rand* in range [0, 1] is generated, and mutation is applied if $rand < P_{\mu}^{k}$. The mutation operation is implemented as a series of random pair wise interchanges of randomly chosen cells. The restriction on the selected pair for swap is that both of them can not be dummy cells at the same time. The reason for this restriction is that swapping of two dummy cells has no realization in circuit layout and therefore this swap does not introduce any new characteristics in the solution.

The number of interchanges to be performed is a function of the size of circuit i.e., the number of cells in the circuit. A random fraction f between 0.03 and 0.05 is generated and $f \times N$ interchanges are made, where N is the total number of cells in the circuit.

For the comparison purpose, fixed mutation probability is also implemented. The results of both schemes are discussed in Chapter 6.

4.2.7 Stopping Criterion

The experiments are carried out for various number of total generations. It is observed that after certain number of generations, GA tends to convergence and there is no further significant improvement in the solution quality. As a result, it is decided to use a fixed number of generations as a stopping criterion for GA.

4.3 TS for Timing and Low Power Driven VLSI Cell Placement

In this section, the implementation details of Tabu Search (TS) are described. TS is a powerful search heuristic. In contrast with GA, TS uses a single solution and tries to optimize it with iterations. The unique feature of TS is its *memory element*, that is used to record some characteristics of a certain number of previous moves. This feature is realized by the use of *tabu list*. The number of moves whose characteristics can be recorded depends on the size of this list. This feature prevents the search process from cycling (i.e., revisiting a point) in the search space. The details of various steps are presented in the following sub-sections.

4.3.1 Initialization and Cost Evaluation

The solution encoding and initialization steps are similar to those described above for the GA implementation. The only difference is that a single random initial solution is created.

4.3.2 Neighborhood Generation

In each iteration, a number of neighbors of the current solution are generated by making perturbations as follows. Two cells are selected randomly with a restriction that both of them are not dummy cells at the same time. Then the locations of

selected cells are mutually interchanged.

The quality of the final solution generated by TS depends on the number of neighbor solutions generated. Too small number of neighbor solutions prevents TS from searching the solution space effectively. On the other hand, too large number of neighbor solutions costs excessive run time without providing a significant improvement in the quality of solution. Therefore, the number of neighbor solutions to be generated in each iteration, needs to be chosen after a careful observation of the underlying problem's nature and size. In this work, the number of neighbor solutions is dependent on the problem size, i.e., the number of cells in the circuit. The effect of neighborhood size on the solution quality is discussed in Chapter 6.

4.3.3 Tabu List and Aspiration Criterion

Tabu list introduces memory element in search process. Its purpose is to avoid revisiting a point (solution) in the search space. This is implemented by storing some characteristic of a certain number of previously accepted moves.

Implementation of tabu list requires two decisions to be made. First, what characteristic of the move should be stored in tabu list. This decision has a significant effect on search quality as well as memory requirements of TS. The chosen characteristic should identify the move, so that it can be accurately used to restrict the corresponding move and to consequently fulfill the purpose of tabu list. In the present implementation, the characteristic of the move stored in tabu list is the in-

dex of any one of the two cells involved in mutual interchange. In case one of the cells is a dummy cell, the index of other cell is stored. The reason is that the move of a dummy cell in the solution representation has no realization in circuit layout. Therefore, it should not be restricted to take part in future perturbations by storing its index in tabu list.

The second decision that needs to be made is regarding the size of tabu list. This decision affects time and space requirements of TS. Various sizes of tabu list were investigated and consequently a reasonable fixed size is decided.

The aspiration criterion used is the following. If the best neighbor solution of the current iteration is better than the global best solution, then tabu list restrictions are overridden and the solution is accepted. Also, the global best solution and the aspiration criterion are updated.

4.3.4 Stopping Criterion

TS was run for a different number of total iterations, and depending on experimental observations, it is decided to run the algorithm for a fixed number of iterations as is done in case of GA. The major reason for stopping after a fixed number of iterations in both cases is that continuing beyond it costs run time without any significant improvement in the results.

Chapter 5

GATS: A Hybrid Algorithm for VLSI standard-cell Placement

5.1 Introduction

This chapter presents a novel hybrid algorithm and describes its implementation details. Hybrid algorithms combine features from various heuristics as an effort to develop efficient techniques for solving hard optimization problems. A brief introduction to hybridization and hybrid algorithms is presented in the first section of this chapter. The following section presents the algorithm named GATS, which is a hybrid of GA and TS.

5.2 Hybridization

Hybridization involves combining of features from more than one algorithm with a view of developing better techniques for solving hard problems [1]. In the scope of the present work, hybridization refers to mixing of good characteristics from different iterative algorithms. There are several iterative heuristics including GA, TS, and Simulated Evolution (SE). The details of the first two are given in Chapter 1, whereas the details of SE and some other iterative heuristics can be found in [1].

Any of these iterative heuristic may be considered for hybridization purpose. The choice of iterative algorithms to be hybridized needs some careful analysis of the underlying problem and these individual algorithms. The application of an individual algorithm to the problem may give an insight of its suitability to the problem. If the results of application of the individual algorithms are encouraging, then their hybridization is also likely to produce the better performance.

5.3 GATS: Hybrid of GA and TS

In this work, GA and TS are used for addressing the VLSI standard cell placement problem. Both of these algorithms exhibited good performance for the present problem. Especially TS proved to be an excellent technique. Therefore, it seems reasonable to hybridize GA and TS with the view of developing an efficient hybrid technique for timing and low power driven placement.

A good property of GA is its implicit parallel nature that helps in exploring the search space efficiently. This parallelism is due to the fact that GA processes a population of solutions instead of a single solution. On the other hand, TS showed better results than GA, and this better performance can be attributed to TS searching mechanism. This means that a good hybrid technique can be developed by combining the features from these two iterative algorithms.

Figure 5.1 illustrates the proposed hybrid algorithm named GATS. An interesting novel idea is the introduction of a population of solutions instead of single solution in TS. This is likely to enhance the power of TS by allowing it to visit the search space in a parallel fashion. The algorithm starts by taking a random initial population of solutions. Then, for each individual in the population, a certain number of neighbor solutions are generated and the best neighbor is found. A characteristic of the move leading to the best neighbor solution is stored in a tabu list. There are as many tabu lists as the number of solutions in the population i.e., N_T. The reason for taking N_T tabu lists is obvious that the series of moves for each individual in the population is different. Therefore, each series should be stored in a separate list so that a tabu list restricts the cyclic moves on its corresponding individual only. However, the aspiration level (AL) is unique for all the individuals. The purpose is that the tabu move on an individual solution is allowed only if it results in a solution that is better than an overall unique best solution.

The above process continues for a certain number of iterations and a record

```
Algorithm GATS
S    :  Current solutions
S*   :  Best solutions
N(S) :  Neighborhood of S ∈ Ω
V*   :  Sample of neighborhood solutions
AL   :  Aspiration levels
N_G  :  Population size for GA portion
N_T  :  Population size for TS portion
N_o  :  Number of Offsprings

Begin
For fixed number of times Do
Start with random initial population N_T
    For fixed number of iterations Do
      For j = 1 To N_T
        Generate neighbor solutions V* (j) ⊂ N (S (j)) by random swap
        Find best S* (j) ∈ V* (j)
        If move S (j) to S* (j) is not in T (j) Then
            Accept move and update T (j)
        Else
            If Cost (S* (j)) < AL (j) Then
                Accept move and update T (j) and AL
            End If
        End If
      End For
    End For
    Pass best or current solutions to GA
    For fixed number of generations Do
        For j = 1 To N_o
            (x, y) ← Choose parents
            Offspring [j] ← Crossover (x, y)
            Evaluate Fitness (offspring [j])
        End For
        Population ← Select (Population, offspring, N_G)
        For j = 1 To N_G
            Apply Mutation (chromosome [j])
            Evaluate Fitness (chromosome [j])
        End For
    End For
    Pass best or current solutions to TS
End For
End.
```

Figure 5.1: GATS: A Proposed Hybrid of Genetic Algorithm (GA) and Tabu Search (TS) for Timing and Low Power Driven VLSI standard-cell Placement.

is kept of the N_T individual best solutions obtained from perturbing N_T individual initial solutions. Then either these best solutions or the current solutions are passed to GA for further optimization. These semi-optimized individuals are likely to produce good offsprings by mating with one another. Now, GA is run for a given number of generations on these passed solutions and a record of the best individuals is kept. Again, either the current individuals or best ones are passed back to TS. The switching between TS and GA is repeated for a given number of times.

5.3.1 Parameters of GATS

There are several performance parameters to be tuned. These parameters can affect the performance of GATS significantly. For instance, a parameter is whether to pass best or current individual solutions between GA and TS portions at the time of alternate switching.

Some possible parameters are listed below.

- The cumulative number of iterations of GATS to be run and the number of times the switching between TS and GA is made.
- Should the number of iterations for TS and the number of generations for GA be equal. If not, then what should be the proportion.
- Should the population size in case of TS and GA be same i.e., $N_T = N_G$. If not, then what should be the ratio among these.

The above parameters can be tuned experimentally to achieve the best performance from GATS. The different settings of these parameters and their effect is discussed in Chapter 6.

Chapter 6

Experiments and Results

6.1 Introduction

This chapter presents the experimental results of different iterative heuristics presented in this thesis. A comparison of results obtained from different techniques is made in the terms of the quality of the final solution as well as the run time. The detailed study of the effects of different important parameters of each heuristic is conducted.

The organization of this chapter is as follows. Section 6.2 describes the details of ISCAS 85/89 circuits, because these are used as benchmarks for evaluating the performance of the proposed algorithms. In Section 6.3, the performance of the proposed algorithms is compared. Then, Section 6.4 analyzes the sensitivity of iterative algorithms to setting of certain parameters like crossover probability and

neighborhood size. In Section 6.6, an interesting comparison between single objective optimization and multiobjective optimization is reported.

6.2 Circuits Details

Eleven ISCAS-85/89 benchmark circuits are used. Most of the considered circuits are sequential because, in real life, almost all the circuits used are sequential in nature. Also, sequential circuits have more severe delay problems than combinational circuits, as most of the combinational circuits are well structured. The key characteristics of these circuits are given in Table 6.1. The number of rows in layout and the average channel heights are estimated using a min-cut based layout placer. The details of the cell characteristics are obtained from the 0.25 μ MOSIS TSMC CMOS technology library [46].

6.3 Comparison of Proposed Techniques

The comparison is divided in two sub-section. First, a comparison between GA and TS is presented and then the performance of the hybrid approach is compared with that of TS approach.

Circuit		Layout	
Name	Cells	Rows	Avg. Channel Height (μm)
s2081	122	4	6.66
s298	136	5	7.08
s386	172	5	7.68
s641	433	7	9.72
s832	310	7	9.78
s953	440	8	11.76
s1196	561	9	11.58
s1238	540	9	11.64
s1488	667	11	10.92
s1494	661	11	10.56
c3540	1753	16	13.68

Table 6.1: Circuit and layout details.

6.3.1 GA versus TS

The results obtained from GA and TS are compared in terms of the overall quality of the best solution and run time in Table 6.2. In this table, L represents the wire length in μm, P represents the cost due to power, which is the sum of the product of switching probabilities and wirelength of all the nets, D represents the delay of the most critical path in pico seconds (ps), $\mu(x)$ is the membership value, and T is the execution time in seconds for reaching the best solution. In case of TS, 5,000 iterations are run, whereas for GA, the stopping criterion is 10,000 generations. The width constraint is satisfied in obtaining all the results shown here.

The results shown are the best case results obtained by the best possible tuning of the various algorithmic parameters of GA and TS. In case of GA, the population

	GA					TS				
Circuit	L (μm)	P	D (ps)	$\mu(x)$	T(s)	L (μm)	P	D (ps)	$\mu(x)$	T(s)
s2081	2426	388	113	0.785	2341	2323	379	111	0.823	298
s298	4062	838	130	0.775	2922	3579	635	127	0.821	212
s386	6824	1665	193	0.695	3945	6643	1595	190	0.709	524
s641	17812	4532	740	0.685	21982	12620	2868	656	0.800	1505
s832	21015	4787	395	0.598	7206	18760	4311	349	0.658	981
s953	31004	5027	235	0.606	11221	27287	4230	214	0.669	1036
s1196	48729	14755	372	0.543	16120	39054	11700	332	0.650	1138
s1238	50387	15035	396	0.536	16208	39186	11594	353	0.656	1124
s1488	69792	17346	784	0.518	21434	56888	13867	662	0.621	2256
s1494	69223	17169	771	0.540	26032	54710	13533	674	0.650	2499
c3540	310996	109850	924	0.425	57724	164581	58143	699	0.708	19215

Table 6.2: Comparison between costs of the best solutions generated by GA and TS.

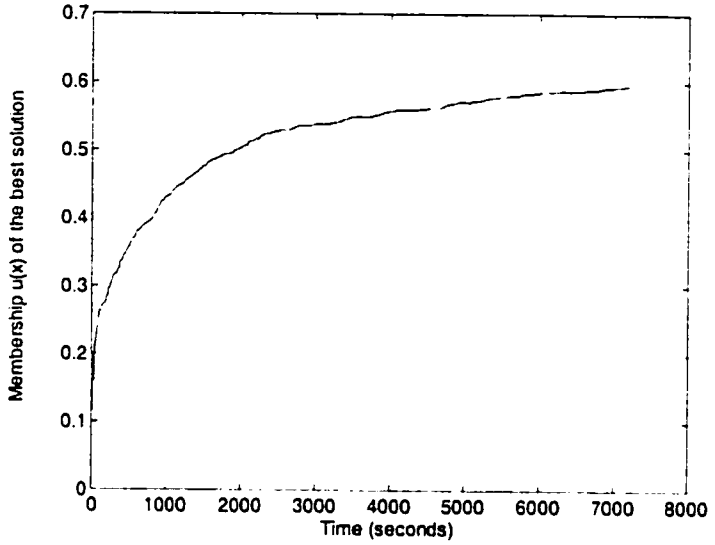
size is 32, and the crossover used is CDX with a probability equal to 0.99. The value of parameter P_{cdx} is set equal to 0.95. It was discussed in Chapter 4 that P_{cdx} determines the probability with which the better of the two offsprings is selected. The selection scheme used is *eernd*. Dynamic mutation probability is used in most of the cases except for circuits s641 and s1196, for which the shown results are for fixed mutation probability. In case of TS, the size of neighborhood is varied from 24 solutions for s2081 to 70 solutions for c3540, while the tabu list size is 100. The effect of using other settings in GA and TS is discussed later in this chapter.

From the results, it is clear that TS performed better than GA for all the circuits in terms of the quality of the best solution as well as the CPU run time. The significant observation is that in case of smaller circuits like s2081, s298, and s386, the quality of the final solutions obtained from GA is comparable with that obtained from TS. But as the size of circuit increases i.e., when the size of search space

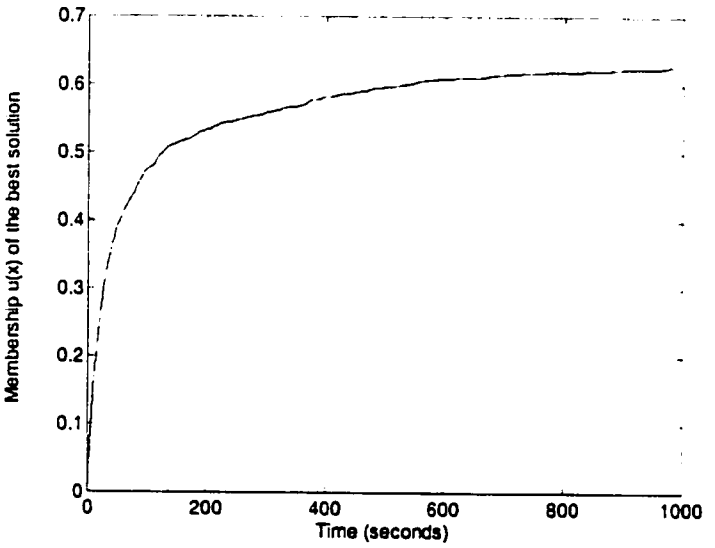
increases, TS consistently performs better, and the difference between GA and TS performance gets higher. Also, the execution time of GA increases significantly with the increase in circuit complexity. The higher execution time of GA can be attributed to its parallel nature, i.e., a population of solutions is to be processed in each generation.

Figures 6.1 (a) and (b) show the trend of best solution's membership for GA and TS respectively, in case of circuit s832. It is clear from the shown plots that TS achieves a membership in 1000 seconds that is better than that reached by GA even after 8000 seconds. Figures 6.2, 6.3, and 6.4 show the trend of actual costs of the best solution for all three objectives in case of GA and TS.

Another interesting and novel comparison is performed between GA and TS as shown in Figures 6.5 and 6.6. This comparison gives an insight of an algorithm's suitability to the problem. For instance, the bar chart shown in Figure 6.5 illustrates the number of generations spent in searching for a particular quality solution in the solution space. These plots indicate that both GA and TS searched most of the time in better solution space and are thus well engineered to the present multiobjective problem. These plots show the best case performance of both techniques.

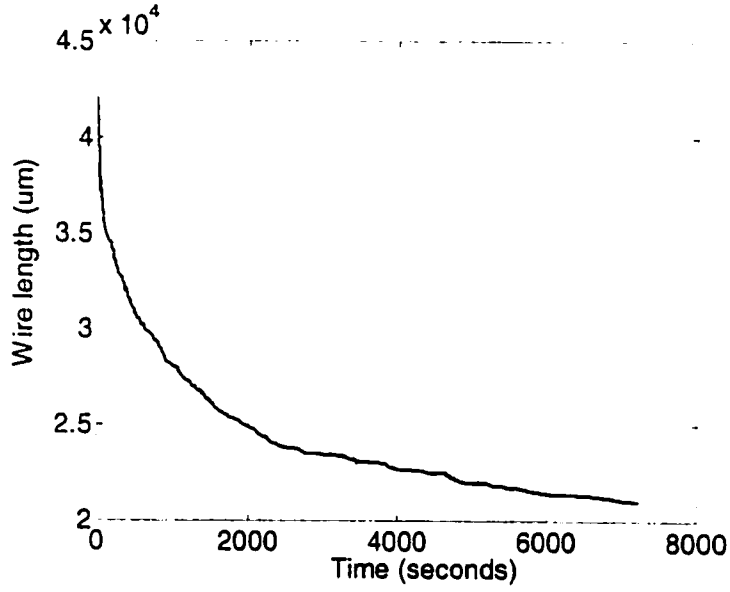


(a)

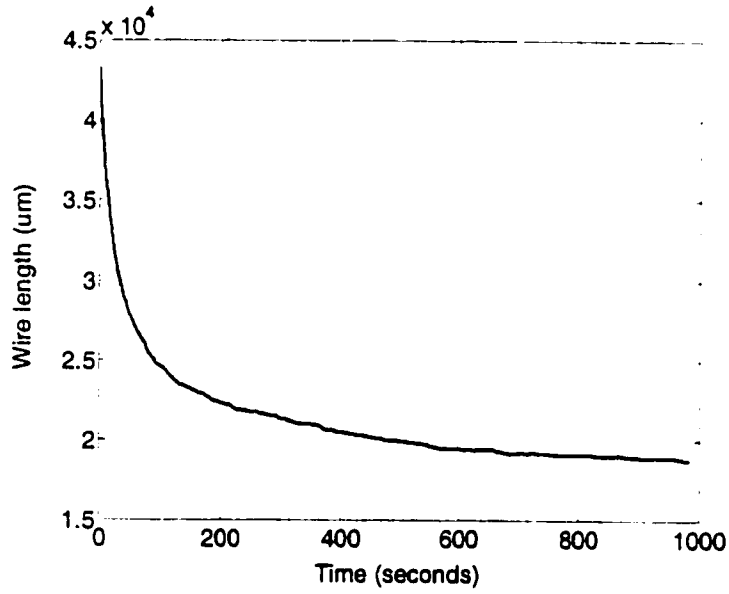


(b)

Figure 6.1: A comparison between GA and TS for circuit s832. (a) and (b) show the fuzzy membership of the best solution against run time for GA and TS respectively.



(a)



(b)

Figure 6.2: A comparison between GA and TS for circuit s832. (a) and (b) show actual costs for wirelength of the best solution against run time in case of GA and TS respectively.

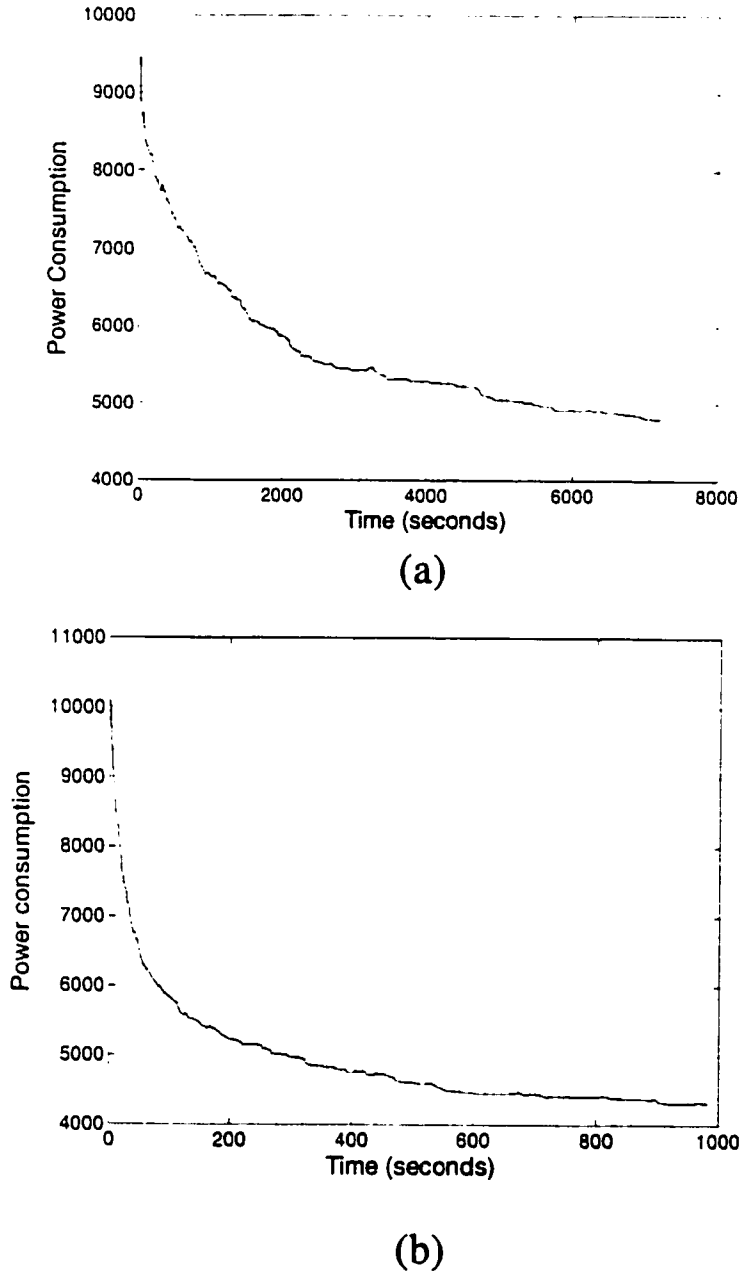
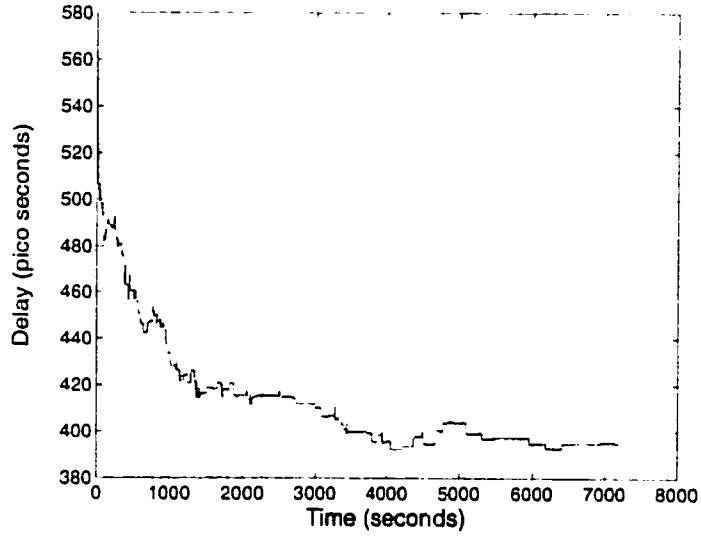
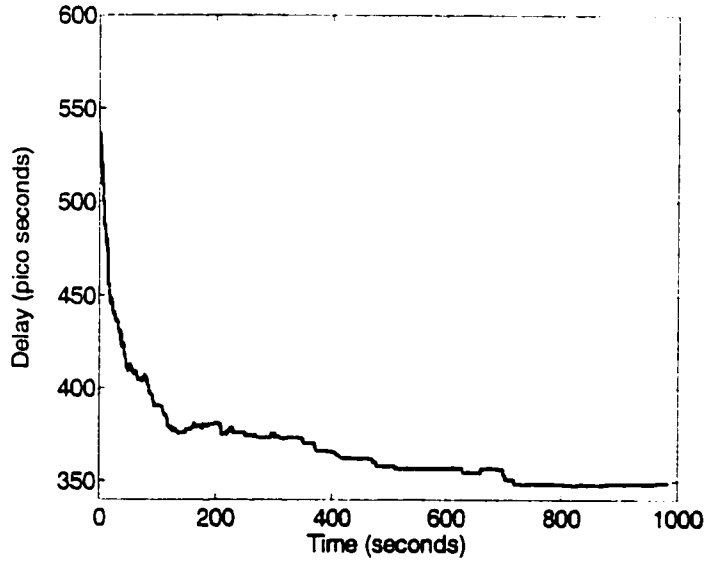


Figure 6.3: A comparison between GA and TS for circuit s832. (a) and (b) show actual costs for power consumption of the best solution against run time in case of GA and TS respectively.



(a)



(b)

Figure 6.4: A comparison between GA and TS for circuit s832. (a) and (b) show actual costs for delay of the best solution against run time in case of GA and TS respectively.

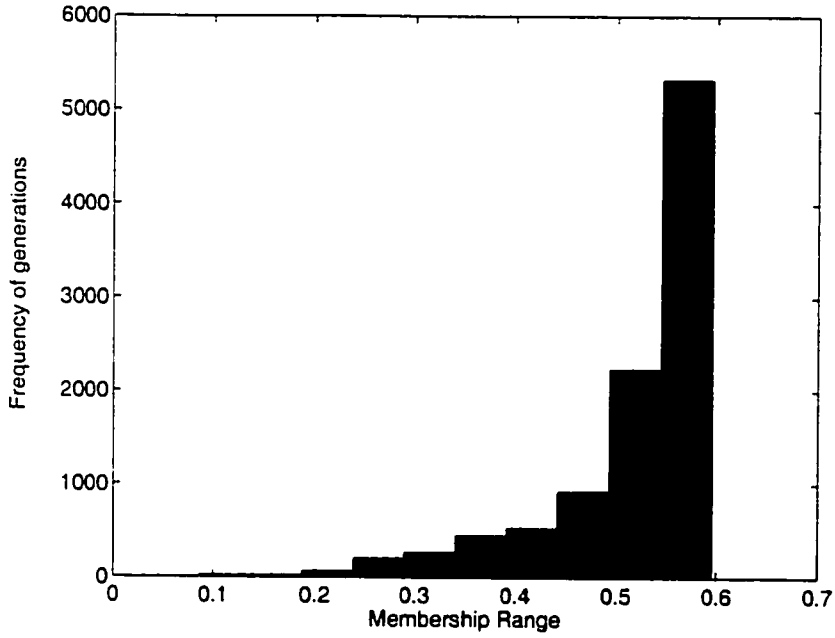


Figure 6.5: A histogram of GA search for circuit s832.

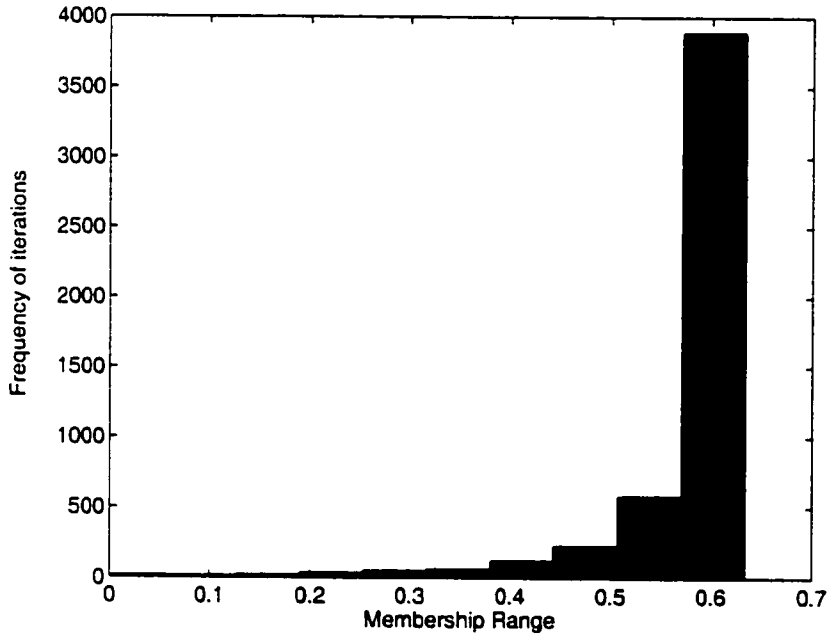


Figure 6.6: A histogram of TS search for circuit s832.

6.3.2 Comparison between TS and GATS

Here, the performance of proposed hybrid algorithm GATS is compared with that of TS. Since TS out-performed GA in terms of the quality of final solution obtained, so the comparison presented in this section is of great interest.

The costs of the best solutions generated by TS and GATS are listed in Table 6.3. The results of TS are obtained by best settings of its parameters as described above. The settings of GATS parameters used for achieving theses results are as follows. Total number of iterations run are 5000, which comprise of 2000 TS iterations and 3000 GA generations. The switch from TS to GA is made only once. The population size N_T used in TS part is 4 while in GA part the population size is 16 chromosomes. This fine tuning of parameters is made after careful study of the results obtained by choosing different settings. The population size in case of TS is reduced after observing that large population size increases run time of TS part without providing any significant performance. By taking this step, the run time of GATS is shortened very significantly.

It can be observed from the results that in most of cases, GATS produced solutions which are better in quality as compared to those obtained from TS. Although, the execution time of GATS is higher than TS, but this is tolerable considering the better quality of solutions. The overall performance of GATS is comparable to that of TS, and much better than GA.

	TS					GATS				
Circuit	L (μm)	P	D (ps)	$\mu(x)$	T (s)	L (μm)	P	D (ps)	$\mu(x)$	T (s)
s2081	2323	379	111	0.823	298	2162	356	110	0.845	426
s298	3579	635	127	0.821	212	3454	631	125	0.832	362
s386	6643	1595	190	0.709	524	6329	1486	191	0.727	656
s641	12620	2868	656	0.800	1505	12433	2882	664	0.802	2143
s832	18760	4311	349	0.658	981	18451	4257	351	0.664	1929
s953	27287	4230	214	0.669	1036	25967	4239	212	0.679	1846
s1196	39054	11700	332	0.650	1138	38574	11526	334	0.654	3276
s1238	39186	11594	353	0.656	1124	39065	11342	351	0.660	3667
s1488	56888	13867	662	0.621	2256	56148	14135	669	0.624	5157
s1494	54710	13533	674	0.650	2499	54914	13763	668	0.649	5643
c3540	164581	58143	699	0.708	19215	164245	57905	703	0.709	33247

Table 6.3: A comparison between the quality of the best solutions generated by TS and GATS.

6.4 Sensitivity Analysis

The performance of iterative algorithms is sensitive to fine tuning of various algorithmic parameters. This section discusses the effect of these settings on the quality of the final solution. For instance, in case of GA, the effect of varying population sizes, crossover and mutation probability is analyzed. Similarly, in case of TS, the effect of neighborhood size is discussed.

The effect of population size on the performance of GA can be seen in Figure 6.7. It is observed that by increasing the population size from 16 to 32 chromosomes, the quality of the solution improves significantly, however, population sizes beyond 32 result in smaller improvement with the excessive CPU time requirement. The problem is that the execution time of GA rises sharply with the increase in population size. The reason for this behavior is that the complexity of GA heavily depends on population size. As the population size increases, the run time boosts up. Due to

this reason, population size is chosen to be equal to 32 chromosomes.

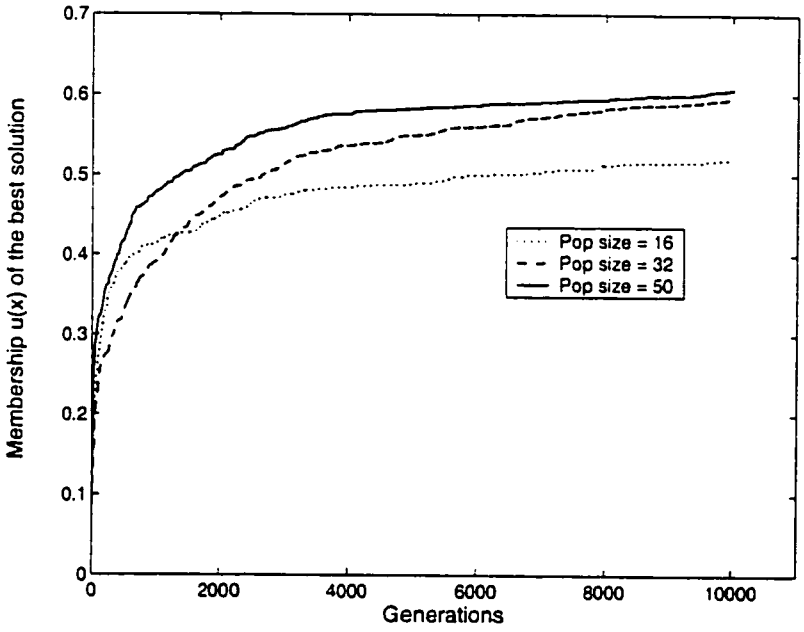


Figure 6.7: Effect of population size on the quality of the best solution generated by GA for circuit s832.

Crossover probability	L (μm)	P	D (ps)
0.70	21669	5132	403
0.80	21661	5031	396
0.90	21605	4972	398
0.99	21015	4787	394

Table 6.4: The effect of varying the crossover probability in GA.

Table 6.4 shows the effect of changing crossover probability in GA. Different crossover rates from 0.7 to 0.99 are tried. The increase in crossover rate improves the quality of the final solution. The reason is that higher crossover rate encourages the inheritance of characteristics and thus GA capability.

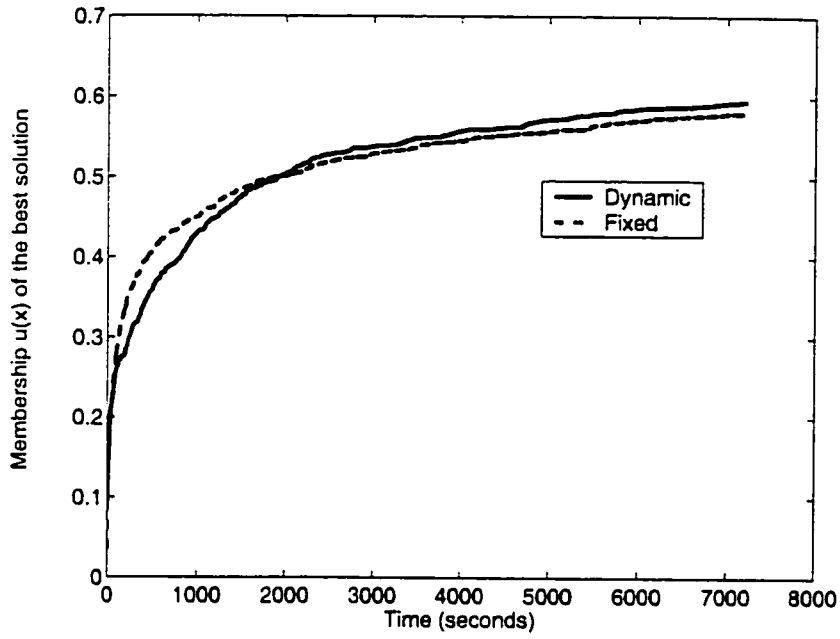


Figure 6.8: A comparison between using dynamic and fixed mutation probability for circuit s832.

Figure 6.8 compares the trend of the best solution cost for circuit s832, both in case of fixed and dynamic mutation probability. It is clear that using dynamic mutation probability results in slightly better performance. The detailed comparison of the quality of the best solutions obtained by using dynamic and fixed mutation probability is given in Table 6.5. It is observed in most cases, that dynamic mutation probability results in a marginal better performance over fixed mutation probability. This trend can be explained as follows. Fixed mutation rate affects chromosomes in the current population irrespective of the diversity in the population. This may disturb the solutions and hence the search direction needlessly. On the other hand, dynamic mutation probability approach has a built in mechanism to control that

how much mutation needs to be applied depending on the diversity. If diversity is sufficient then mutation rate is dropped to as low as 0.02.

	Fixed Mutation Probability = 0.1				Dynamic Mutation Probability			
Circuit	L (μm)	P	D (ps)	$\mu(x)$	L (μm)	P	D (ps)	$\mu(x)$
s2081	2551	410	114	0.764	2426	388	113	0.785
s298	4424	861	130	0.749	4062	838	130	0.775
s386	7584	1809	195	0.650	6824	1665	193	0.695
s641	17812	4532	740	0.685	18320	4365	736	0.679
s832	21817	5107	368	0.587	21015	4787	395	0.598
s953	32031	5156	230	0.597	31004	5027	235	0.606
s1196	48729	14755	372	0.543	49162	14721	403	0.533
s1238	52679	15473	410	0.517	50387	15035	396	0.536
s1488	70400	17568	747	0.517	69792	17346	784	0.518
s1494	71021	17497	803	0.525	69223	17169	771	0.540
c3540	314172	108447	954	0.418	310996	109850	924	0.425

Table 6.5: A comparison between using fixed and dynamic mutation probability in GA.

In case of TS, an important parameter is neighborhood size i.e., the number of neighbor solutions generated in each iteration. The quality of the final solution heavily depends on the tuning of this parameter. Experiments are conducted for different values of this parameter and results indicate that the increase in neighborhood size improves the performance of TS at the cost of an increase in run time. Figure 6.9 shows the effect of neighborhood size for circuit s832. The reason for the improvement in the performance of TS by increasing the above parameter is that the search process becomes more aggressive as larger number of neighbor points in search space are explored in each iteration.

For GATS, experiments are also performed using equal population sizes for TS

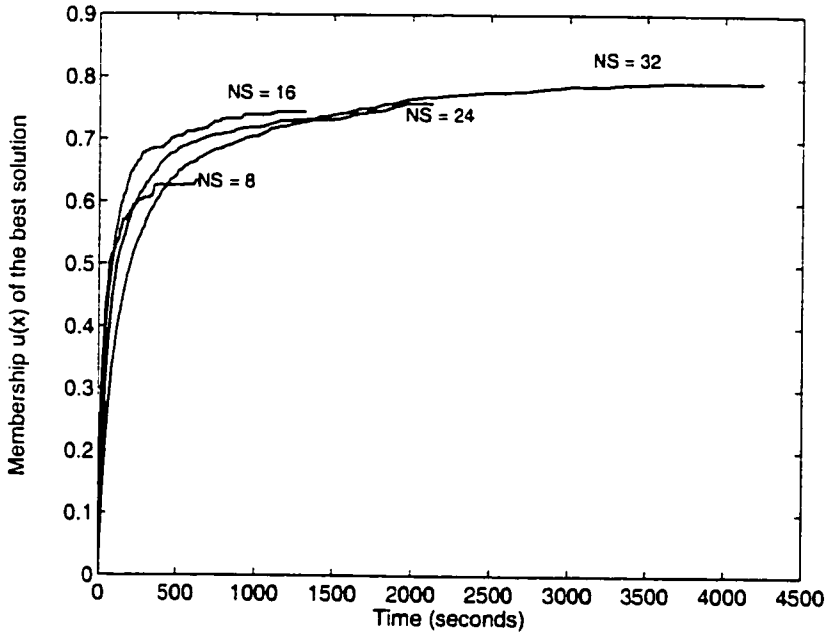


Figure 6.9: The effect of varying the neighborhood size on the performance of TS for circuit s832.

and GA parts i.e., $N_G = N_T = 16$. The comparison between the cases when $N_T = 4$ and $N_T = 16$ is shown in Table 6.6. It is clear from the results of the two cases that taking smaller population size in TS part is a better choice, because it does not affect the quality of solution significantly but cuts down the run time of GATS by a significant amount.

	$N_T = 16$					$N_T = 4$				
Circuit	L (μm)	P	D (ps)	$\mu(x)$	T (s)	L (μm)	P	D (ps)	$\mu(x)$	T(s)
s2081	2098	329	110	0.847	809	2162	356	110	0.845	426
s298	3518	700	124	0.827	939	3454	631	125	0.832	362
s386	5886	1360	181	0.763	1729	6329	1486	191	0.727	656
s641	12458	2922	661	0.802	19647	12433	2882	664	0.802	2143
s832	17927	4109	331	0.681	3762	18451	4257	351	0.664	1929
s953	24412	3795	205	0.710	6198	25967	4239	212	0.679	1846
s1196	37135	11041	336	0.667	10128	38574	11526	334	0.654	3276
s1238	39371	11549	342	0.660	10009	39065	11342	351	0.660	3667
s1488	55789	13712	664	0.628	11123	56148	14135	669	0.624	5157
s1494	54864	13681	661	0.647	13940	54914	13763	668	0.649	5643
c3540	164193	57943	701	0.709	57060	164245	57905	703	0.709	33247

Table 6.6: Effect of TS population size (N_T) on the performance of GATS.

6.5 Comparison of Various Crossover Operators and Selection Schemes for GA

In this section, first, the performance of different crossover operators employed in this work is analyzed. Then, the effect of using various selection schemes listed in Chapter 4 is studied.

Different crossover operators used in this work include square crossover [7], order crossover, PMX, and a proposed modified application of PMX termed Controlled Dual PMX (CDX). The performance of these operators is compared with one another. Figure 6.10 shows the effect on the quality of the best solution by employing CDX, PMX and order crossover operators. The proposed CDX is clearly outperforming PMX and order crossover operators. Order crossover performs very poor as compared with the other two crossover operators. For the results shown in the Figure, all three crossover operators are applied with a probability of 0.99. Further-

more. the settings of all other parameters like mutation rate and selection scheme are kept the same for obtaining these results.

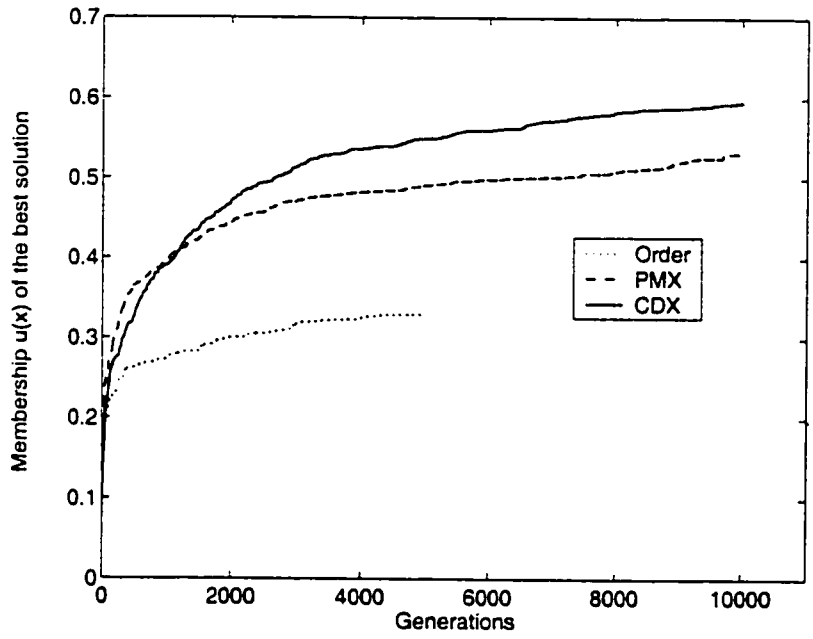


Figure 6.10: A comparison of Order, PMX, and CDX for circuit s832.

The reason for excellent performance of the proposed CDX is that it provides a balance between greediness and randomness. It selects the better of two newly generated offsprings with a certain probability P_{cdx}, but also selects either of the two randomly with a probability $1 - P_{cdx}$. The worst performance of order crossover is not surprising considering the work reported in [3, 37]. They also reported similar behavior of order crossover for single objective (wirelength only) VLSI cell placement.

A comparison of the performance of various selection schemes used is also carried

out. Seven different schemes are investigated in this work. The details of all these schemes is given in Chapter 4. It is observed after experimenting that *elitist* selection schemes outperform the others. In addition, when the performance of five elitist schemes used in this work is compared with one another, it is found that *ernd* and *eernd* perform better than *erlt* and *eerlt* respectively. The reason for this trend is clear that *erlt* and *eerlt* have very greedy behavior because these schemes encourage the selection of fitter individuals even after selecting the best ones. On the other hand, *ernd* and *eernd* have a reasonably balanced approach. After selecting the best ones, these schemes allow all remaining individuals to be selected at random. As a result, *ernd* and *eernd* allow diversity in the population, which is known to be necessary for the efficient working of GA.

For simplicity, the best three schemes are compared with each other. Figure 6.11 shows the trend of membership value for the best solution obtained when using the three different elitist schemes *eernd, ernd, and eprnd.* The settings for other parameters are as follows. The population size is 50, crossover applied is CDX with a probability of 0.99, and the mutation rate is dynamic. It can be seen that *eernd* gives the best performance. This behavior can be interpreted as follows. The *eernd* scheme selects the best $\frac{N_p}{2}$ chromosomes in contrast with *ernd* and *eprnd* that select only the best chromosome. This large number of better chromosomes is more likely to produce fitter offsprings in the following generations.

On the other hand, *ernd* and *eprnd* schemes allow larger number of randomly

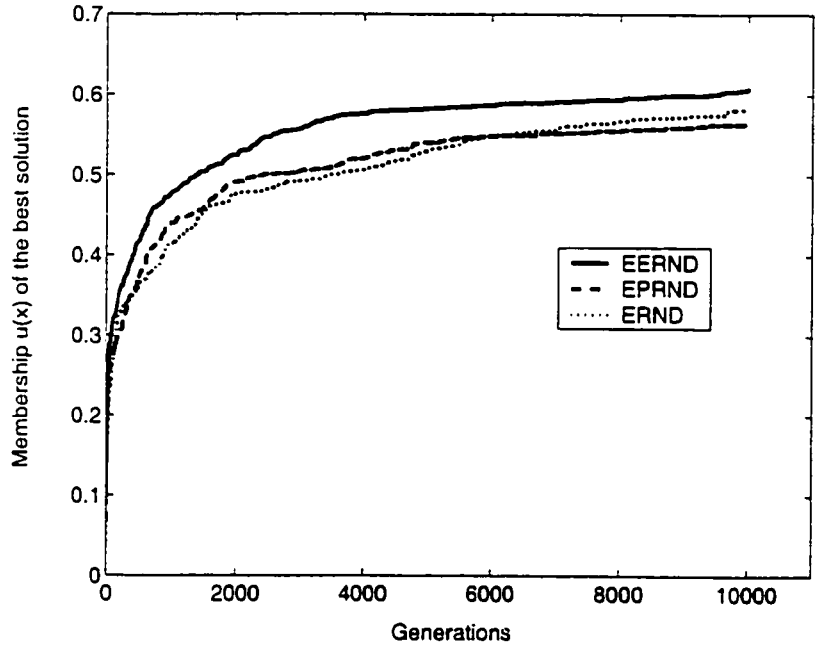


Figure 6.11: A comparison of three different selection schemes of GA for s832.

selected individuals to propagate to next generation. These randomly selected chromosomes have unpredictable quality and thus the search direction of GA is more disturbed than that in case of *eernd*.

6.6 Single Objective Versus Multiobjective Optimization

In the earliest reported efforts for VLSI cell placement, wirelength was the only objective to be optimized. Therefore, those placement techniques were of the type of single objective optimization (SOP). With the advancement in technology and due to

some other factors, two more objectives namely performance and power consumption also came into focus. This resulted in the need of multiobjective optimization (MOP) techniques. This work also addresses such multiobjective optimization (MOP) in which, all of above three objectives are being targeted.

It is desirable to compare these two approaches, i.e., MOP and SOP, in terms of studying their effects on the overall quality of the final solution. Another purpose of this comparison is to investigate the effect on the costs of individual objectives when the search process is targeting multiple possibly conflicting objectives. Also, the effect on the cost of an objective is studied, when another objective is being targeted for optimization.

The following three sub-sections present the comparison between MOP and SOP for wirelength, SOP for delay, and SOP for power consumption respectively. Each sub-section, presents the comparisons in case of GA, TS and GATS.

6.6.1 wirelength only optimization versus MOP

In this section, a comparison between SOP for wirelength and MOP is presented. This is an interesting comparison since earliest reported placement techniques targeted wirelength as the only objective, and here the effects of MOP on wirelength cost are studied. Also the effect of SOP for wirelength on the costs of other objectives, i.e., delay and power consumption can be analyzed.

Table 6.7 lists the values of all three objectives in the solutions obtained from

GA in both cases, i.e., the case of MOP and the case in which optimization is performed for wirelength only. Similarly, Table 6.8 and 6.9 show the same comparison for the solutions obtained from TS and GATS respectively. In these tables, L represents the wirelength in μm, P represents the cost due to power, D is the delay of the most critical path in pico seconds and T is the execution time in seconds for reaching the best solution. The power cost is the sum of the products of switching probabilities and wirelength of all the nets. It is observed in most of the circuits that wirelength optimization approach results in slightly smaller wirelength but not without an increase in values of the other two objectives. The power consumption objective is less affected because it is somewhat directly proportional to wirelength, i.e., minimizing the wirelength of all nets also covers the nets having high switching probabilities. The delay on the other hand, may not be minimized by minimizing wire length of nets because the value of delay depends on the delay of the nets that are on the longest path in the circuit.

6.6.2 Power only optimization vs MOP

Another interesting comparison is between the single-objective optimization for power and multiobjective optimization. The costs of the three objectives for the best solutions obtained from GA, TS, and GATS in both cases are given in Table 6.10, Table 6.11, and Table 6.12 respectively.

For most of the circuits, it is observed that optimization for power only performs

	For wirelength only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	2379	464	117	2426	388	113
s298	3986	816	152	4062	838	130
s386	6765	1776	205	6824	1665	193
s641	18159	4654	774	18320	4365	736
s832	20844	5156	428	21015	4787	395
s953	31065	5356	236	32031	5156	230
s1196	46371	14364	431	49162	14721	403
s1238	49878	15451	469	52679	15473	410
s1488	69515	17751	914	69792	17346	784
s1494	69013	17556	900	71021	17497	803
c3540	308260	110607	1235	310996	109850	924

Table 6.7: A Comparison between the quality of the best solutions obtained from GA by performing SOP for wirelength and MOP.

	For wirelength only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	2172	447	112	2323	379	111
s298	3500	787	129	3579	635	127
s386	6615	1708	193	6643	1595	190
s641	12398	3092	701	12620	2868	656
s832	17987	4038	347	18760	4311	349
s953	25972	4023	207	27287	4230	214
s1196	38563	12222	404	39054	11700	332
s1238	36775	11382	400	39186	11594	353
s1488	55249	13804	775	55788	13820	657
s1494	54485	13942	814	54710	13533	674
c3540	153540	56591	810	164581	58143	699

Table 6.8: A Comparison between the quality of the best solutions obtained from TS by performing SOP for wirelength and MOP.

	For wirelength only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	1974	344	118	2098	329	110
s298	3497	705	138	3518	700	124
s386	5762	1417	197	5886	1360	181
s641	12310	3295	686	12458	2922	661
s832	17633	4131	381	17927	4109	331
s953	24338	4325	224	24412	3795	205
s1196	35773	11397	371	37135	11041	336
s1238	36154	11398	413	39371	11549	342
s1488	55660	14286	837	55789	13712	664
s1494	54790	13578	657	54914	13763	668
c3540	155935	60207	937	164193	57943	701

Table 6.9: A Comparison between the quality of the best solutions obtained from GATS by performing SOP for wirelength and MOP.

	For power only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	3046	381	123	2426	388	113
s298	6176	762	172	4062	838	130
s386	8996	1514	230	6824	1665	193
s641	22418	4287	783	18320	4365	736
s832	25762	4691	463	21015	4787	395
s953	43184	4487	329	32031	5156	230
s1196	55412	14373	459	49162	14721	403
s1238	59564	15373	468	52679	15473	410
s1488	76561	16653	896	69792	17346	784
s1494	74375	16313	903	71021	17497	803
c3540	325340	107659	1104	310996	109850	924

Table 6.10: A Comparison between the quality of the best solutions obtained from GA by performing SOP for power consumption and MOP.

	For power only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	3226	310	118	2323	379	111
s298	6654	605	169	3579	635	127
s386	9753	1306	238	6643	1595	190
s641	18372	2540	755	12620	2868	656
s832	23707	3462	453	18760	4311	349
s953	42271	2852	344	27287	4230	214
s1196	46600	10571	451	39054	11700	332
s1238	48634	10730	477	39186	11594	353
s1488	64769	12237	876	55788	13820	657
s1494	63628	11941	896	54710	13533	674
c3540	190553	55152	870	164581	58143	699

Table 6.11: A Comparison between the quality of the best solutions obtained from TS by performing SOP for power consumption and MOP.

	For power only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	3275	293	129	2098	329	110
s298	5398	575	165	3518	700	124
s386	8955	1154	232	5886	1360	181
s641	18355	2483	708	12458	2922	661
s832	23880	3310	435	17927	4109	331
s953	39777	2560	320	24412	3795	205
s1196	45125	9434	458	37135	11041	336
s1238	45925	9356	466	39371	11549	342
s1488	64736	12068	906	55789	13712	664
s1494	54864	13681	661	54914	13763	668
c3540	210316	57605	844	164193	57943	701

Table 6.12: A Comparison between the quality of the best solutions obtained from GATS by performing SOP for power consumption and MOP.

better in terms of power cost. But at the same time, the other two objectives are affected badly for all the circuits. This behavior is easy to understand. When optimization is done for power only, the target is to minimize the wirelength of the nets having high switching probabilities. All other nets may be ignored in this process and thus the overall wirelength is not reduced to that extent as in the case of multiobjective or wirelength only optimization. The worst results are that of delay objective. Again the reason is obvious that the process of optimizing power is very limited in terms of the number of affected nets and thus it may totally ignore the nets lying on the timing-critical paths. As a result, delay costs are not optimized at all. As seen in the previous sub-section, when optimization was performed for wirelength only, the delay costs were lower than in the present case. This is because the wirelength optimization targets all the nets and in this way, the wirelength of the nets on critical paths is also minimized with the other nets in circuit.

6.6.3 Delay only optimization vs MOP

In this section, a comparison between SOP for delay and MOP is made. Table 6.13 shows the values of wire length, power, and delay costs for the solutions obtained from GA in case of SOP for delay and that of MOP. Tables 6.14 and 6.15 show the similar comparison in case of the solutions obtained from TS and GATS respectively.

As expected, the single-objective optimization for delay ignores the other objectives. The costs of wirelength and power consumption are very high in this case for

	For delay only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	5382	1056	110	2426	388	113
s298	8639	1906	124	4062	838	130
s386	11037	2945	188	6824	1665	193
s641	31978	7866	680	18320	4365	736
s832	30863	8161	340	21015	4787	395
s953	47478	8861	219	32031	5156	230
s1196	72187	22333	352	49162	14721	403
s1238	72689	21823	384	52679	15473	410
s1488	97998	24294	672	69792	17346	784
s1494	99119	24083	672	71021	17497	803
c3540	424774	149357	830	310996	109850	924

Table 6.13: A Comparison between the quality of the best solutions obtained by performing optimization for delay only and multiobjective optimization.

	For delay only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	6140	1318	110	2323	379	111
s298	10524	2514	121	3579	635	127
s386	12965	3599	179	6643	1595	190
s641	42954	10408	642	12620	2868	656
s832	37804	9867	303	18760	4311	349
s953	57022	11245	197	27287	4230	214
s1196	81319	24546	320	39054	11700	332
s1238	86711	25435	338	39186	11594	353
s1488	113713	27050	554	55788	13820	657
s1494	109767	26835	556	54710	13533	674
c3540	535921	186235	686	164581	58143	699

Table 6.14: A Comparison between the quality of the best solutions obtained from TS by performing SOP for delay and MOP.

	For delay only			For all objectives		
Circuit	L (μm)	P	D (ps)	L (μm)	P	D (ps)
s2081	6569	1434	106	2098	329	110
s298	11319	2751	119	3518	700	124
s386	12596	3480	177	5886	1360	181
s641	40529	9563	635	12458	2922	661
s832	35180	9498	297	17927	4109	331
s953	49416	9658	187	24412	3795	205
s1196	79528	24799	298	37135	11041	336
s1238	83620	24390	320	39371	11549	342
s1488	107599	25827	515	55789	13712	664
s1494	54864	13681	661	54914	13763	668
c3540	501545	173429	652	164193	57943	701

Table 6.15: A Comparison between the quality of the best solutions obtained from GATS by performing SOP for delay and MOP.

all the circuits. The reason for this behavior is that the optimization for delay targets a limited number of nets which lie on the critical paths. All the other nets are totally ignored. As a consequence, the overall wirelength and power consumption is not optimized at all.

In conclusion, it can be said that MOP is better than SOP in the sense that all the objectives are simultaneously optimized without adversely affecting the cost of any single objective. On the other hand, SOP for any one of the objectives may ignore the other objectives at all. While performing MOP, still a certain objective can be given preference over others according to the designer requirements.

Chapter 7

Conclusions and Future Directions

In this thesis, the problem of timing and low power driven VLSI standard-cell placement is addressed. This is formulated as a multiobjective optimization problem (MOP) and the use of fuzzy rules is proposed for designing aggregate cost function. Two iterative algorithms namely GA and TS are presented for solving this hard problem. In addition, a hybrid algorithm is proposed that combines the powerful features from two iterative heuristics namely GA and TS. The results of the above techniques are promising and show that these are well engineered for the problem. Here are some concluding statements regarding the thesis work.

- The present work successfully addressed the important issue of reducing power consumption in VLSI circuits.

- Tabu search outperformed genetic algorithm in terms of the quality of the final solution as well as execution time.

- Hybridization of GA and TS provided better quality results than both GA and TS.

- The performance of GA depends heavily on population size whereas that of TS depends heavily on neighborhood size.

- Considering the above observation, it was suggested in case of GATS to use smaller population size for the TS sub-process. This resulted in a significant saving of the execution time without affecting the quality of solutions obtained from GATS.

- Multiobjective optimization (MOP) resulted in solutions that are better in terms of the overall cost. On the other hand, single objective optimization (SOP) for a particular objective may ignore other objectives at all. For instance, in case of delay optimization, the costs of interconnect wirelength and power consumption are severely affected.

Future Directions

Some possible directions of the future work are following.

- Investigation of some other iterative heuristics for the present problem.

- Some other hybrid techniques can be proposed and experimented for further improvement of the results.

- The presented standard-cell placement techniques can be extended to perform the optimization of channel routing step of VLSI physical design.

Bibliography

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

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

[3] K. Shahookar and P. Mazumder. VLSI Cell Placement Techniques. *ACM Computing Surveys*, 2(23):143–220, June 1991.

[4] Eager. Advances in Rechargable Batteries Pace Portable Computer Growth. *Proceedings of Silicon Valley Personal Computer Conference*, pages 693–697, 1991.

[5] Massoud Pedram. CAD for Low Power: Status and Promising Directions. *IEEE International Symposium on VLSI Technology, Systems and Applications*, pages 331-336, 1995.

[6] Unni Narayanan, G.I. Stamoulis, and Rabindra Roy. Characterizing Individual Gate Power Sensitivity in Low Power Design. *12th International Conference on VLSI Design*, pages 625–628, January 1999.

[7] J. P. Cohoon and W. D. Paris. Genetic Placement. *IEEE Transactions on Computer Aided Design*, pages 956–964, 1987.

[8] A. Chandrakasan, T. Sheng, and R. W. Brodereson. Low Power CMOS Digital Design. *Journal of Solid State Circuits*, 27, April.

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

[10] M.S. Bright and T. Arslan. A Genetic Algorithm For The High-Level Synthesis of DSP Systems For Low Power. *Genetic Algorithms in Engineering Systems: Innovations and Applications*, (446):174–179, September 1997.

[11] Chetana Nagendra, Mary Jane Irwin and Robert Michael Owens. Area-Time-Power Tradeoffs in Parallel Adders. *IEEE Transactions on Circuits and Systems-II Analog and Digital Signal Processing*, 43(10):689–702, October 1996.

[12] A. Chandrakasan et al. Optimizing Power using Transformations. *IEEE Transactions on Computer Aided Design*, 1(14), January 1995.

[13] A. Chatterjee and R. Roy. Sythesis of Low Power Linear DSP Circuits using Activity Metrics . *In International Conference on VLSI Design, India*, January 1994.

[14] L. Goodby, A Orailoglu, and P. Chau. Microarchitectural Sythesis of Performance-Constrained, Low-Power VLSI Design . *In Proceedings of the International Conference on Computer Design, Boston,MA*, pages 323–326, October 1994.

[15] A. Raghunathan and N. Jha. Behavioral Synthesis for Low Power. *In Proceedings of the Int'l Conference on Computer Design, Boston, MA*, pages 318–322, October 1994.

[16] A. Raghunathan and N. Jha. ILP Formulation for Low Power Based on Minimizing Switched Capacitances During Data Path Allocation. *In Proceedings of the Int'l Symposium on Circuits & Systems*, 1995.

[17] H. Savoj, R. Brayton, and H. Touati. Extracting Local Don't Cares for Network Optimization. *In Proceedings of the Int'l Conference on Computer-Aided Design*, pages 514–517, November 1991.

[18] A. Shen et al. On Average Power Dissipation and Random Pattern Testability of Combinational Logic Circuits. *In Proceedings of the Int'l Conference on Computer-Aided Design*, pages 402–407, November 1992.

[19] A. Ghosh et al. Estimation of Average Switching Activity in Combinational and Sequential Circuits. *In Proceedings of the 29th Design Automation Conference*, pages 253–259, June 1992.

[20] C. Lemonds and S. S. Mahant Shetti. A Low Power 16 by 16 Multiplier using Transition Reduction Circuitry. *In Proceedings of the Int'l Conference on Low Power Design*, pages 139–142, April 1994.

[21] K. Roy and S. Prasad. SYCLOP: Synthesis of CMOS Logic for Low Power Applications. *In Proceedings of the Int'l Conference on Computer Design: VLSI in Computer and Processors*, pages 464–467, October 1992.

[22] K. Keutzer. DAGON: Technology Mapping and Local Optimization. *In Proceedings of the 24th Design Automation Conference*, pages 341–347, June 1987.

[23] B. Lin. Technology Mapping for Low Power Dissipation. *In Proceedings of the Int'l Conference on Computer Design:VLSI*, October 1993.

[24] V. Tiwari, P. Ashar, and S. Malik. Technology Mapping for Low Power. *In Proceedings of the 30th Design Automation Conference*, pages 74–79, June 1993.

[25] C. Y. Tsui, M. Pedram, and A. M. Despain. Low Power State Assignment Targeting Two and Multi Level Logic. *In Proceedings of the Int'l Conference on Computer Aided Design*, pages 82–87, November 1994.

[26] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing Synchronous Circuitry by Retiming. *In Proceedings of the 3rd CalTech Conference on VLSI*, pages 23–36, March 1983.

[27] J. Monteiro, S. Devadas, and A. Ghosh. Retiming Sequential Circuits for Low Power. *In Proceedings of the Int'l Conference on Computer-Aided Design*, pages 398–402, November.

[28] Anantha P. Chandrakasan. Low Power Digital CMOS Design. *Ph.D. Thesis, University of California at Berkeley*, August 1994.

[29] Alidina et al. Precomputation-Based Sequential Logic Optimization for Low Power. *IEEE Transactions on VLSI Systems*, pages 426–436, April.

[30] I. S. Choi and S. Y. Hwang. Circuit Partitioning algorithm for Low-Power Design Under Area Constraints Using Simulated Annealing. *IEE Proceedings Circuits Devices Systems*, 146(1):8–15, February 1999.

[31] Kai-Yuan Chao and D.F. Wong. Floorplanning for Low Power Design. *IEEE International Symposium on Circuits and Systems*, 1:45–48, 1995.

[32] Kaushik Roy. Power-Dissipation Driven FPGA Place and Route under Timing Constraints. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Application*, 46(5):634–637, May 1999.

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

[34] Hirendu Vaishnav and Massoud Pedram. PCUBE: A Performance Driven Placement Algorithm for Low Power Design. *IEEE Design Automation Conference, with Euro-VHDL*, pages 72–77, 1993.

[35] A. Srinivasan, K. Chaudhary, and E. S. Kuh. Ritual: A Performance-driven Placement Algorithm. *IEEE Transactions on Circuits and Systems -II*, 11(39):825–840, November 1992.

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

[37] K. Shahookar and P. Mazumder. A Genetic Approach to Standard Cell Placement using Meta-Genetic Parameter Optimization. *IEEE Transactions on Computer Aided Design*, 5(9):500–511, May 1990.

[38] Thomas Lengauer. *Combinational Algorithms for Integrated Circuit Layout.* John Wiley and Sons, New York, 1990.

[39] M. Burstein and M. N. Youssef. Timing influenced layout design. *Proceedings of 22nd Design Automation Conference*, pages 124–130, 1985.

[40] R. Nair et al. Generation of Performance Constraints for Layout. *IEEE Transaction on Computer Aided Design, CAD*, 8(8):860–874, August 1989.

[41] H. Youssef, Rung-Bin Lin, and E. Shragowitz. Bounds on Net Delays for VLSI Circuits . *IEEE Transactions on Circuits and Systems -II*, 11(39):815–824, October 1992.

[42] S. Sutanthavibul, E. Shragowitz, and Rung-Bin Lin. An Adaptive Timing-driven Placement for High Performance VLSI's. *IEEE Transactions on Computer Aided Design*, 10(12):1488–1489, October 1993.

[43] M. Marek-Sadowska and S. Lin. Timing-driven placement. *Proc. of ICCAD'89*, pages 94–97, 1989.

[44] Sadiq M. Sait, H. Youssef, K. W. Nassar, and M. S. T. Benten. Timing Driven Genetic Placement. *International Journal of Computer Systems: Science and Engineering*, February 1997.

[45] MCNC Group. OASIS 2.0 Reference Manual. Technical report, 1990.

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

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

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

[49] H.J. Zimmerman. *Fuzzy Set Theory and Its Applications.* Kluwer Academic Publishers, 3rd edition, 1996.

[50] 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.

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

[52] R. Yager. Multiple Objective Decision-Making using Fuzzy Sets. *International Journal of Man-Machine Studies*, pages 9:375–382, 1977.

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

[54] J. H. Holland. *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press, Ann Harbor, Michigan, 1975.

[55] F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, MA, 1997.

[56] J. P. Kelly, M. Laguna, and F. Glover. A Study of Diversification Strategies for the Quadratic Assignment Problem. *Computers Operations Research*, 21(8):885–893.

Vitae

- **Mahmood-ur-Rehman Minhas**
- Born in Rawalpindi, Pakistan.
- Received Masters Degree in Computer Science from International Islamic University, Islamabad, Pakistan.
- Joined Computer Engineering Department, KFUPM, as a Research Assistant in January 1999.
- Received Master of Science Degree in Computer Engineering from KFUPM, Dhahran, Saudi Arabia in June 2001.