ABSTRACT

KLOYPAYAN, JIRAWAN. Solving Complex Modeling of System-on-a-Chip (SOC) Test Automation and Optimal Resource Allocation by Neural Networks. (Under the direction of Professor Yuan-Shin Lee.)

The objective of this research is to optimize the testing time and test resource allocation for *System-on-a-Chip* (SOC). The mathematical formulation and the neural networks with different techniques are proposed to solve these SOC test problems. First, a *fixed-weight neural network* combined with heuristic algorithms has been developed to solve the SOC test scheduling problems. The objective of this SOC test automation is to minimize the SOC testing time subject to different constraints: (i) precedence constraint, (ii) resource constraint, (iii) core constraint, and (iv) power constraint. Heuristic algorithms are often used to prevent the neural network from getting trapped in a local optima. The developed neural network can effectively solve the SOC test scheduling models with disjunctive constraints. The results show that the proposed method can efficiently solve a large-size SOC test scheduling problem within reasonable computing time. Second, to solve the resource allocation and the width selection problems for SOC test automation, a *maximum neural network* (MNN) has been proposed in this research for handling more complex SOC test problems. The SOC test automation problem with resource allocation is a NP-hard problem. The proposed maximum neural network can be used to solve the NP-hard SOC test problems within polynomial time. The results show that, by using the developed maximum neural network, the overall testing time for the SOC can be minimized with optimal resource allocation and *test access mechanism* (TAM) width selection. The computation time of the

proposed method is significantly less than the time for traditional methods such as the integer linear programming (ILP) or heuristic algorithms. Third, the SOC test automation problems with core test wrapper design have been studied in this research. The core test wrapper design provides an interface between the core and the SOC in which the core is embedded. After the core test wrapper is designed, the total SOC testing time and the resource allocation for SOC test automation are optimized by using the developed maximum neural network. The proposed method is tested on five SOC benchmarks. The results show that it is possible to find the optimal SOC testing time of the complex SOC systems with shorter computation time than with the existing traditional methods. The techniques presented in this research can be used in test automation for System-on-a-Chip (SOC) design.

Solving Complex Modeling of System-on-a-Chip (SOC) Test Automation and Optimal Resource Allocation by Neural Networks

By

Jirawan Kloypayan

A dissertation submitted to the Graduate Faculty of

North Carolina State University

in partial fulfillment of the

requirements for the Degree of

Doctor of Philosophy

INDUSTRIAL ENGINEERING

Raleigh

2002

APPROVED BY

_____

Dr. Yuan-Shin Lee
Chair of Advisory Committee

_____

Dr. Ezat T. Sanii
Advisory Committee

_____

Dr. Robert E. Young
Advisory Committee

_____

Dr. Elmor Peterson
Advisory Committee

_____

Dr. Krishnendu Chakrabarty
Advisory Committee

BIOGRAPHY

Jirawan Kloypayan received her B.S. degree in Material Science and M.S. degree in Industrial Engineering from Chulalongkorn University, Thailand. She received her second M.S. degree (1998) in Integrated Manufacturing System Engineering and Ph.D. degree in Industrial Engineering (2002) both from North Carolina State University, USA. Her research interests include soft computing techniques, computer-aided manufacturing, and computational geometry for design and manufacturing. She has accepted a faculty position at the Department of Industrial Engineering at Thammasart University, Bangkok, Thailand.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation and gratitude to my advisor, Dr. Yuan-Shin Lee, for his valuable guidance and assistance throughout this research. I would like to extend my appreciation to Dr. Krisnendu Chakrabarty for his time, helpful encouragement and advice. I appreciate the constructive suggestions and comments from my graduate committee: Dr. R. Young, Dr. E. T. Sanii and Dr. Elmor Peterson.

I would also like to express my thanks to my research groupmates: John, Bahattin, Susana, Yongfu, Abhinand, Weighang and Ron, for their valuable suggestions during group meetings and personal discussions. I would like to extend my gratitude to Dr. Jun, Dr. Yau and Dr. Ju for instructive discussions during their visit. I would also like to specially thank Saowanee, Phoemphun, and Teerada for their friendship in going through this phase of time with me.

I would also like to express my gratitude to the Royal Thai Government. Without the support from Royal Thai Government, it would not be able for me to do this research for Ph.D. degree in the U.S.

Finally, I would like to express my utmost appreciation to my family for their encouragement and support.

TABLE OF CONTENTS

page

LIST OF TABLES . . . . . **vii**

LIST OF FIGURES . . . . . **ix**

1. INTRODUCTION. **1**

1.1. System-on-a-Chip (SOC) Testing. 1

1.2. Problems Description. 3

1.3. Outline of Dissertation. 3

2. USING NEURAL NETWORK WITH FIXED-WEIGHT NET IN MODELING AND SOLVING THE SYSTEM-ON-A-CHIP (SOC) TEST SCHEDULING. **5**

2.1. Introduction. 5

2.2. Neural Network (NN) for Adaptive Learning and Optimization. 7

2.3. Formulation of the SOC Core Test Scheduling Problems. 9

2.4. Constructing the Neural Network for Solving the SOC Testing Problems. . 11

2.4.1. Constructing the Precedence (PC), Resource (RC) and Core (CC) Constraints. 12

2.4.2. Constructing the Power Constraint (PoC) Unit for the Neural Network 15

2.4.3. Conducting the Searching and Optimizing for SOC Test Systems. . . 17

2.5. Computer Implementation and Examples. 19

2.6. Summary. 21

3. DEVELOPING MAXIMUM NEURAL NETWORK TO SOLVE TEST RESOURCE ALLOCATION AND TESTING TIME MINIMIZATION FOR THE SYSTEM-ON-A-CHIP (SOC). 36

3.1. Introduction. 36
3.2. A Maximum Neural Network (NN) for Adaptive Learning and Optimization. 39
3.3. Formulation of SOC Testing with Resource Allocation Problems. 41
3.4. Constructing the Maximum Neural Network (MNN) for Solving the SOC Testing Problems. 43
3.5. An Example and Testing Result of the Proposed Maximum Neural Networks 45
3.6. Summary. 46

4. OPTIMIZING SYSTEM-ON-A-CHIP (SOC) TEST AUTOMATION WITH CORE WRAPPER DESIGN BY MAXIMUM NEURAL NETWORKS. 53

4.1 Introduction. 53
4.2 Core Wrapper Design by Partitioning of TAM Chain Items (PTI) Method and Largest Processing Time (LPT) Method. 55
4.3 Modeling the SOC Test Optimization Problems with Core Wrapper Design. 58
4.4 Constructing the Maximum Neural Network (MNN) to Solve the SOC Testing Problems. 59
4.5 An Example and Testing Results of the Proposed Modeling and MNN. . . 63
4.6 Summary. 63

5. COMPUTER IMPLEMENTATIONS AND RESULTS. 71

5.1. Results for the SOC Test Design with Resource Allocation Problem Using the Maximum NN. 71
5.2 Results for the SOC Test Automation with Core Wrapper Design. 80
5.3 Summary. 82

6. CONCLUSIONS AND FUTURE RESEARCH. 92

REFERENCES. 94

LIST OF TABLES

Page

Table 2.1 Test data of the cores for the SOC system S_1. 28

Table 2.2 Test data of the cores for the SOC system S_2 and S_3. 31

Table 2.3 The SOC test scheduling solutions from the proposed NN after 20 runs. . 35

Table 3.1 Test data of the core for the SOC system S_4. 50

Table 3.2 The results from using the proposed maximum neural network applied to the example SOC system S_4 with three TAMs. 51

Table 4.1 Test data for each core in SOC chip d695. 69

Table 4.2 The best solution of SOC testing time, width allocation and core assignment for the SOC system d695 with three TAMs. 70

Table 5.1 The results from using the proposed maximum neural network applied to the example SOC system S_4 with two TAMs 74

Table 5.2 Comparison of the SOC testing time for different methods. 76

Table 5.3 The results from using the proposed maximum neural network applied to the example SOC system S_4 with four TAMs. 77

Table 5.4 The best solutions of testing time for the system S_4 when the number of TAM lines are varying. 77

Table 5.5 The testing time for the system S_2 with the power constraint when the total TAM widths are varying (the maximum power is 100 mW). . . . 79

Table 5.6 The testing time for the system S_2 with the power constraint when the maximum power dissipation is varying (the total TAM width = 28). 79

Table 5.7 Test data for each core in SOC chip g1023. 83

Table 5.8 Test data for each core in SOC chip p34392. 83

Table 5.9 Test data for each core in SOC chip p22810. 84

Table 5.10 Test data for each core in SOC chip p93791. 85

Table 5.11 Comparison of computation time and the resulting SOC testing time for different methods. 86

Table 5.12 The comparison of SOC testing time and computation time with and

without power constraint for the example system d695 with two TAMs. . . 90
Table 5.13 The comparison of SOC testing time and computation time with and
without power constraint for the example system d695 with three TAMs. . . 90

LIST OF FIGURES

Page

Figure 1.1 An example of System-on-a-Chip (SOC). 4
Figure 2.1 Picture of an example System-on-a-Chip (SOC) design and testing. . . 23
Figure 2.2 An example of a generic core-based system with one external test bus, and shared and dedicated BIST logic for the cores. 23
Figure 2.3 A single-layer neural network. 24
Figure 2.4 A neural unit in the neural network. 24
Figure 2.5 The proposed Neural Network. 25
Figure 2.6 The proposed structure of a general constraint (PC, RC, CC) unit. . . . 26
Figure 2.7 The proposed structure of a power constraint (PoC) unit. 26
Figure 2.8 The proposed searching and optimizing algorithm for SOC test scheduling. 27
Figure 2.9 The SOC System S_1 with 8 cores. 28
Figure 2.10 The best result test schedule for the system S_1, with only RC and CC constraints. 29
Figure 2.11 The best result test schedule for the system S_1, with RC, CC and PC constraints. 29
Figure 2.12 A feasible test schedule for the system S_1, with all the constraints RC, CC, PC, and PoC. 30
Figure 2.13 The best result test schedule for the system S_1, with all the constraints RC, CC, PC, and PoC after eliminating the idle time by the proposed method. 30
Figure 2.14 The second example of the SOC system S_2 with 13 cores. 31
Figure 2.15 The best result test schedule for the system S_2 with all the constraints RC, CC, PC, PoC. 32
Figure 2.16 The third example of the SOC system S_3 with 20 cores. 33
Figure 2.17 The best result test schedule for the system S_3 with all the constraints RC,

CC, PC, and PoC. 34
Figure 3.1 An example of core-based SOC system with two TAM lines. 47
Figure 3.2 An example of general maximum neural network. 48
Figure 3.3 The relation of the computation time and the SOC model complexity (Using the traditional method to solve NP-complete problems). 48
Figure 3.4 The proposed neural network architecture. 49
Figure 3.5 The proposed searching and optimizing algorithm for SOC test design. . 49
Figure 3.6 TAM design and test scheduling for the system S_4 with three TAMs and the total TAM width of 44. 52
Figure 4.1 Embedded core test infrastructure for SOC testing. 65
Figure 4.2 An example of a core with wrapper. 65
Figure 4.3 An example of wrapper scan chain design by the PTI and LPT algorithm. . 66
Figure 4.4 The proposed maximum neural network (MNN) architecture. 67
Figure 4.5 The proposed MNN searching and optimizing algorithm for SOC test design. 68
Figure 4.6 Test Access Mechanism (TAM) design for the SOC system d695 with three TAMs and total TAM width W is equal to 28 bits. 70
Figure 5.1 TAM design and test scheduling for the system S_4 with two TAMs and the total TAM width of 20. 75
Figure 5.2 The comparison of SOC testing time for SOC system S_4 with two TAMs using different methods 76
Figure 5.3 The relationship of the total SOC testing time and the number of TAM lines and the total TAM width. 78
Figure 5.4 The comparison of SOC testing time for SOC d695 with two TAMs using Integer Linear Programming (ILP), $P_{PAW\text{-enumerate}}$, and the proposed MNN. . 86
Figure 5.5 The comparison of the computation time of different methods when the number of SOC cores is increased. 87
Figure 5.6 The comparison of the computation time of different methods when the number of TAM width is increased. 87

Figure 5.7 The SOC testing time (objective) and the computation time of the proposed MNN when the number of iterations varies. 88
Figure 5.8 The comparison of SOC testing time for the example system d695 with three TAMs using Integer Linear Programming (ILP), $P_{PAW\text{-enumerate}}$, ECTSPSol, and the proposed MNN. 89
Figure 5.9 The comparison of SOC testing time for the example system p93791 with three TAMs using $P_{PAW\text{-enumerate}}$, ECTSPSol, and the proposed MNN. 89
Figure 5.10 The comparison of SOC testing time and computation time for the example system d695 with and without power constraints. 91

CHAPTER 1

INTRODUCTION

Currently, the advances in semiconductor technology have led to more complex *System-on-a-Chip* (SOC) designs. The SOC sizes range from 20-50 million transistors, with integrated logic, dynamic random access memory (DRAM), and analog [Shaikh 00]. Because of the high complexity and high density of the SOC system, testing SOC problems becomes crucial. In this research, we investigate new soft computing tools to solve the complex SOC test automation problems. In the following sections, System-on-a-Chip design and testing are introduced.

1.1 System-on-a-Chip (SOC) Testing

In the semiconductor industry, a new system design, called *System-on-a-Chip* (*SOC*) design, is currently being introduced to use multiple embedded modules built on a single chip. With today's technology, a single chip can consist of millions of transistors or components [Chauhan 00, Aikyo 00]. Figure 1.1 shows an example of SOC. To design a SOC system on a single chip, a designer often uses pre-designed, reusable mega cells known as *cores* in the SOC design [Chandramouli 96]. Embedding the cores onto SOC increases the width of the system bus and thus increases overall system performance, i.e., it can offer higher speed and lower power consumption [Daeje 98, Shubat 01]. A *core* can be defined as a complex piece of reusable module design such as microprocessors, bus interface, and memories [Crouch 99]. Cores are usually provided by the core providers and are treated as *intellectual property* (IP) so the detailed designs inside the cores are unknown to the SOC system designers. Due to the intellectual property (IP) issue, a core is provided to the SOC designer/integrator as a "black box" with predefined input/outputs and functional specifications. For SOC testing, a core provider provides a modular core with pre-computed tests for possible manufacturing defects. In the traditional system-on-board, a chip is

designed, manufactured, and tested by a component provider before sending it to a system integrator. Unlike system-on-board, a core is only a description of a module and is not yet manufactured when it is sent to a system integrator. A core integrator receives cores from each core provider and integrates cores at the chip-level and tests them using SOC testing mechanisms [Rajsuman 00]. Because of the design complexity and the huge number of components in the System-on-a-Chip design, testing of the SOC design becomes critical for the semiconductor industry.

Zorian et al. introduced generic conceptual test access architecture for embedded core [Zorian 99]. There are three elements in the embedded core test infrastructure; (i) test pattern source and sink, (ii) *test access mechanism* (TAM), and (iii) *core test wrapper*. A test pattern source generates test stimuli for an embedded core. A test pattern sink compares the responses from an embedded core to the expected responses. A test pattern source and sink can be designed either off-chip, using external *automatic test equipment* (ATE), or on-chip, using *built-in-self test* (BIST), or a combination of both [Zorian 99]. A BIST provides better accuracy and performance-related defect coverage, but it also increases the silicon area. The test patterns generated from a test pattern source are transported by test access mechanism (TAM) to a core under test. A TAM also transports the test stimuli from a core under test to a test pattern sink. A core test wrapper is an interface between a core and the system in which the core is embedded. The wrapper provides the switching between normal functional access and test access via the TAM [Marinissen 00].

When an SOC system designer/integrator gets cores from a core provider, he/she encounters two major tasks in solving the SOC test problems. The first task is how to design a *test access mechanism* (TAM) and the second task is how to find the test schedule for a SOC system to minimize the time-to-market. TAMs must be designed to transport the pre-computed tests from system I/Os to core I/Os [Iyengar 01b]. Test scheduling determines the order in which the various cores are tested and what testing resources are used for SOC testing subject to variety of hardware, capacity and sequence constraints. In this dissertation, we focus our investigation on solving the test scheduling problems for SOC test automation.

1.2 Problems Description

In this dissertation, three major research issues of the SOC test automation are investigated. First, by considering a SOC system consisting of main test resources such as external test and built-in-self-test (BIST), the SOC test scheduling is studied to minimize the SOC testing time subject to different constraints: (i) precedence constraint, (ii) resource constraint, (iii) core constraint, and (iv) power constraint. Second, a maximum neural network (MNN) is proposed to solve the test resource allocation problems for SOC. Third, after core wrapper design, the SOC test automation problems with resource allocation are studied to minimize the total SOC testing time. In this research, developing soft computing techniques of an unsupervised maximum neural network (MNN) are proposed to optimize the overall SOC testing time with core wrapper design and optimal resource allocation.

1.3 Outline of Dissertation

This research is organized as follows: In Chapter 2, the modeling and the soft computing techniques to solve the System-on-a-Chip (SOC) test automation problems are investigated. After the formulation of the SOC test automation problems, a neural network (NN) combined with heuristic random search methods is proposed to minimize the testing cost that occurs during SOC testing. In Chapter 3, an unsupervised maximum neural network is proposed to solve resource allocation problems for the SOC test automation. In Chapter 4, a maximum neural network is also proposed to solve the complex SOC test automation with core wrapper design. In Chapter 5, computer implementation of the proposed methods and the results of benchmarking SOC examples are presented. Finally, concluding remarks and future research are discussed in Chapter 6.

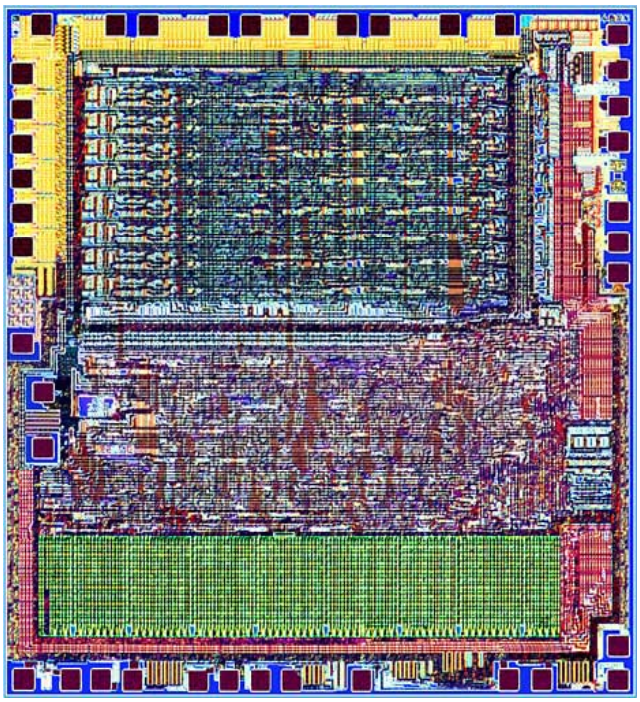


Figure 1.1 An example of System-on-a-Chip (SOC) [Temple 02]

CHAPTER 2

USING NEURAL NETWORK WITH FIXED-WEIGHT NET IN MODELING AND SOLVING THE SYSTEM-ON-A-CHIP (SOC) TEST SCHEDULING

This chapter presents the modeling and the proposed solution approach for solving the new *System-on-a-Chip* (SOC) design test scheduling problems. To solve the SOC test design problems, a neural network (NN) combined with heuristic algorithm has been developed. The SOC design test scheduling and optimization are subject to four different constraints: (i) precedence constraint, (ii) resource constraint, (iii) core constraint, and (iv) power constraint. The results demonstrate that the developed model with the soft computing techniques can successfully solve a large size SOC test scheduling problem within a reasonable time. The techniques presented in this chapter can be used for the optimization of the SOC design testing that is important for current development in the semiconductor and electronics industry.

2.1 Introduction

In *System-on-a-Chip* (SOC) test design, several test methods can be used, which include external-test, scan-test and *Built-in-Self-Test* (BIST) [Aerts 98]. BIST is a new testing method, and it is defined as an embedded ability of a core (i.e., a circuit or system) to test itself [Fausett 94, Chakrabarty 00c]. BIST can detect many faults within a small number of testing cycles. However, it is difficult to achieve high fault coverage by using BIST alone; thus an *external-test* is usually used together with BIST to cover a wide range of defects in testing [Nourani 00]. Figure 2.1 shows an example of an SOC used in industry [Pino 96 96]. Figure 2.2 shows an example of a generic core-based testing system with the application of BIST in testing [Chakrabarty 00c]. In Figure 2.2, the example testing system consists of one external test bus and six cores. In SOC testing, a System-on-a-Chip (SOC) is tested only

once as part of the overall system-chip design, which is quite different from a conventional system-on-board (SOB) test that traditionally takes multiple tests [Zorian 98]. In Figure 2.2, cores 1, 2, 3 and 4 are tested using either external-test or BIST logic. Core 5 is tested using only BIST, and core 6 is tested using only external-test. In Figure 2.2, cores 1, 2 and 5 have their own dedicated BIST logics. On the other hand, cores 3 and 4 share a BIST logic.

When a SOC system designer/integrator gets cores from a core provider, he/she encounters two major tasks in solving the SOC test problems. The first task is how to design a *test access mechanism* (TAMs), and the second task is how to find the test scheduling for an SOC system to minimize the time-to-market. TAMs must be designed to transport the pre-computed tests from system I/Os to core I/Os [Iyengar 01b]. Test scheduling determines the order in which the various cores are tested and what testing resources are used for SOC testing subject to a variety of hardware, capacity and sequence constraints. There are some researchers studying the first issue of the TAM design, but very few are studying the SOC test scheduling problems [Chakrabarty 00c].

In this chapter, we focus on the research of finding the optimal solution for solving the SOC test scheduling problems. The objective of the SOC cores test scheduling is to minimize the total SOC testing time subject to the following constraints:

(i) *Resource conflicts* between cores that share the same test component (i.e., TAMs or BIST logics),

(ii) *Core conflicts* between test components that are used to test that core,

(iii) *Precedence* constraints among different tests, and

(iv) *Power* consumption constraints of the test system.

The precedence constraint is considered to first test a SOC core that is prone to failure. The resource and the core constraints are considered to avoid conflicts among test resources and cores, respectively. The power constraint is considered when several SOC cores are tested at the same time, requiring that the consumed power may not exceed the maximum allowed power rating of the system. There are researchers proposing some heuristic methods to minimize the SOC testing time. Sugihara (1998) proposed a test method

for minimizing core-based system LSI testing time based on a combination of both external-test and BIST for each core [Sugihara 98]. In the earlier work, Chakrabarty (2000) proposed a method using integer linear programming (ILP) to minimize the SOC testing time with various constraints without any redesign of the embedded cores [Chakrabarty 00a, Chakrabarty 00b, Chakrabarty 00c]. In Chakrabarty's (2000) work presented in [Chakrabarty 00a], a method of TESTRAIL was proposed as a test access mechanism (TAM), which can provide access to one or more cores. The results showed that these SOC test problems are NP-hard problems [Chakrabarty 00a, Chakrabarty 00c, Iyengar 01b].

When solving the SOC testing problems by using linear programming, as the size of problem increases, there is a polynomial growth in the number of constraints and variables [Flores 99]. Using heuristic methods does not guarantee the optimal solution when the size of problems increases [Dagli 94]. Since the search of the entire space is often intractable due to the number of possible solutions, finding the optimal solution becomes difficult to achieve.

In this chapter, we investigate the modeling and the soft computing techniques to solve the SOC test scheduling problems. After the formulation of the SOC design testing problems, a neural network (NN) combined with heuristic random search method (tabu search) is proposed to minimize the testing cost that occurred during SOC testing.

2.2 Neural Network (NN) for Adaptive Learning and Optimization

Recently, *neural networks* (NNs) have been used in solving scheduling problems. A neural network is a system constructed to mimic the functions of a brain. A neural network consists of a system of individual neural units with weighted interconnections, as shown in Figure 2.3. Each neural unit consists of nodes linked together with the associated weights between each node. There are several types of neural networks [Dagli 94]. Figure 2.3 shows one type of neural networks called single-layer neural network, which consists of only input units and output units. In Figure 2.3, an associated weight w_{ij} connects an input unit x_i with an output unit y_j. Each output unit y_j has a bias term a_{0j}. Figure 2.4 shows a neural unit in a neural network. As shown in Figure 2.4, a neuron receives a set of weight inputs that are

added and then passed through an *activation function* [Dagli 94, Lee 00]. In general, each neural unit consists of two parts: a linear *summation* S_j and a nonlinear *activation function* $f(S_j)$, and they can be formulated as follows (also shown in Figure 2.4) [Kartalopoulos 96]:

$$S_j = \sum_{i=1}^{I} \left(w_{ij} \times x_i\right) + a_{0j} \tag{2.1}$$

$$y_j = f(S_j) \tag{2.2}$$

where x_i is the input i to a neural unit, $i = 1, \ldots, I$;

y_j is the output j from the neuron unit, $j = 1,\ldots,J$;

w_{ij} is the weight associated with each input unit i and output unit j; and

a_{0j} is the bias value for an output j.

A neural network can be classified into three categories by different training types: supervised training, unsupervised training and fixed-weight nets [Fausett 94]. In a supervised training NN, the training is done by adjusting the weights according to a learning algorithm. The weights are adjusted until the difference of the actual outputs and the target outputs is less than or equal to the desired error. In an unsupervised training NN, the weights are adjusted without the use of target outputs. The outputs from an unsupervised NN are classified into sets without being compared to a desired output. In a fixed-weight NN, the weights are fixed and set to represent the constraints and the quantity to be maximized or minimized. Neural networks have been used in solving the job shop scheduling problems [Flores 99, Foo 88, Yu 97, Jain 98, Lee 00, Yang 00]. For example, Yang, et al. (2000) proposed a NN in solving job-shop scheduling problems that satisfy three constraints: a job constraint, a machine constraint and a precedence constraint [Yang 00]. In his paper, the precedence constraint is among the operations of the same job and a starting time unit considers release time and due date of each job.

Neural networks are used to solve the optimization problems primarily due to their simplicity and the ability to learn and adapt. The speed and robustness of neural networks have made them very attractive in solving constraint satisfaction and optimization problems

[Flores 99]. However, using the neural network approach alone may not get the optimal solution [Jain 98]. This is due to its lack of generic pattern between inputs and outputs in solving the optimization problems. When solving larger size optimization problems, using only neural networks has been identified to be insufficient [Jain 98, Lee 00]. Hybrid technologies that combine the neural network with other strategies such as tabu search or simulated annealing may lead to better solutions with a more reasonable computational time. Although neural networks may not be as effective as the conventional optimization methods, they offer advantages in dealing with large scales optimization problems [Lee 00, Sabuncuoglu 96]. They can quickly find near optimal solutions when solving large size optimization problems [Fausett 94, Lee 00].

In this chapter, a neural network combined with heuristic random search is proposed to solve the SOC core test scheduling problems. Details of how to model the SOC test scheduling problems by using the neural network approach are presented in the following sections.

2.3 Formulation of the SOC Core Test Scheduling Problems

In this chapter, we consider the problems in which the SOC core-based systems are tested by using either *external-testing*, *BIST* or *combination* of both test resources, as shown earlier in Figure 2.2. To achieve the high fault coverage during SOC testing, a combination of BIST and external-testing must be used as much as possible [Iyengar 01b]. To improve the resource efficiency, BIST logic can be either shared by several cores or dedicated for a specific core, as shown in Figure 2.2. The test sets for all the cores, which include both BIST and external core components, are given. The precedence constraints among cores are also known. To demonstrate the formulation, an example test resource consisting of one external test bus and several Built-in-Self-Tests (BISTs) is used for illustration, as shown in Figure 2.2. The SOC test scheduling problems can be formulated as follows:

Let $T = \{t_{11}, t_{12},..., t_{im}, ..., t_{IM}\}$ denote the set of start times for the test patterns (external-test or BIST), which is applied to the core i of the SOC system, $i = 1,\ldots,I$, and m is

the test resource (for example, either the external-test bus or BIST), $m = 1 \ldots M$. Let L = $\{l_{11}, l_{12}, .., l_{im}, ..., l_{IM}\}$ denote the set of corresponding test lengths for the test sets. The objective is to find the shortest total testing time for the core-based SOC test design.

To find a feasible SOC test, the conflicts of the resource and core constraints need to be solved. Test sets can be conflicting, if: (i) they share an external test bus at the same time, (ii) they are BIST test sets for cores that share the same BIST resource, or (iii) they are the external and BIST components of the same core's test set. The conflict between cores i and j tested on the same test resource m will not occur if and only if either (i) $t_{im} - t_{jm} - l_{jm} \geq 0$, or (ii) $t_{jm} - t_{im} - l_{im} \geq 0$. This means that the testing periods of both cores (i, j) on the same test resource (m) cannot be overlapped. On the other hand, the conflict between the test resources m and k for the same core i will not occur if and only if the test periods of core i on the different test resources (m, k) are not overlapped, which is either (i) $t_{ik} - t_{im} - l_{im} \geq 0$, or (ii) $t_{im} - t_{ik} - l_{ik} \geq 0$. The resource constraint and the core constraint are disjunctive constraints (i.e., multiple alternative constraints). The precedence among cores q and i can occur if $t_{qr} - t_{im} - l_{im} \geq 0$, where core i is tested before core q. To satisfy the power capacity constraint, the summation of all the power dissipation $\sum_{n=1}^{N} p_n$ (for cores that have the overlapped testing time) cannot exceed the maximum power rating (P_{max}) of the test system. When the SOC test problems are solved, these constraints need to be satisfied [Kloypayan 02a, Kloypayan 02b].

The objective of the SOC test design modeling is to minimize the maximum of the test completion time, i.e., the summation of the starting time t_{im} and the total processing time l_{im}, subject to all the constraints. The modeling of the SOC core testing can be formulated as follows:

Objective: minimize $C = \max\{ t_{im} + l_{im} \}$ (2.3)

Subject to:

$t_{qr} - t_{im} - l_{im} \geq 0$, (precedence constraint)

$t_{im} - t_{jm} - l_{jm} \geq 0$, or $t_{jm} - t_{im} - l_{im} \geq 0$, (resource constraint)

$t_{im} - t_{ik} - l_{ik} \geq 0$, or $t_{ik} - t_{im} - l_{im} \geq 0$, (core constraint)

$w(\sum_{n=1}^{N} p_n) \leq P_{\max}$, (power constraint)

$$w = \begin{cases} 1 & if \quad \min_n\{t_n + l_n\} - \max_n\{t_n\} > 0 \\ 0 & otherwise \end{cases}, n = 1,...,N,$$

$t_{jm}, t_{ik}, t_{im} \geq 0$,

$l_{jm}, l_{ik}, l_{im} \geq 0$.

where t_{im} is the starting time of core i on a test resource m;

l_{im} is the processing time of core i on test resource m;

i, q and j represent a core , $i = 1 \ldots I$, $q = 1 \ldots I$, and $j = 1 \ldots I$;

m, k, r represent a test component, $m = 1,...,M$, $k = 1,...,M$ and $r = 1,...,M$;

p_n is the power dissipation that core n consumes while being tested, $n = 1,...,N$;

P_{max} is the maximum allowed power dissipation of the testing system; and

N is the number of cores that have the overlapped testing time, $N \geq 2$.

To solve the SOC core test scheduling problems, a neural network combined with a random search is proposed in this chapter. Details of how to construct the proposed NN and examples of solving the SOC test problems are presented in the following sections.

2.4 Constructing the Neural Network for Solving the SOC Testing Problems

Figure 2.5 shows the structure of the proposed neural network (NN). The proposed NN is a fixed-weight NN that consists of four blocks of neural units: (i) precedence constraint (PC) units, (ii) resource constraint (RC) units, (iii) core constraint (CC) units, and (iv) power constraint (PoC) units. As Figure 2.5 shows the first three units (PC, RC and CC)

have similar structures, but the fourth unit (PoC unit) has a different structure due to the complex effects of satisfying the power capacity constraint.

As shown in Figure 2.5, the NN receives the inputs of the starting time *ST* for each core. First, the precedence constraint (PC) units are processed. The PC unit considers the precedence among different cores, which are assumed to be known at the beginning. Then, the starting times of the PC units are adaptively adjusted and sent to the next resource constraint (RC) units, as shown in Figure 2.5. RC units consider that the same test resource cannot test two cores at a time. After the resource constraint is considered, the adjusted starting times from RC units are sent to the core constraint (CC) units, as shown in Figure 2.5. CC units ensure that the same core is not tested on different test resources at a time. After the starting times of all the cores have been adaptively adjusted by the CC units, they are forwarded to the power (PoC) units, as shown in Figure 2.5. Power constraint (PoC) units consider the power consumed by a set of cores having the testing time overlapped, which should not exceed the maximum allowed power rating of the SOC system. The adaptive adjusting procedure repeats until all the constraints have been satisfied, as shown in Figure 2.5. The output of the proposed NN is the set of the scheduled starting times for all the cores. Details of how to construct the constraint units (PC, RC, CC, and PoC units) are discussed in the following two sections.

2.4.1 Constructing the Precedence (PC), Resource (RC) and Core (CC) Constraints

As mentioned earlier in Section 2.2, a neural unit consists of two parts: a linear summation and a nonlinear activation function. In the proposed NN, the linear summation function (S_{NN_l}) and the nonlinear activator function (NN_l) for the constraint units (PC, RC or CC unit) are defined as follows:

$$S_{NN_l} = (w_a \cdot ST_{ex}(u-1)) + (w_b \cdot ST_{fy}(u-1)) + B_{NN_l} \tag{2.4}$$

$$NN_l = f(S_{NN_l}) = \begin{cases} 0 & , if \quad S_{NN_l} \geq 0 \\ S_{NN_l} & , if \quad S_{NN_l} < 0 \end{cases} \tag{2.5}$$

$$ST_{ex}(u) = w_c \cdot NN_l + ST_{ex}(u-1) \quad (2.6)$$

$$ST_{fy}(u) = w_d \cdot NN_l + ST_{fy}(u-1) \quad (2.7)$$

where ST_{ex} is the starting time of cores e on test resource x;

ST_{fy} is the starting time of cores f on test resource y;

u is number of iterations, $u = 1, 2, \ldots U$;

w_a, w_b, w_c, and w_d are the weights associated with each node; and

B_{NN_l} is a bias term.

Figure 2.6 shows a general neural unit of the PC unit, CC unit or RC unit for the proposed neural network. As shown in Figure 2.6, each neural unit takes inputs, which are the starting times $ST_{ex}(u-1)$ and $ST_{fy}(u-1)$ of a pair of the considering testing cores (e, f). As shown in Equations (2.4)-(2.7), the searching process of this neural unit works as follows. First, the summation function (S_{NN_l}) is calculated by using Equation (2.4) for the summation of the bias term (B_{NN_l}) with the multiple terms of starting times, $ST_{ex}(u\text{-}1)$ and $ST_{fy}(u\text{-}1)$, and their associated weights w_a and w_b. Then, the activator function NN_l is calculated, as shown in Figure 2.6. In Equation (2.5), the activator function NN_l is set to zero when the constraint $S_{NN_l} \geq 0$ is satisfied; otherwise NN_l is set to S_{NN_l}.

Equations (2.6) and (2.7) show that, when the given constraint is violated (i.e., $S_{NN_l} < 0$), the starting time $ST_{ex}(u\text{-}1)$ of a core e on the test component x is pushed by ($w_c - NN_l$) and the starting time $ST_{fy}(u\text{-}1)$ of a core f on the test component y is pushed by ($w_d - NN_l$). Figure 2.6 shows that, when the constraint is violated (i.e., $S_{NN_l} < 0$), the neural unit sends the adjusted weight w_c and w_d back to adaptively adjust the starting time $ST_{ex}(u\text{-}1)$ and $ST_{fy}(u\text{-}1)$ by using Equations (2.6) and (2.7). If the starting time $ST_{fy}(u)$ for core j is less than 0 (i.e., $ST_{fy}(u) < 0$), the starting time $ST_{fy}(u)$ is set to 0. The searching process continues until all the constraints are satisfied. Figure 2.6 shows that, after satisfying all the constraints, the outputs of the constraint unit are the adaptively adjusted starting times $ST_{ex}(U)$ and $ST_{fy}(U)$ for the pair of the considered testing cores (e, f).

In Equations (2.4)-(2.7), all the weights (w_a, w_b, w_c, w_d) and the bias term (B_{NN_l}) need to be determined according to the *precedence* constraint (PC), the *resource* constraint (RC) and the *core* constraint (CC), as defined earlier in Equation (2.3). Based on Equation (2.3), the correspondent weights (w_a, w_b, w_c, w_d) and the bias term (B_{NN_l}) for the precedence constraint (PC) units can be determined as follows:

For PC unit: $ST_{ex} \Leftarrow t_{qr}$, $ST_{fy} \Leftarrow t_{im}$, and

$$B_{NN_l} \Leftarrow -l_{im},\ w_a \Leftarrow 1,\ w_b \Leftarrow -1,\ w_c \Leftarrow -1,\ \text{and}\ w_d \Leftarrow 1. \tag{2.8}$$

For the resource constraint (RC) units, the correspondent weights (w_a, w_b, w_c, w_d) and the bias term (B_{NN_l}) are determined as follows:

For RC unit: $ST_{ex} \Leftarrow t_{im}$, $ST_{fy} \Leftarrow t_{jm}$, and

IF ($ST_{ex} \geq ST_{fy}$)

$$\text{THEN}\ \ B_{NN_i} \Leftarrow -l_{jm},\ w_a \Leftarrow 1,\ w_b \Leftarrow -1,\ w_c \Leftarrow -1,\ \text{and}\ w_d \Leftarrow 1; \tag{2.9}$$

IF ($ST_{ex} < ST_{fy}$)

$$\text{THEN}\ \ B_{NN_i} \Leftarrow -l_{im},\ w_a \Leftarrow -1,\ w_b \Leftarrow 1,\ w_c \Leftarrow 1,\ \text{and}\ w_d \Leftarrow -1. \tag{2.10}$$

For the core constraint (CC) units, the correspondent weights (w_a, w_b, w_c, w_d) and the bias term (B_{NN_l}) can be determined as follows:

For CC unit: $ST_{ex} \Leftarrow t_{im}$, $ST_{fy} \Leftarrow t_{ik}$, and

IF ($ST_{ex} \geq ST_{fy}$)

$$\text{THEN}\ \ B_{NN_i} \Leftarrow -l_{ik},\ w_a \Leftarrow 1,\ w_b \Leftarrow -1,\ w_c \Leftarrow -1,\ \text{and}\ w_d \Leftarrow 1; \tag{2.11}$$

IF ($ST_{ex} < ST_{fy}$)

$$\text{THEN}\ \ B_{NN_i} \Leftarrow -l_{im},\ w_a \Leftarrow -1,\ w_b \Leftarrow 1,\ w_c \Leftarrow 1,\ \text{and}\ w_d \Leftarrow -1. \tag{2.12}$$

In the next section, details are presented for the construction of the power constraint (PoC) unit.

2.4.2 Constructing the Power Constraint (PoC) Unit for the Neural Network

The last constraint for the SOC test model is the power capacity constraint (PoC unit) of the proposed NN. The power constraint (PoC) needs to be considered in the SOC model to limit test concurrency and to ensure that the power rating of the SOC is not exceeded. The concept of the power constraint (PoC) logic is to find groups of cores that have the testing period overlapped. In each group, the summation of the power dissipation of the same group is calculated. If the summation of the power of any group is greater than the maximum power dissipation capacity during testing, the start times of the chosen cores in that group need to be changed to different testing periods. The process continues until every group satisfies the power constraint, i.e., there is no group with the total power dissipation exceeding the maximum power capacity.

Figure 2.7 shows the structure of a power constraint PoC unit in the proposed NN. The number of input nodes is equal to the number of available test resources for a SOC system. As shown in Figure 2.7, the PoC unit takes the input [ST_i, t_i, P_i] of the starting times ST_i, the processing times t_i, and the power dissipation P_i for a group of the considered testing cores on different test resources. In Figure 2.7, the number J of the output nodes is set as $J = \left\lfloor C_2^M + C_3^M + C_4^M + ... + C_M^M \right\rfloor$, where M is the number of the available test resources in the SOC system. For each PoC unit in the proposed NN, the linear summation function (SP_j) and the nonlinear activator function (NP_j) are defined as follows:

$$SP_j = \sum_{i=1}^{M} a_{ij} P_i - P_{\max} \tag{2.13}$$

$$Diff_j = \min_{i=1}^{M}(ST_i(u-1) + t_i) - \max_{i=1}^{M}(ST_i(u-1)), \quad \text{for } \forall a_{ij} > 0 \tag{2.14}$$

$$PP_j = \begin{cases} 0 & , if \;\; Diff_j \le 0 \\ Diff_j & , if \;\; Diff_j > 0 \end{cases} \tag{2.15}$$

$$NP_j = f(SP_j) = \begin{cases} 0 & , if \;\; SP_j \le 0 \\ PP_j & , if \;\; SP_j > 0 \end{cases} \tag{2.16}$$

$$ST_i(u) = w_{ij} \cdot NP_j + ST_i(u-1) \tag{2.17}$$

where P_i is the power dissipation of the core tested on test resource i,

$i = 1,...,M$;

a_{ij} is the weights associated with input i and output node j, j = 1,..., J;

P_{max} is the maximum power dissipation allowed during testing;

t_i is the processing time of core on test resource i;

u is number of iterations, u = 1,…,U;

w_{ij} are the weights associated with each node, w_{ij} = -1, 0, or 1; and

ST_i is the starting time of a core on test component i.

After getting the inputs from the groups of cores on different test resources, the summation function SP_j and the nonlinear activator function NP_j are calculated by using Equations (2.13)-(2.17). If the power constraint is violated (i.e., $SP_j > 0$) and the group of inputs is overlapped (i.e., $Diff_j > 0$), the starting times ST_i of the first core and the last core of the input cores are adjusted by using Equation (2.17). The weight w_{ij} for the first core and the last core are set to 1 and –1, respectively. On the other hand, all the other weights w_{ij} of the input cores are set to zero, as defined in Equation (2.16). The procedure continues until all the cores satisfy the power capacity constraint in the SOC model, as defined earlier in Equation (2.3).

After all the constraints, i.e., the precedence constraint (PC), the resource constraint (RC), the core constraint (CC) and the power capacity constraint (PoC) have been constructed, they are integrated into the proposed NN for the SOC test system, as shown earlier in Figure 2.5. The complexity of the proposed NN with the different constraints can be found as follows. Assume there are a total of n cores and each core has at most 2 different test resources (i.e., external-test and BIST). The precedence constraint (PC) ensures the cores to be first tested on BIST before using external-test. In Equation (2.3), there are n sequence constraint inequalities for the precedence constraints, which requires n PC constraint units. For the resource constraint (RC) neural unit, the worse case scenario is that

there are only one external-test and one BIST. There are a total of $2*C^2_n$ sequence constraint inequalities, which require *2n(n-1)* RC neural units. For the core constraint (CC) neural unit, there are *n* sequence constraint inequalities, which require *n* CC units. The total number of interconnections in the proposed neural network is found to be $2n^2$. The connection complexity of the proposed neural network without power constraint is found to be $O(n^2)$. With the consideration of power constraint (PoC), the complexity of the power constraint unit is $O(n*2^M)$, where *M* is the number of the test resources and *M* is usually a small number. The complexity of the whole neural network is $O(\max\{n^2, n \cdot 2^M\})$. If we assume that the relation of *M* and *n* is $M = \sqrt{n}$, then the complexity of the proposed neural network is $O(2^{\sqrt{n}})$.

Using the proposed NN, feasible solutions can be found for the SOC test problems. However, the solution found by the NN may not be an optimal solution due to the limitation of neural network capabilities [Fausett 94]. To get an optimal solution, we use the constructed neural network combined with heuristic random search techniques. Details of the approach and the optimization method are discussed in the next section.

2.4.3 Conducting the Searching and Optimizing for SOC Test Systems

Figure 2.8 shows the proposed searching and optimizing algorithm for solving the SOC test problems. The proposed NN first processes the initial input to find a feasible solution. The initial input to the NN is a set of the initial starting times of each core at time *T* = 0. For the initial input, a random search technique (Tabu search) is used in this chapter to generate initial test cores for finding the feasible solutions. The order of cores and test components is first chosen randomly as the initial input to the NN. The searching is conducted by the developed NN until a feasible solution is reached by the network. During the searching by the NN, the generated feasible solution from each run is compared with the existing most-fit solution, as shown in Figure 2.8. Among all the feasible solutions, the one with the minimum completion time is selected. As shown in Figure 2.8, a random search is used to improve the feasible solutions during the search iterations. To prevent being trapped

at a local optimal, any input to the NN that has already been chosen will not be used again in the continuous search iterations.

As shown in Figure 2.8, the search is stopped when either one of the following conditions is met: (i) an optimal time has been reached, (ii) the number of iterations performed has exceeded the maximum number of allowed iterations, or (iii) the searching space has been exhausted. The condition (i) of the optimal time can be determined by using the lower bound of the SOC test model as shown in Equation (2.3). There are three different cases of the lower bound in the SOC testing. For the first case of SOC test without the precedence and the power constraints, the lower bound is equivalent to the summation of all the test time of the cores tested on the external-test resources. This is because all the tests on the external-test bus cannot overlap with each other, and the lower bound is equal to the summation of all the external-test times [Chakrabarty 00c]. For the second case of SOC test with the precedence constraint and without the power constraint, the lower bound is equivalent to the minimum test time on the precedent BIST tests plus the summation of all the core test times on the external-test resources. For the last case of SOC test with the power constraint, the lower bound is found to be the same as the second case, i.e., it is equivalent to the minimum test time on the precedent BIST tests plus the summation of all the core test times on the external-test resources.

As shown in Figure 2.8, a procedure of eliminating the elapsed time is enforced in the algorithm to delete any elapsed time between two tests on the same test resource. After the feasible solutions are generated by the NN, the idle time between any two adjacent tests is eliminated to shorten the completion time of the tests. The adjusted starting time is fed back to the NN for the next searching iteration until all the constraints are satisfied. As shown in Figure 2.8, the output from the NN is the set of resulting starting times and ending testing times of all the SOC cores after all the constraints are satisfied.

2.5 Computer Implementation and Examples

The proposed modeling, neural network and the optimization algorithm have been implemented on 800 MHz personal computers using MATLAB® software. Several industry SOC testing examples are used for demonstration of the developed techniques.

Table 2.1 shows the first SOC test example S_1 presented in [Iyengar 01b], which considers only the resource (RC) and core (CC) constraints. As shown in Figure 2.9, the example SOC test system S_1 consists of eight different SOC cores (namely c880, c2670, c7552, s953, s5378, s1196, s13207 and s1238 in Table 2.1). All the cores, except core s13207, share the same external-test bus. Cores c2670, c7552 and s13207 have their own dedicated BIST logic. As shown in Figure 2.9, cores c880, s953, s5378, s1196 and s1238 share the same *built-in-self-test* (BIST) logic. In Figure 2.9, the testing cycle times for the external-test and BIST of each core are shown within parentheses, and the power dissipation (either on external-test or BIST) of each core is shown within brackets. In this chapter, according to the industry practice, it is assumed that the power dissipation on BIST is 10 times of that on the external-test. In this example of SOC test system, the maximum allowed power dissipation (P_{max}) is assumed to be 750 mW.

Figure 2.10 shows the best solution of SOC test schedule for the example system S_1 considering only the resource (RC) and the core (CC) constraints. The optimal schedule is found to be 6809 clock cycles, which is the same as reported in [Iyengar 01b]. On finding the optimal solution, the developed NN took only 1.07 CPU seconds, compared to 3 CPU seconds by the mixed integer linear programming method (MILP) in [Iyengar 01b]. Figure 2.11 shows the optimal test schedule for the example system S_1 by adding the precedence (PC) constraint of cores being tested on BIST before external-test. As shown in Figure 2.11, the optimal test schedule in this case is 7065 clock cycles (the same as reported in [Iyengar 01b]), which is larger than that of Figure 2.10 due to the added precedence (PC) constraint. The developed NN takes only 1.89 CPU seconds, which is much less than 90 CPU seconds by the MILP in [Iyengar 01b].

So far, we have not considered the power constraint PoC in the example system S_1. Figure 2.12 shows a feasible test schedule first found by the developed NN for the example system S_1, with the consideration of all the constraints (RC, CC, PC, and PoC constraints). Due to the added power constraint PoC, the completion time of the test schedule is 8192 clock cycles, which is larger than the completion time of the example shown in Figure 2.11. Notice that, in Figure 2.12, idle time exists in the feasible test schedule, which can be further optimized. The idle time gaps in the feasible test schedule may occur when the solution space is very big. The developed NN eliminates the idle time gaps during its optimization process and searches for globally optimal test solution. Figure 2.13 shows the optimal test schedule generated by the developed NN for the example test system S_1. As shown in Figure 2.13, eliminating the idle time by the developed NN, the completion time of the optimal test schedule for system S_1 is found to be 7065 clock cycles which is much better than the original result in Figure 2.12.

Figure 2.14 shows the second example of the SOC test system S_2 with thirteen cores, one external-test bus and four BIST logics. Detailed data of the second example system S_2 are listed (as the first thirteen cores) in Table 2.2. As shown in Figure 2.14, three groups of cores share three BIST logics, and core 7 has its own dedicated BIST. Except for cores 7 and 11, all the other cores are accessible to the external-test bus, as shown in Figure 2.14. The maximum allowed power dissipation for the example system S_2 is assumed to be 750 mW. Figure 2.15 shows the optimal test schedule generated by the developed method for the example of SOC system S_2. As shown in Figure 2.15, the optimal test schedule is 10713 clock cycles. The optimal SOC test solution for the developed NN takes 26.61 CPU seconds for the example system S_2, compared to 142 CPU seconds of a similar system with 12 cores by the MILP method reported earlier in [Iyengar 01b].

When the size of the SOC problems gets bigger, the computation time by the existing MILP method grows exponentially [Iyengar 01b]. By using the developed NN method, the optimal solution of a larger system (> 20 cores) can be generated within reasonable computing time. Figure 2.16 shows the third example SOC system S_3. The example system

S_3 is considered as a large size SOC system, and S_3 consists of twenty cores, one external-test bus and six BIST logics, as shown in Figure 2.16. Detailed data of all the twenty cores of the testing system S_3 are shown in Table 2.2 (cores 1 to 20). In the SOC system S_3, five groups of cores share five BIST logics, and core 19 has its own dedicated BIST, as shown in Figure 2.16. In the test system S_3, except for cores 7, 11, 14, 18 and 19, all the other cores are accessible to the external-test bus. Figure 2.17 shows the optimal SOC test solution of 14,990 clock cycles generated by the developed NN for the example system S_3. Notice that, in Figure 2.17, the optimal SOC test schedule found by the NN is the optimal result and the completion time is equivalent to the lower bound of the test system S_3. By using the developed method, the computation time for the large size system S_3 is 750 CPU seconds (using MATLAB® program).

Table 2.3 shows the summary and comparison of the computation results generated by the developed NN method (after twenty runs of searching iterations) for the example systems S_1, S_2 and S_3 with different constraints. As shown in Table 2.3, each example system is tested with four combinations of constraints: (i) resource constraint and core constraint, (ii) resource constraint, core constraint and precedence constraint, (iii) resource constraint, core constraint, precedence constraint, and power constraint that considers only BIST resource, and (iv) similar to (iii) but considering both the external-test and BISTs. As shown in Table 2.3, for twenty runs, the proposed NN finds nine out of twelve testing problems with the best solutions (the lower bound). The examples and Table 2.3 show that the proposed NN method is not only capable of solving the large size SOC test problems but also able to find the optimal solutions within reasonable time. Except for some very large size SOC problems (20 cores), the proposed method generates the optimal SOC test solutions within efficient computing time, as shown in Table 2.3.

2.6 Summary

In this chapter, the modeling of System-on-a-Chip (SOC) test optimization has been formulated with different resource, capacity and precedence constraints. A neural network combined with heuristic algorithm has been developed to solve the large size SOC design

testing problems. Computer implementation and examples are presented. As demonstrated by the results from the testing examples, the developed method can not only solve the large size SOC test problems, but is also capable of finding the optimal solutions within reasonable computing time.

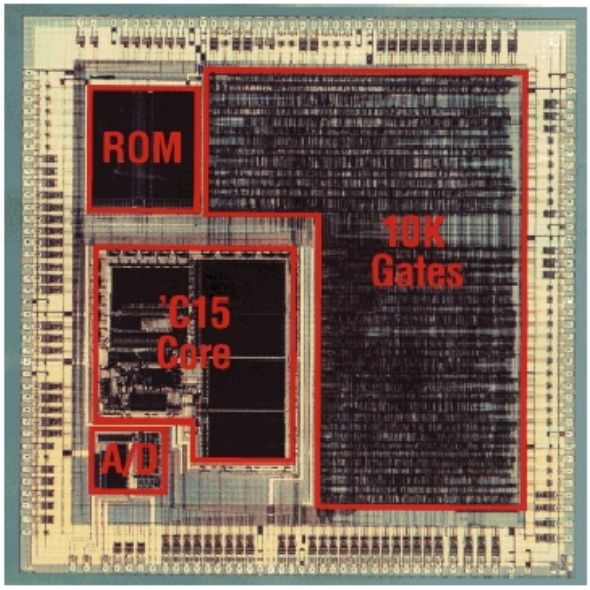


Figure 2.1 Picture of an example System-on-a-Chip (SOC) design and testing [Pino 96]

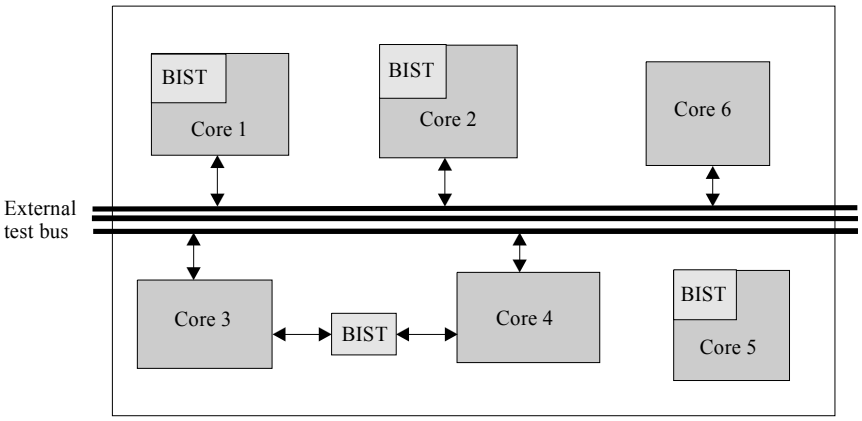


Figure 2.2 An example of a generic core-based system with one external test bus, and shared and dedicated BIST logic for the cores [Chakrabarty 00c]

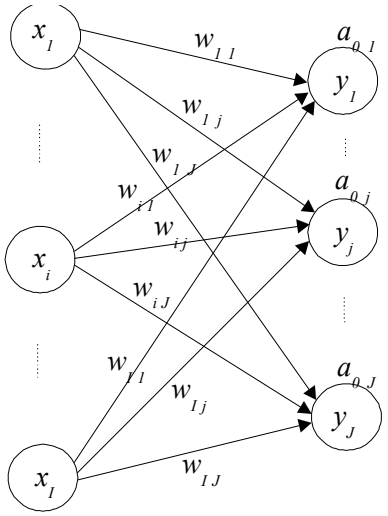


Figure 2.3 A single-layer neural network

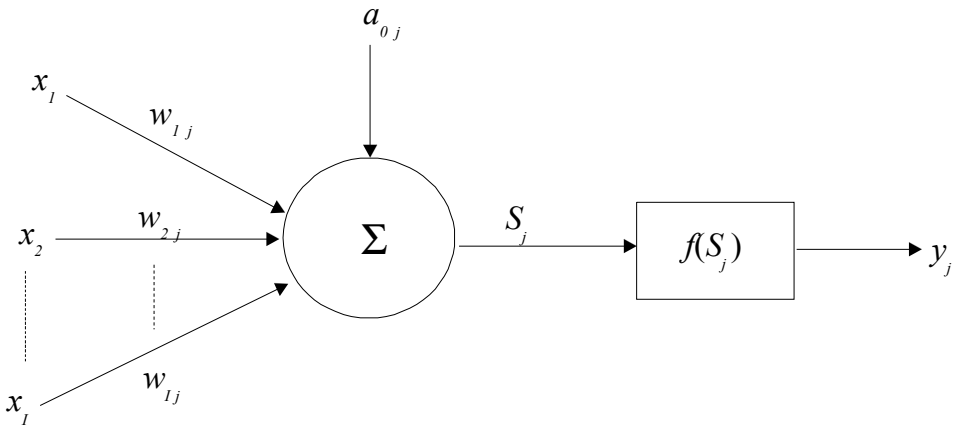


Figure 2.4 A neural unit in the neural network

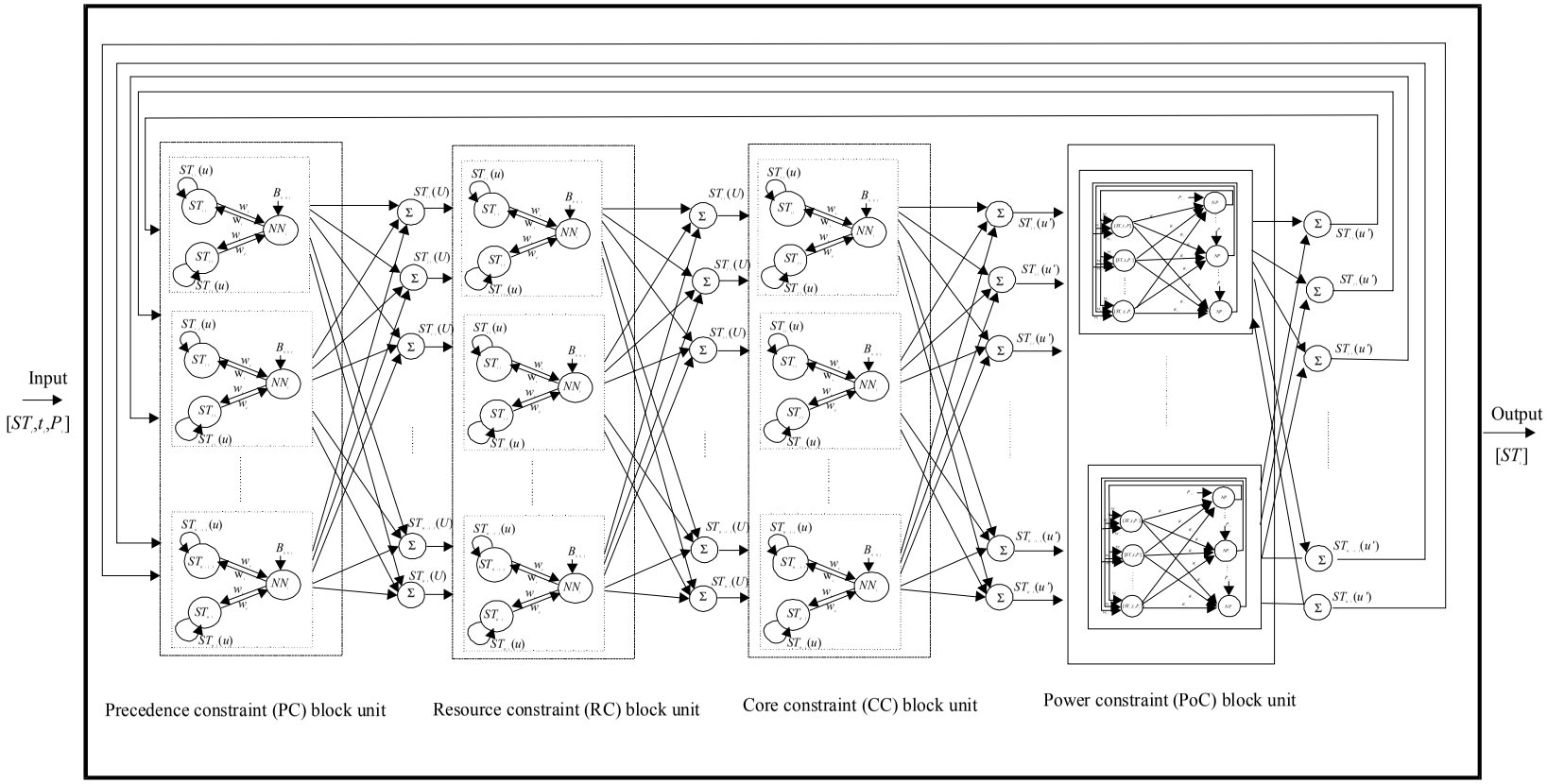


Figure 2.5 The proposed neural network

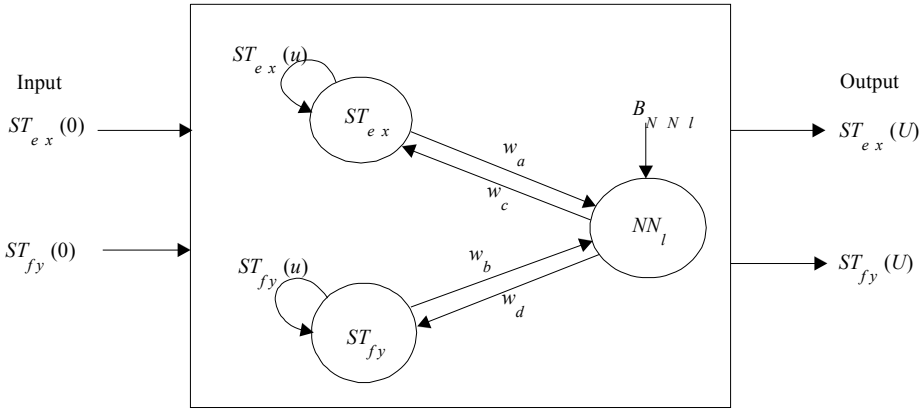


Figure 2.6 The proposed structure of a general constraint (PC, RC, CC) unit

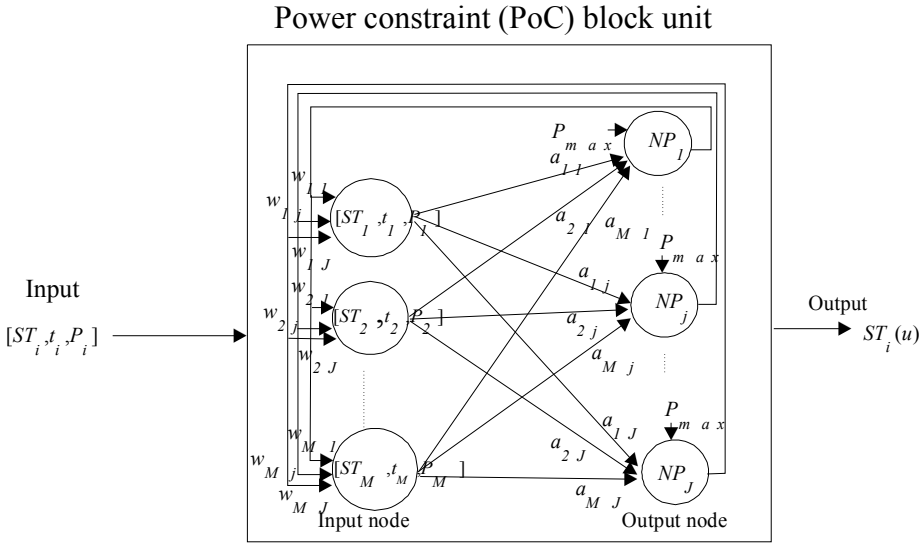


Figure 2.7 The proposed structure of a power constraint (PoC) unit

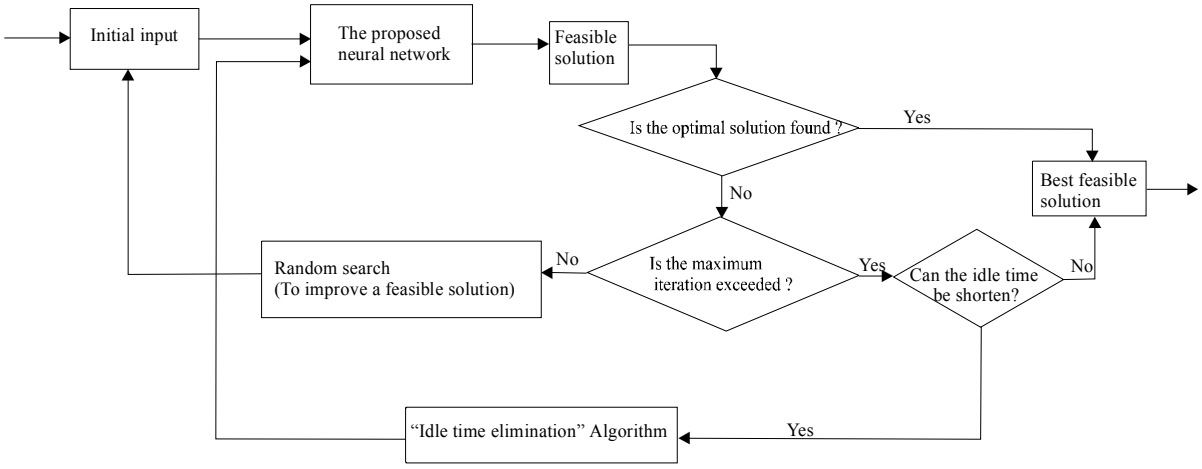
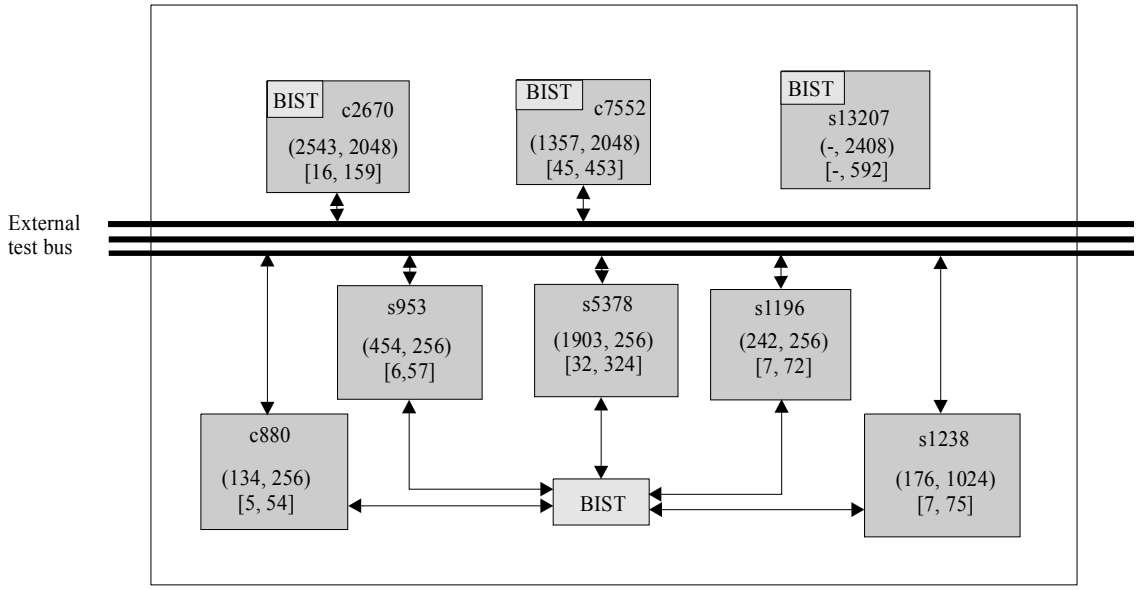


Figure 2.8 The proposed searching and optimizing algorithm for SOC test scheduling

Table 2.1 Test data of the cores for the SOC system S_1 [Iyengar 01b]

Circuit (core)	Core index i	Number of scan element, s	Number of scan patterns	Number of scan cycles	Power (P_i, mW) (External Test)	Number of BIST patterns	Number of BIST cycles	Power (P_i, mW) (BIST)
c880	1	60	26	134	5	4096	256	54
c2670	2	233	158	2543	16	32758	2048	159
c7552	3	207	96	1357	45	32768	2048	453
s953	4	52	90	454	6	4096	256	57
s5378	5	228	118	1903	32	4096	256	324
s1196	6	32	80	242	7	4096	256	72
s13207	7	790	-	-	-	32768	2048	592
s1238	8	32	58	176	7	16384	1024	75



( ) : (Testing time cycle for external test, Testing cycle for BIST test)
[ ] : [Power consumed while tested on external test, Power consumed while tested on BIST]

Figure 2.9 The SOC system S_1 with 8 cores

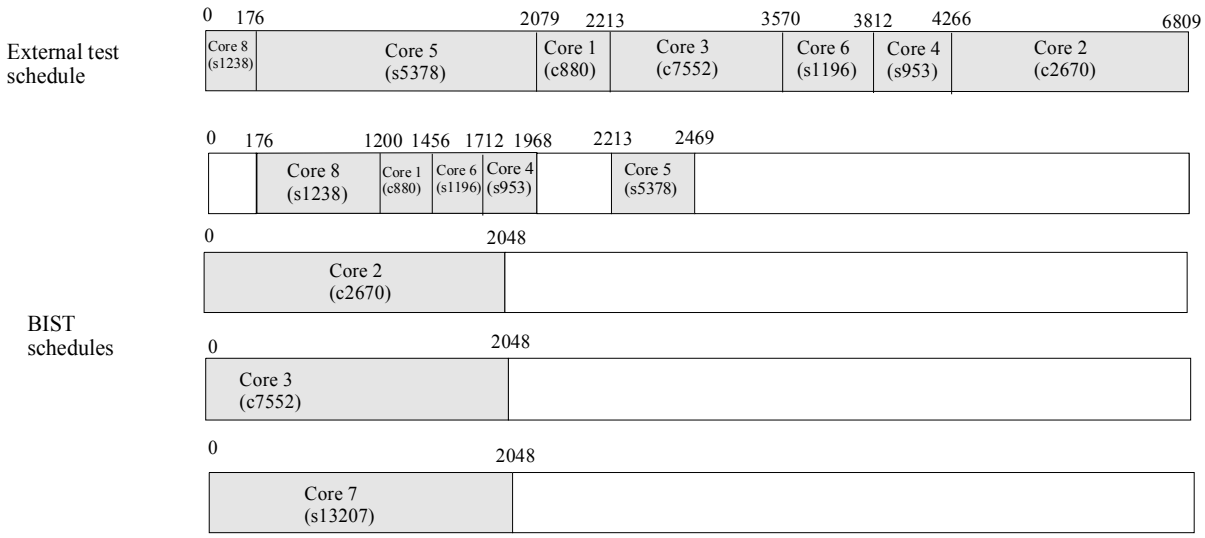


Figure 2.10 The best result of test schedule for the system S_1 with only RC and CC constraints (Computing time = 1.07 CPU sec< 3 CPU sec by using MILP [Iyengar 01b])

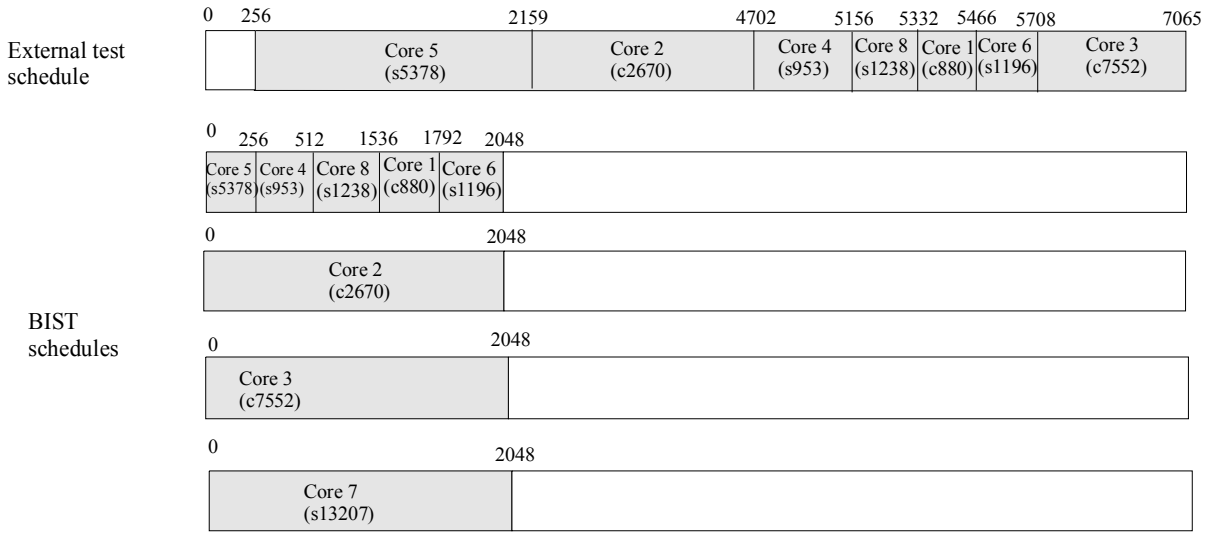


Figure 2.11 The best result of test schedule for the system S_1 with RC, CC and PC constraints (Computing time 1.89 CPU sec < 90 CPU sec by using MILP [Iyengar 01b])

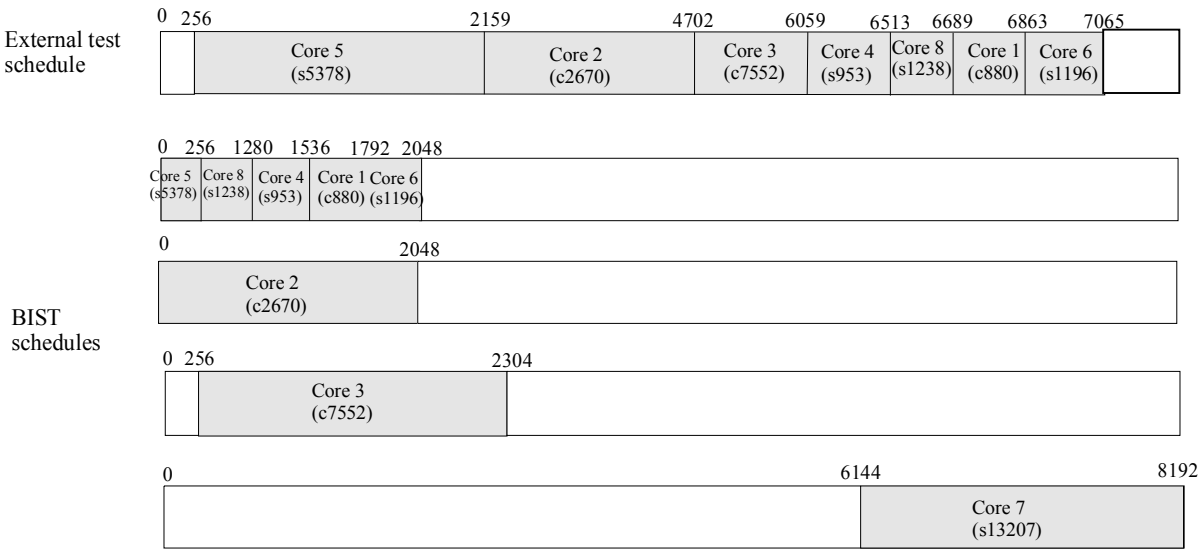


Figure 2.12 A feasible test schedule for the system S_1 with all the constraints RC, CC, PC, and PoC
(The feasible test schedule has some idle time that should be eleminated)
(The completion time = 8192 clock cycles)

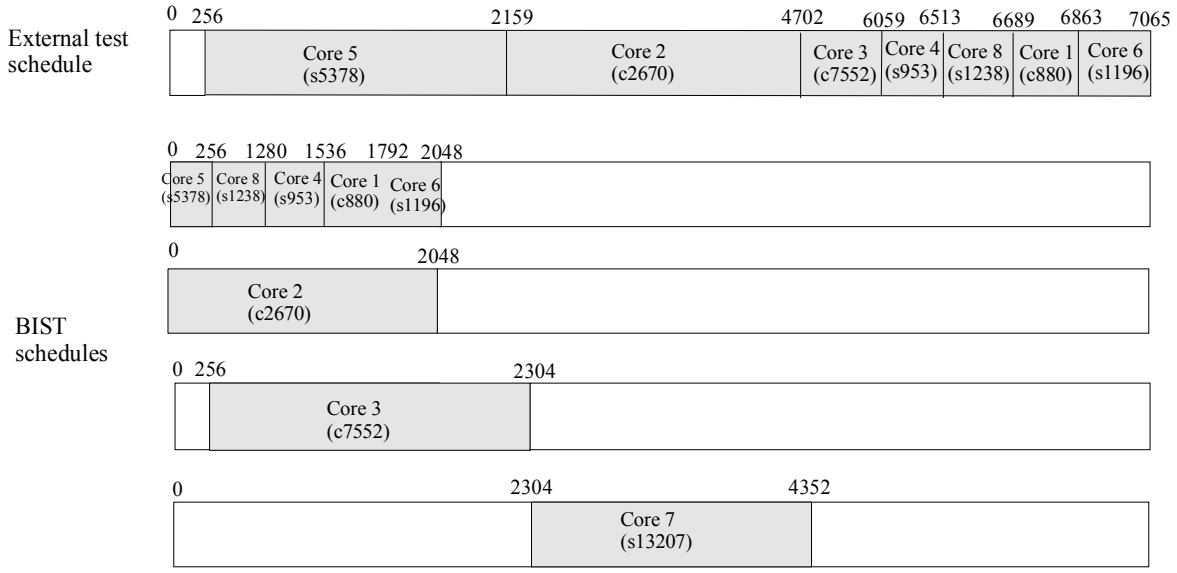
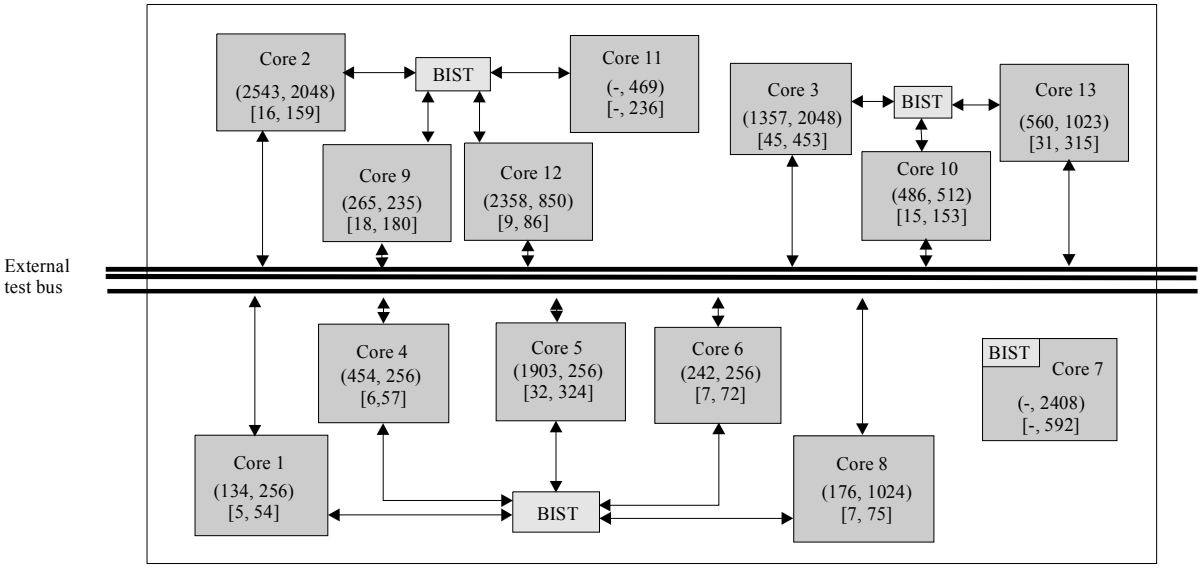


Figure 2.13 The best result of test schedule for the system S_1 with all the constraints RC, CC, PC, and PoC after eliminating the idle time by the proposed method
(The optimal schedule is 7065 cycles)



( ) : (Testing time cycle for external test, Testing cycle for BIST test)
[ ] : [Power consumed while tested on external test, Power consumed while tested on BIST]

Figure 2.14 The second example of the SOC system S_2 with 13 cores

Table 2.2 Test data of the cores for the SOC system S_2 and S_3

Core index (i)	External test		BIST		Core index (i)	External test		BIST	
	Testing time (t_{im})	Power P_{im} (mW)	Testing time (t_{im})	Power P_{im} (mW)		Testing time (t_{im})	Power P_{im} (mW)	Testing time (t_{im})	Power P_{im} (mW)
1	134	5	256	54	11	-	-	469	236
2	2543	16	2048	159	12	2358	9	850	86
3	1357	45	2048	453	13	560	31	1023	315
4	454	6	256	57	14	-	-	1240	412
5	1903	32	256	324	15	512	41	342	415
6	242	7	256	72	16	1204	8	342	78
7	-	-	2048	592	17	1357	16	342	162
8	176	7	1024	75	18	-	-	1240	95
9	265	18	235	180	19	-	-	2048	365
10	486	15	512	153	20	1204	8	342	85

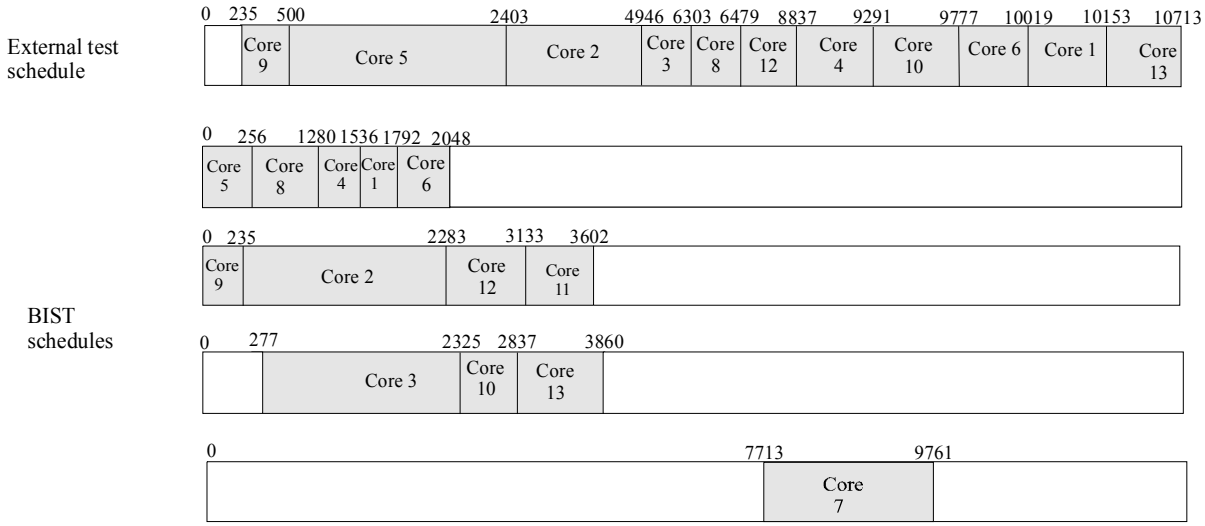
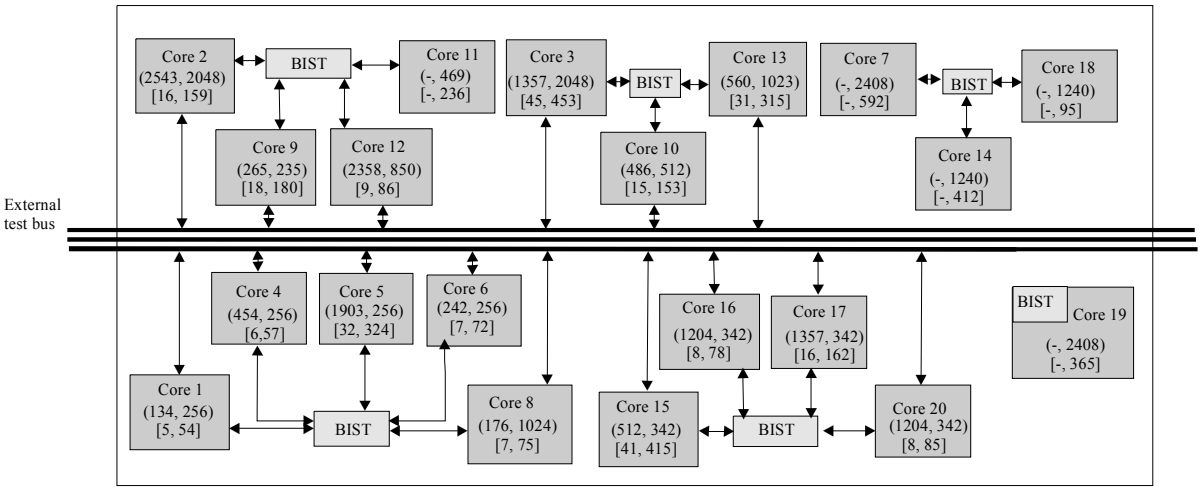


Figure 2.15 The best result of test schedule for the system S_2 with all the constraints RC, CC, PC and PoC
(The computation time is 26.6 CPU sec < 142 CPU sec by using MILP [Iyengar 01b])



( ) : (Testing time cycle for external test, Testing cycle for BIST test)
[ ] : [Power consumed while tested on external test, Power consumed while tested on BIST]

Figure 2.16 The third example of the SOC system S_3 with 20 cores

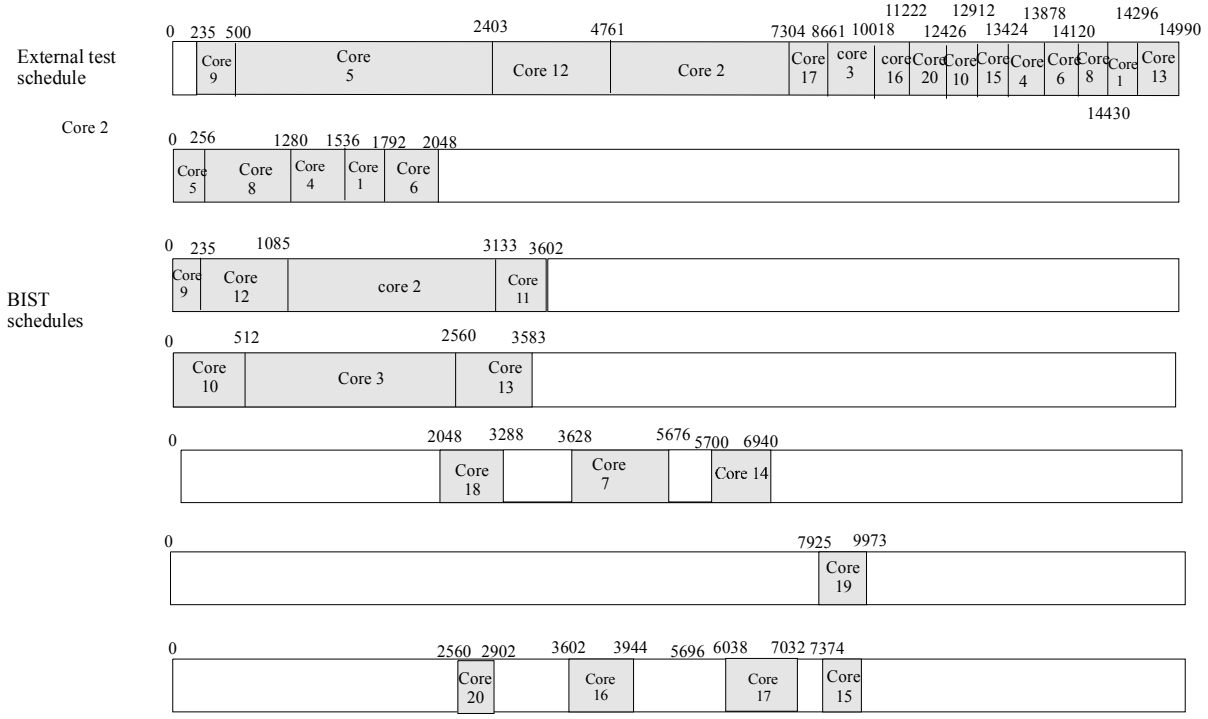


Figure 2.17 The best result of test schedule for the system S_3 with all the constraints RC, CC, PC, and PoC
(The optimal schedule is 14990 clock cycles)

Table 2.3 The SOC test scheduling solutions from the proposed NN after 20 runs

Problems	Lower bound	Sol_{best}	ΔSol %	Sol_{avg}	Computational time (second)		
					$T_{average}$	T_{max}	T_{mi}
8 cores with 2 constraints	6809	6809	0	-	1.07	3.68	0.27
8 cores with 3 constraints	7065	7065	0	-	1.89	5.93	0.38
8 cores with 4 constraints (BIST)	7065	8192	15.95	8192	65.48	94.31	50.92
8 cores with 4 constraints (ALL)	7065	8547	20.98	8619	357.64	440.94	294.45
13 cores with 2 constraints	10478	10478	0	-	1.93	5.28	0.88
13 cores with 3 constraints	10713	10713	0	-	10.23	34.43	1.04
13 cores with 4 constraints (BIST)	10713	10713	0	10859	26.61	95.8	2.5
13 cores with 4 constraints (ALL)	10713	10713	0	10889	326	1759.3	34.3
20 cores with 2 constraints	14755	14755	0	-	5.03	10.11	2.58
20 cores with 3 constraints	14990	14990	0	-	23.06	101.06	2.36
20 cores with 4 constraints (BIST)	14990	14990	0	15178	750	2551	141
20 cores with 4 constraints (ALL)**	14990	15369	2.53		20900	-	-

** run only one time.

Lower bound: optimal value from ILP method [Iyengar 01b]
Sol_{best}: value of the best solution found by the proposed NN out of 20 runs.
ΔSol%: = ((Sol_{best} - Opt)/Opt)*100 if the optimum is known,
= ((Sol_{best} - LB)/LB)*100 otherwise.
Sol_{avg} : average solution value over 20 runs or nothing if all runs gave the optimum value.
$T_{average}$: average computing time, in seconds.
T_{max}: maximum computing time for one run, in seconds
T_{min}: minimum computing time for one run, in seconds

CHAPTER 3

DEVELOPING MAXIMUM NEURAL NETWORK TO SOLVE TEST RESOURCE ALLOCATION AND TESTING TIME MINIMIZATION FOR THE SYSTEM-ON-A-CHIP (SOC)

This chapter presents a maximum neural network to minimize the testing time and optimize resource allocation for a *System-on-a-Chip* (SOC) test system design. By determining the allocation of cores to *test access mechanism* (TAM) and the TAMs width in a SOC system, the testing time can be reduced. Constraints considered in the SOC test system design include the following: (i) SOC cores allocation (ii) TAMs width selection, and (iii) power consumption. The objective of this investigation is to achieve the optimal testing time for a SOC test system design with the optimal core allocation and TAMs width selection within a reasonable computation time.

3.1 Introduction

System-on-a-Chip (SOC) integrated circuits are composed of a huge number of processors, memories, and peripheral interface devices in the form of embedded cores [Yu 01]. In the SOC design and development, a System-on-a-Chip (SOC) is tested only once as part of the overall system-chip design, which is quite different from a conventional system-on-board (SOB) test that traditionally takes multiple tests [Zorian 98]. For testing, a core provider provides a modular core with pre-computed tests for detecting possible manufacturing defects. Due to the design complexity and the huge number of components (may have more than a million components), dealing with the large amount of test data that can be transferred between the tester and the chips during testing becomes more challenging [Zorian 97].

When a SOC system designer/integrator gets cores from a core provider, he/she encounters two major tasks in solving the SOC test problems. The first task is how to design a test access mechanism (TAMs) and the second task is how to determine the test schedules for a SOC system to minimize the time-to-market [Iyengar 01b]. TAMs must be designed to transport pre-computed tests from system I/Os to core I/Os. In recent years, several new TAMs have been proposed such as Test Bus [Varma 98], and TESTRAIL [Marinissen 98]. Test scheduling determines the order in which the various cores are tested and what testing resources are used for SOC testing subject to variety of hardware, sequence and capacity constraints.

Figure 3.1 shows an example of a core-based SOC with two TAMs [Chakrabarty 00a]. The system has 10 cores: 2 combinational cores and 8 cores with internal scan. These cores must be allocated to each TAM and the TAM partition is considered such that the overall testing time is minimized. The SOC testing design with resource allocation problems are NP-complete [Chakrabarty 00a]. When a SOC system is small (less than 10 cores), using traditional methods such as integer linear programming (ILP) or a heuristic approach can solve the problems easily. However, when the system becomes more complex, using traditional methods may not be efficient or effective because the optimal solution is difficult to achieve and the computation time is long.

Several research works have been done assuming the TAM is already determined. Sugihara (1998) proposed a test method for minimizing core-based system LSI testing time based on a combination of both external-test and built-in-self test (BIST) for each core [Sugihara 98]. Chakrabarty (2000) proposed a method using integer linear programming (ILP) to minimize the SOC testing time with various constraints without redesigning the embedded cores [Chakrabarty 00c]. The results showed that these SOC test problems are NP-hard problems [Chakrabarty 00c]. In our earlier work, a neural network combined with heuristic algorithms has been developed to solve the SOC test scheduling problems. The proposed method can successfully solve large size SOC test scheduling problems within a reasonable amount of time [Kloypayan 02a, Kloypayan 02b].

Recently, the test resource allocation problems have been considered when solving the SOC test scheduling problems. Chakrabarty (2000) formulated the problem as an Integer Linear Programming (ILP) model assigning each core to a test bus in order to minimize the test time [Chakrabarty 00a, Chakrabarty 00b]. The study investigates how to allocate *N* test lines to a fixed number of test buses. The place-and-route and power constraints of SOC testing design were also considered in [Chakrabarty 00b]. A drawback of using an ILP model is that the computation time becomes very lengthy when the system becomes complex. Bagchi et al. (2000) studied the similar problems as in [Chakrabarty 00a, Chakrabarty 00b] by clustering the groups of cores into different modules for scheduling the testing time [Bagchi 01]. Yu et al. (2001) proposed a method of using 2-dimensional bin-packing (or rectangle packing) model to minimize the test application time while offering full scan/partial scan functional tests for different TAMs under the constraint of peak power consumption [Yu 01]. This method is general because it is not restricted to one specific TAM, and the optimal number of test buses is determined rather than leaving it fixed for core integrators. Ebadi et al. (2001) proposed a method of using a Genetic Algorithm to design the optimal test access architecture [Ebadi 01]. Iyengar et al. (2002) proposed a rectangle packing optimization algorithm to minimize the testing time by matching the appropriate core's test needs and the widths of the TAMs that it is assigned [Iyengar 02]. In this method, the number of wires to a TAM is not fixed. This method can reduce the computation time compared to the ILP-based technique and it can get a comparable testing time result. Without TAM partition, a core user has to pay more consideration to the wiring design.

In this chapter, a SOC system with an optimal partition of the total TAM width and an optimal core assignment to the TAMs are considered to minimize the overall testing time. In this chapter, a *maximum neural network* (MNN) is proposed to solve the SOC testing design with resource allocation problem. A maximum neural network (MNN) is an unsupervised competitive neural network and is used because a target output, which is the test bus assignment vector in this case, is difficult to be determined. In this chapter, the issues of an optimal partition of the total TAM width and the optimal core assignment to the TAMs

are considered to minimize the overall testing time. Power consumption is also considered in this chapter. Details of the network will be discussed in the next section.

This chapter is organized as follows. A general concept of the neural networks is introduced in Section 3.2. Section 3.3 presents the modeling of the SOC testing with resource allocation problems. Section 3.4 shows how to construct the maximum neural network for solving the SOC testing with resource allocation problems. Section 3.5 provides the summary.

3.2 A Maximum Neural Network (MNN) for Adaptive Learning and Optimization

Neural networks (NNs) have been used in solving scheduling and optimization problems recently. A neural network is a system constructed to mimic the functions of a brain, and consists of individual neural units with weighted interconnections. Each neural unit consists of nodes linked together with the associated weights between each node.

A neural network is a parallel, distributed information processing structure consisting of many processing elements interconnected via weighted connections. Neural networks can provide optimal solutions to difficult optimization problems, which the traditional heuristic methods cannot generally accomplish. Solving the optimization problems requires the minimization of some cost functions subject to a set of constraints. These cost functions are known as energy functions, and the neural network can produce good solutions by minimizing the energy function [Smeda 99]. In addition, the computation time for achieving the optimal solution is less than that of traditional optimization techniques [Sellers 96].

A neural network can be classified into three categories based on different learning rules: supervised learning, unsupervised learning and reinforced learning [Kartalopoulos 96]. In supervised learning, the training is done by adjusting the weights according to a learning algorithm. The weights are adjusted until the difference of the actual outputs and the target outputs is less than or equal to the desired error. In an unsupervised learning, the weights are adjusted without the use of target outputs. The outputs from an unsupervised NN are classified into different sets without being compared to a desired output. Reinforced

learning, unlike supervised learning, does not indicate how close the actual output is to the desired output, but it indicates whether the actual output is the same as the desired output. The error signal generated during the training session is binary: pass or fail.

In this chapter, an unsupervised learning NN called maximum neural network is proposed to optimize the resource allocation problems for SOC testing design. The maximum neural network can solve an NP-complete problem in a polynomial time [Takefuji 92, Kloypayan 02c]. The maximum neural network (MNN) is one of the unsupervised neural networks that can minimize a cost function considering various constraints. The operation of the maximum network is based on the group update. Figure 3.2 shows an example of a maximum neural network. In Figure 3.2, the network is composed of *n* groups where each group consists of *m* binary neurons. One and only one among *m* neurons with the maximum input per group is encouraged to be selected. The goal of the maximum neural network is to minimize the energy function *E* that represents the objective function and the constraints of the problem. The function of each neuron and the input states are determined as follows [Galan-Marin 01]:

$$x_{ij} = \begin{cases} 1 & if\ Y_{ij} = \max\{Y_{i1},....Y_{im}\} \\ 0 & otherwise \end{cases}, \tag{3.1}$$

$$Y_{ij}(u+1) = Y_{ij}(u) - \alpha \frac{dY_{ij}}{du}, \tag{3.2}$$

$$\frac{dY_{ij}}{du} = -\frac{\partial E(x_{11},..,x_{ij},..,x_{nm})}{\partial x_{ij}}, \tag{3.3}$$

where Y_{ij} and x_{ij} are the input and output of the neuron, respectively;

u is the number of iterations, $u = 1,\ldots U$;

α is a learning rate; and

E is the energy function of the problem.

For each iteration, the input Y_{ij} is updated by multiplying the learning rate α with the derivative of energy function E, as shown in Equations (3.2) and (3.3). The output x_{ij} for each group of inputs is determined by using Equation (3.1).

The objective function of a SOC testing system is to minimize the overall testing time subject to various constraints. The objective function considers the optimal allocation of cores to each TAM and the selection of TAM width for minimizing the SOC testing time. When the SOC testing system becomes more complex (i.e., increased number of cores, TAM line or the total TAM width), the traditional methods such as linear programming cannot efficiently find the optimal solution within a reasonable computation time or sometimes cannot effectively find the solution. In this chapter, a maximum neural network is proposed to find the optimal solution for the complex systems within reasonable computation time.

In the next section, the model of SOC testing with resource allocation problem is formulated. Then, a maximum neural network is applied to solve the problem, as discussed in Section 3.4.

3.3 Formulation of SOC Testing with Resource Allocation Problems

In this chapter, we assume that a core user has already determined the TAM design after careful consideration of system-level I/O, area and power issues. When the TAM has already been designed, a problem arises regarding the assigning of the cores into each TAM to minimize the overall testing time and to find the optimal width for each TAM, when the total of the TAM width is given.

Let T_{ij} represent the testing time of each core i on TAM_j. The testing time T_{ij} can be formulated as follows:

$$T_{ij} = \begin{cases} t_i & ,if\ \phi_i \le w_j \\ (\phi_i - w_j + 1)\cdot t_i & ,if\ \phi_i > w_j \end{cases}, \tag{3.4}$$

$$\phi_i = \max_i \{n_i, m_i\}, \tag{3.5}$$

$$t_i = (p_i + 1)\ f_i / N_i\ + p_i, \tag{3.6}$$

where ϕ_i is the test width of core *i*;

t_i is the number of test cycles for core *i*;

n_i is the number of test inputs of core *i*;

m_i is the number of test outputs of core *i*;

w_j is the width of TAM $_j$;

p_i is the number of test patterns of core *i*;

f_i is the number of flip-flops contained in core *i*; and

N_i is the number of internal scan chains for core *i*.

Let a SOC system have *I* cores and *J* TAM lines. The width of each TAM line is represented by w_j, which has to be less than or equal to w_{max}. The summation of the width w_j for all TAMs is equal to the total TAM width *W*. A power consumption for core *i* on TAM$_j$ under serialized test is $\frac{P_i}{\phi_i - w_j + 1}$. In this chapter, a power dissipation P_i and the maximum power of the SOC system Ω are given. We assume that the power allocated to each TAM is equal, which is $\frac{\Omega}{J}$. When a core *i* is tested on TAM$_j$, the power consumption should not exceed the allocated power of TAM$_j$ $\left(i.e., \frac{P_i}{\phi_i - w_j + 1} \le \frac{\Omega}{J} \right)$. To minimize a SOC testing time with resource allocation, the model can be formulated as follows:

$$\text{Objective:} \quad \text{Minimize } \{\max(\sum_{i=1}^{I} T_{ij} \cdot x_{ij})\} \quad , 1 \le j \le J \tag{3.7}$$

$$\text{Subject to} \quad \sum_{j=1}^{J} x_{ij} = 1, \ 1 \le i \le I \quad \text{(Resource Allocation)} \tag{3.8}$$

$$\sum_{j=1}^{J} w_j = W, \quad \text{(TAM width distribution)} \tag{3.9}$$

$$w_j \le w_{max}, \tag{3.10}$$

$$\frac{x_{ij} P_i}{\phi_i - w_j + 1} \le \frac{\Omega}{J}, \qquad \text{(Power Constraint)} \tag{3.11}$$

$$x_{ij} = 0, \text{ or } 1 \tag{3.12}$$

where T_{ij} is the testing time for core i on TAM$_j$, as shown in Equation (3.4),

$1 \le i \le I$, $1 \le j \le J$;

W is the total width of TAMs;

w_j is the width of TAM$_j$;

w_{max} is the maximum width of TAMs;

Ω is the maximum power rating in the SOC system;

P_i is the power consumption of core i; and

x_{ij} is the 0-1 variable defined as

$$x_{ij} = \begin{cases} 1, & \textit{if Core i is assigned to TAM j} \\ 0, & \textit{otherwise} \end{cases}.$$

The SOC testing with resource allocation problem is a NP-complete. Figure 3.3 shows the relation of the computation time and the model complexity when using the traditional method such as integer linear programming (ILP) to solve an NP-complete problem. When a system becomes more complicated, the computation time gets longer, as shown in Figure 3.3.

In this chapter, the maximum neural network is used. By using the maximum neural network, the near-optimal or good solution can be found within an acceptable time when a system becomes more complex. Next, the proposed method is applied to the SOC testing model, and the details of the results from the proposed method are provided.

3.4 Constructing the Maximum Neural Network (MNN) for Solving the SOC Testing Problems

Figure 3.4 shows the architecture of the proposed MNN. There are $J + 1$ input units: [Y_{ij}, j=1,…,J] and [w_j, j = 1,..,J], and I output units: [x_{ij}] for i= 1,…,I. Y_{ij} is an energy function, which is the testing cost of the SOC testing system design. In Figure 3.4, an output

x_{ij} defines that core i is assigned to TAM_j. By using Equations (3.7)-(3.12), each core i can be assigned only to one TAM line, TAM_j. In the maximum neural network, only one output is selected (i.e., in Figure 3.4, there is only one output x_{ij}, j=1…J, has a non-zero value and all others are zero). In the proposed SOC test model, an input Y_{ij} and an output x_{ij} can be calculated as follows:

$$x_{ij} = f(Y_{ij}) = \begin{cases} 1 & if\ Y_{ij} = \max\{Y_{ia}\}\ \ (a = 1,...,J) \\ 0 & otherwise \end{cases}, \tag{3.13}$$

$$Y_{ij}(u+1) = Y_{ij}(u) - \Delta Y_{ij}, \tag{3.14}$$

$$\Delta Y_{ij} = \alpha(u) \cdot T_{ij} + \gamma(u) \cdot \left(\frac{x_{ij} P_i}{\phi_i - w_j + 1} - \frac{\Omega}{J} \right), \tag{3.15}$$

$$w_j(u+1) = w_j(u) - \beta(u) \cdot (w_j(u) - w_{\max}),\ j = 1,..,J\text{-}1, \tag{3.16}$$

$$w_J(u+1) = W - \sum_{j=1}^{J-1} w_j(u+1), \tag{3.17}$$

where $\alpha(u)$, $\gamma(u)$ and $\beta(u)$ are learning rates;

u is number of iterations, $u = 1,\ldots,U$;

$w_j(u)$ is the width of TAM_j at iteration u, $1 \le j \le J$; and

T_{ij} is the testing time for core i on TAM_j determined by Equation (3.4), and $1 \le i \le I$, $1 \le j \le J$.

The learning rates α, β and γ may be decreased at each iteration or they may be constant throughout the learning process. The rate of decrease depends on the speed of convergence to the optimum solution or the predefined termination condition [Kartalopoulos 96].

The proposed MNN begins the first iteration by assigning the energy function Y_{ij} values and the width w_j for each TAM by generating a random number. The output x_{ij} can be calculated by using the function $f(Y_{ij})$, defined by Equation (3.13), as shown in Figure 3.5. Then, Y_{ij} and w_j for the next iteration are updated using Equations (3.15), (3.16) and (3.17), as shown in Figure 3.4. The searching procedure continues until either the optimal solution

has been reached or the termination conditions are met. The complete searching algorithm of the proposed MNN is shown as follows:

Step_1. Set $u = 0$.

Step_2. Assign each w_j by using uniform random number from 1 to W.

Step_3. Assign each Y_{ij} value by using uniform random number.

Step_4. Calculate x_{ij} by using Equation (3.12).

Step_5. Calculate the total testing time for system. If the new testing time is smaller than the old testing time and the power constraint is not violated, keep the new testing time as the minimum testing time.

Step_6. Update Y_{ij} and w_j for the next iteration (u+1).

Increment u by 1. If the state of the system reaches the equilibrium state or the terminate conditions are met, then stop this procedure. Otherwise, repeat to Step_4.

The termination condition used in this chapter includes the following two conditions: (i) the iteration number has reached a predefined number, or (ii) when the MNN system reaches a stable point. The procedure stops once either of these conditions is met.

Figure 3.5 shows the proposed searching algorithm for solving the SOC test with resource allocation problem. At the beginning, the initial input to the MNN is a set of the energy function and the width w_j of each TAM_j generated by using random numbers. The output x_{ij} from the MNN is for the SOC test resource allocation. If MNN reaches an optimal solution, the procedure stops; otherwise, the procedure continues until the predefined number of iterations is reached.

3.5 An Example and Testing Result of the Proposed Maximum Neural Networks

The proposed modeling, neural network and the optimization algorithm have been implemented on 800 MHz personal computers using MATLAB® software. One of the results of computer implementation is shown in this section.

An example testing data, as shown in Table 3.1, consisting of 10 cores is used for demonstrating of the developed techniques. These test data of Table 3.1 are used to calculate the testing time for each core as shown in Equations (3.4), (3.5) and (3.6). Table 3.2 shows the optimal width allocation and the optimal core assignment to each TAM for the example SOC system S_4 with three TAMs. In Table 3.2, the test bus assignment vector shows the core assignment to each TAM. For example, when the total width W is equal to 44 (the seventh row of Table 3.2), the optimal width distribution (w_1, w_2, w_3) is equal to (6, 13, 25), and the test bus assignment vector of the cores is [1, 1, 3, 3, 2, 3, 1, 3, 1, 1].

Figure 3.6 shows the Test Access Mechanism (TAM) design and test schedule for this SOC system when the total TAM width W is equal to 44 (the seventh row of Table 3.2). From the assignment test vector [1, 1, 3, 3, 2, 3, 1, 3, 1, 1] of Table 3.2 and TAM design of Figure 3.6, the cores at columns 1, 2, 7, 9 and 10 (i.e., C6288, C7552, S39532, S15850, and S38417, respectively) are assigned to TAM_1 with width of 6. The core at column 5 (i.e., S38584) is assigned to TAM_2 with width of 13, and other cores are assigned to TAM_3 with width of 25. The total testing time of the system is 1656820 cycles. TAM_1 and TAM_2 have idle times of 1345 and 2800, respectively, as shown in Figure 3.6(b). All the computation time for this example case is less than 60 seconds, as shown in Table 3.2.

More details of practical examples and analytical results of the proposed NN will be presented in Chapter 5.

3.6 Summary

In this chapter, we have proposed a maximum neural network to minimize the total SOC testing time by finding the optimal allocation of cores to each TAM line and the optimal selection of the TAM width. The techniques presented in this chapter can be used for the optimization of the System-on-a-Chip test system design that is critical for the semiconductor and electronics industry. Computer implementation and results of practical examples will be presented in Chapter 5.

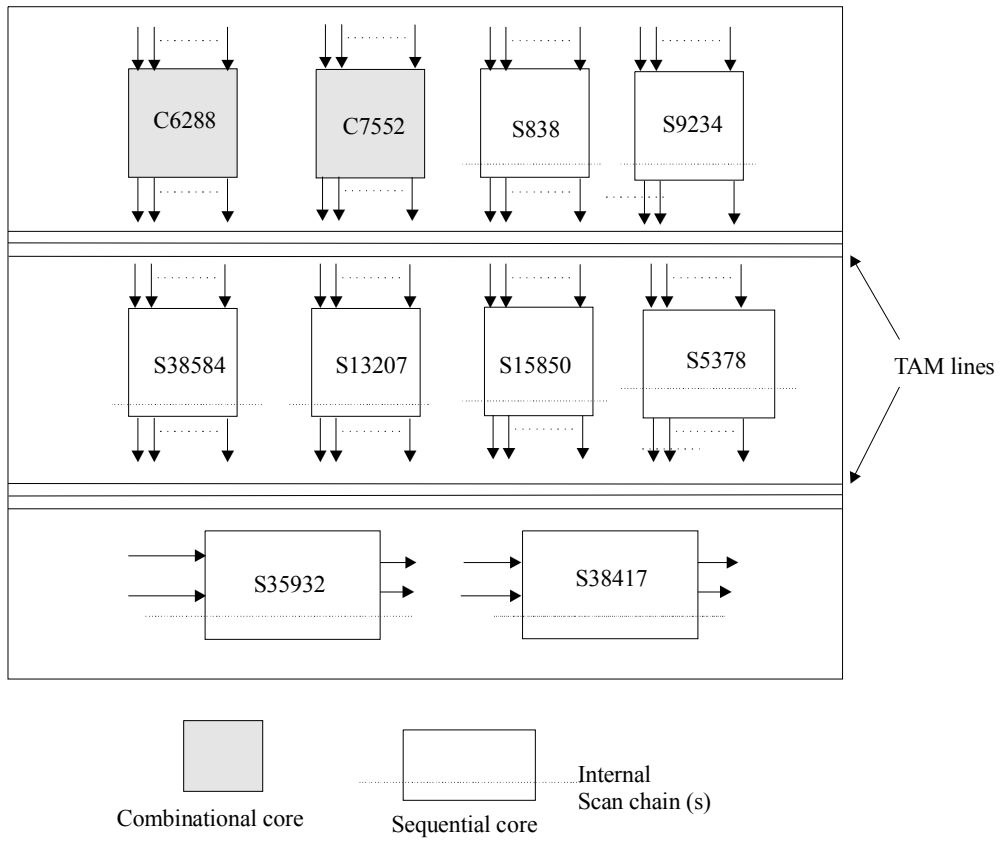


Figure 3.1 An example of core-based SOC system with two TAM lines [Chakrabarty 00a]

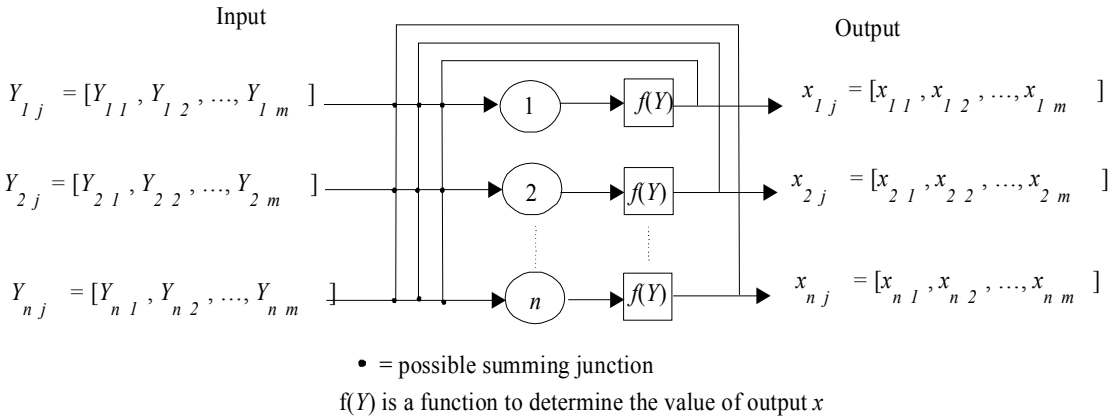


Figure 3.2 An example of general maximum neural network

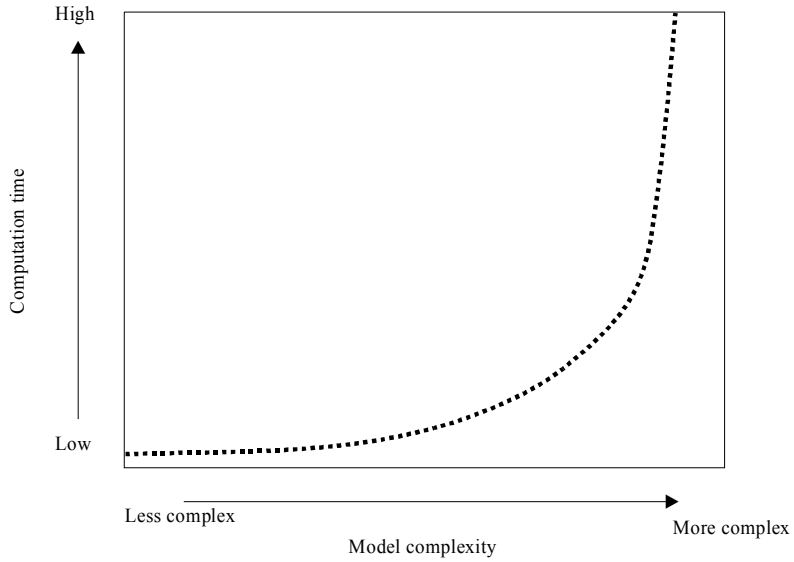


Figure 3.3 The relation of the computation time and the SOC model complexity (Using the traditional method to solve NP-complete problems)

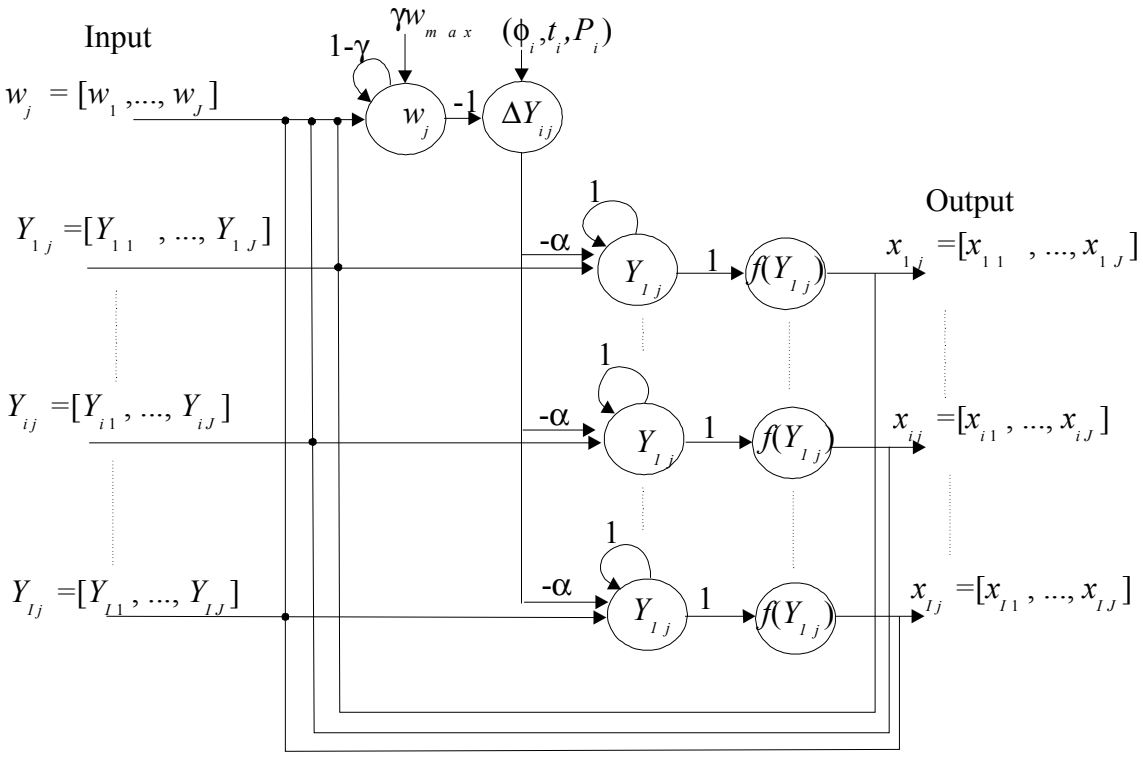


Figure 3.4 The proposed neural network architecture

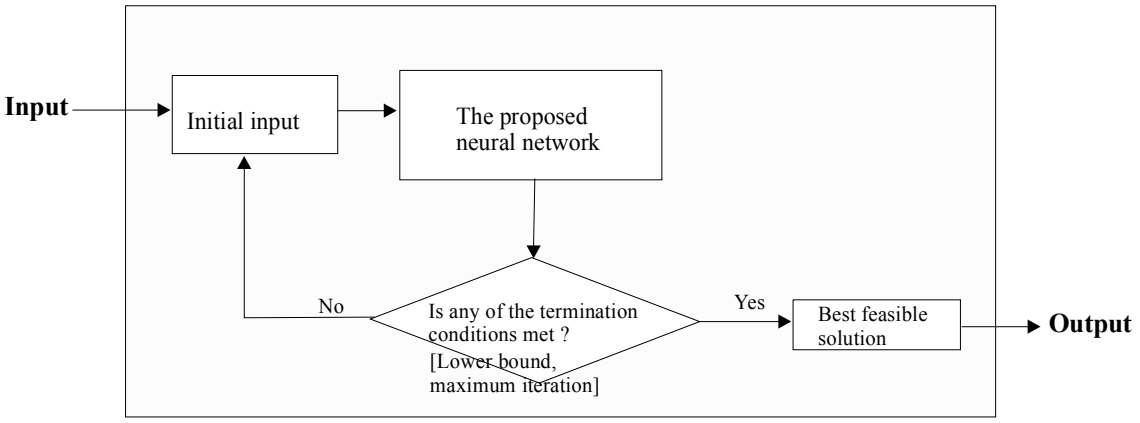


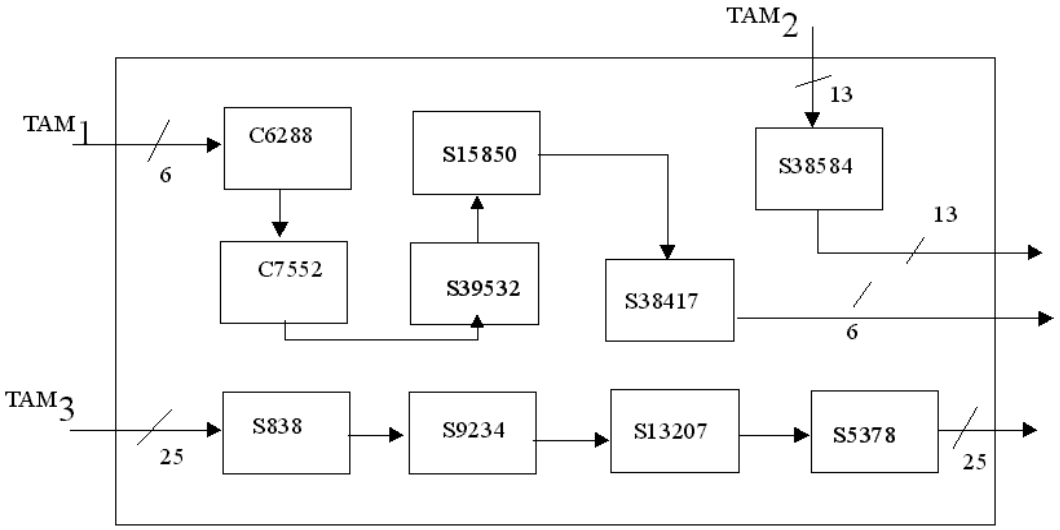
Figure 3.5 The proposed searching and optimizing algorithm for SOC test automation

Table 3.1 Test data of the core for the SOC system S_4

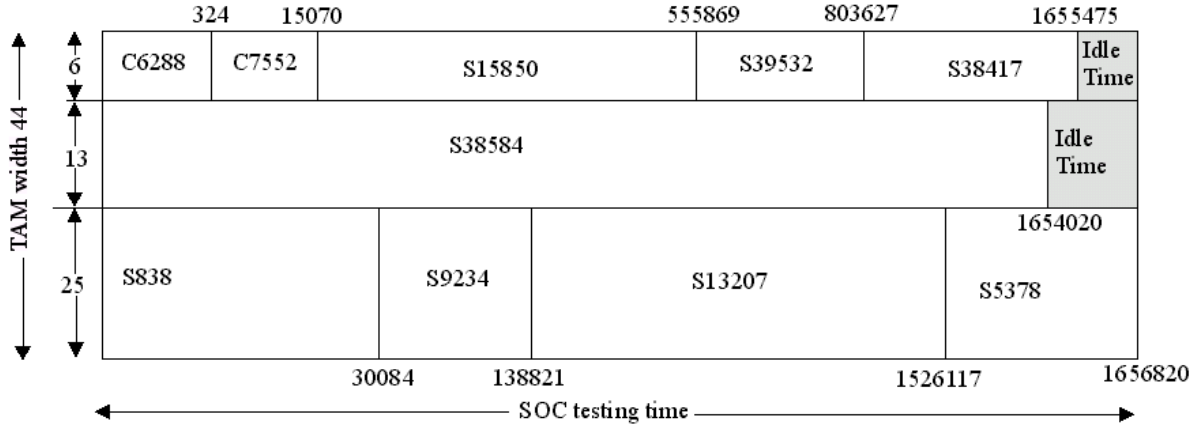
Core	i	Number of test inputs n_i	Number of test outputs m_i	$\phi_i = \max\{n_i, m_i\}$	Number of test patterns p_i	Number of test cycles t_i
C6288	1	32	32	32	12	12
C7552	2	207	108	207	73	73
S838	3	36	3	36	75	2507
S9234	4	40	43	43	105	5723
S38584	5	70	336	336	110	5105
S13207	6	78	168	168	234	9634
S15850	7	93	166	166	95	3359
S5378	8	39	53	53	97	4507
S35932	9	67	352	352	12	714
S38417	10	60	138	138	68	3656

Table 3.2 The results from using the proposed maximum neural network applied to the example SOC system S_4 with three TAMs

Total test width W	Testing Time (on $w1$, on $w2$, on $w3$)	Testing time (cycles) (Best Solution)	Bandwidth distribution (Best solution) (w_1, w_2, w_3)	Test bus assignment vector	Execution time (min)
16	(1812004, 1809212, 1805916)	1812004	(8, 7, 1)	(3, 1, 3, 2, 3, 1, 2, 2, 1, 2)	1
20	(1769686, 1730391, 1765784)	1769686	(1, 1, 18)	(1, 2, 1, 3, 2, 3, 1, 3, 1, 1)	0.58
24	(1724065, 1715280, 1694161)	1724065	(22, 1, 1)	(1, 1, 2, 3, 2, 3, 1, 1, 1, 1)	0.50
32	(1715067, 1710175, 1632988)	1715067	(23, 2, 7)	(3, 1, 1, 1, 2, 1, 3, 1, 3, 3)	0.45
36	(1656211, 1684650, 1670179)	1684650	(4, 7, 25)	(1, 3, 3, 3, 2, 3, 1, 3, 1, 1)	0.43
40	(1694281, 1689755, 1580312)	1694281	(29, 6, 5)	(3, 1, 1, 1, 2, 3, 1, 1, 1, 1)	0.40
44	(1655475, 1654020, 1656820)	1656820	(6, 13, 25)	(1, 1, 3, 3, 2, 3, 1, 3, 1, 1)	0.40
48	(1632676, 1664230, 1579976)	1664230	(32, 11, 5)	(1, 1, 1, 1, 2, 3, 1, 1, 1, 1)	0.40
52	(1624947, 1609250, 1586130)	1624947	(8 ,2, 42)	(2, 3, 3, 3, 3, 2, 1, 3, 1, 1)	0.38
56	(1633600, 1616141, 1589767)	1633600	(17, 11, 28)	(3, 2, 3, 3, 1, 3, 2, 3, 2, 2)	0.40



(a) TAM design



(b) Test schedule for the system S_4

Figure 3.6 TAM design and test schedule for the system S_4 with three TAMs and total TAM width of 44

CHAPTER 4

OPTIMIZING SYSTEM-ON-A-CHIP (SOC) TEST AUTOMATION WITH CORE WRAPPER DESIGN BY MAXIMUM NEURAL NETWORKS

This chapter presents a revised *maximum neural network* (MNN) approach to minimize the testing time and optimize test resource allocation for *System-on-a-Chip* (SOC) with embedded core wrapper design. In this chapter, the embedded core wrapper scan chain is considered in the SOC test optimization problem. The objective is to solve the optimal testing time for SOC testing automation within a reasonable computation time.

4.1 Introduction

System-on-a-Chip (SOC) is designed by embedding large pre-designed and pre-verified modules, called cores, onto one single circuit [Gupta 97]. A core is a description of a module and is not yet manufactured [Zorian 99]. Different core providers provide different cores along with their corresponding test sets. Cores, unlike components on a circuit board, need to be tested after being embedded. Testing is a critical issue in the embedding process because the application in which the core is integrated may strongly influence the test strategy in terms of fault-coverage, power consumption and silicon area [Benso 00]. Since the cores are designed by separate design teams (core providers) with a high degree of specialization in their areas of functional expertise and are optimized for different criteria, design for test (DFT) integration can create a major challenge [Gallagher 01, Irion 01].

Figure 4.1 shows an example of embedded core test infrastructure that consists of three elements: test pattern source and sink, test access mechanism (TAM), and core test wrapper [Zorian 99]. In the test pattern source and sink, the source generates the test stimuli for the embedded core, and the sink compares the response(s) to the expected response, as shown in Figure 4.1. Source and sink can either be implemented off chip (Automatic Test

Equipment, ATE) or on chip (Built-in Self Test, BIST), or in a combination of both [Marinissen 00]. A TAM transports test stimuli from a test pattern source to cores under test, and it also transports test responses from the cores under test to a test pattern sink, as shown in Figure 4.1. The core test *wrapper* is the interface between the core and the SOC system in which the core is embedded, as shown in Figure 4.1. By finding the efficient test wrappers and the TAMs, the SOC testing time can be minimized [Iyengar 01a, Koranne 02].

When considering core wrapper design along with SOC testing automation problem, the testing time of the system can be minimized [Iyengar 01a, Korranne 02, Marinissen 00]. Iyengar et al. (2001) proposed a wrapper/TAM (test access mechanism) co-optimization method to find an optimal assignment of cores to each of the TAMs to minimize the overall testing time [Iyengar 01a]. In their method, ILP and the co-optimization algorithm were used to find the optimal results. However, when the size of the system gets larger (more than two TAMs are used, or the TAM width becomes wider), the computation time gets extremely long. In the co-optimization method [Iyengar 01a], the power constraint is not considered. Korranne (2002) improved the SOC test schedule by using reconfigurable core wrappers [Korranne 02]. The set of reconfigurable scan chains were calculated using graph theory techniques to solve the embedded core-based System-on-a-Chip test scheduling problem (ECTSP). The testing time of a SOC system obtained by their method (ECTSPSol) is improved compared to the ILP formulations. Power consumption constraint was not considered in their study.

The objective function of a SOC testing problem is to minimize the overall testing time subject to various constraints. The objective function considers the optimal allocation of cores to each TAM, the power consumption, and the selection of TAM width for minimizing the SOC testing time. The SOC testing design with resource allocation problems are NP-complete [Chakrabarty 00b]. When a SOC system is small, traditional methods such as integer linear programming (ILP) or a heuristic approach may be able to reach an optimal solution. However, as a system becomes more complex, traditional methods may not be as efficient because the optimal solution is difficult to locate and the computation time gets

much longer [Kloypayan 02a, Kloypayan 02c]. When the SOC testing designers try to increase the number of cores, the TAM lines or the total TAM width, the traditional methods may not be effective in finding the optimal solution within a reasonable computation time and they sometimes cannot find the optimal solution at all.

In this chapter, a core wrapper design and the SOC resource allocation problems are studied to find the optimal SOC testing time. We adopt the concept of wrapper design from Marinissen et al. [Marinissen 00] for SOC design testing. A *maximum neural network* (MNN) is proposed to allocate cores to the TAMs to minimize the overall testing time. The proposed MNN is an unsupervised competitive neural network and is used to solve the NP-complete problem in this chapter. The remainder of this chapter is organized as follows. Section 4.2 presents a core wrapper design by the Partitioning of TAM Chain Items (PTI) method and Largest Processing Time (LPT) method. Section 4.3 constructs the model of the SOC test optimization problems with core wrap design. Section 4.4 presents a concept of the proposed MNN, and the procedure of constructing the MNN to solve the SOC testing problems. Concluding remarks are provided in Section 4.5.

4.2 Core Wrapper Design by Partitioning of TAM Chain Items (PTI) Method and Largest Processing Time (LPT) Method

In this chapter, we assume that a core user has already determined the TAM design after careful consideration of system-level I/O, area and power issues. Other than TAM design, core wrapper design should be considered to achieve more efficiency in testing the SOC. As shown in Figure 4.1, a wrapper is a thin shell around a core that connects TAMs to a core. Figure 4.2 shows an example of a core with wrapper design [Marinissen 00]. A wrapper scan chain design can provide for width adjustment of a mismatch between core input/output width and TAM width, as shown in Figure 4.2. We adopt the concept of wrapper design from Marinissen et al. [Marinissen 00], which considers only core-internal test. One of the problems in considering a core-internal test is partitioning the set of wrapper scan chain elements to several wrapper scan chains, which are equal to the number of TAM

lines. Let T be a testing time for a core; then the SOC testing design with core wrapper design problem can be formulated as follows [Marinissen 00]:

$$T = \{1 + \max(s_i, s_o)\} \cdot p + \min(s_i, s_o) \quad (4.1)$$

where s_i is the scan-in for a core;

s_o is the scan-out for a core; and

p is the number of test patterns.

As shown in Equation (4.1), the testing time for each core depends on the length of a wrapper scan chain, i.e., $max(s_i, s_o)$. To reduce the SOC testing time, it is important to balance the length of wrapper scan chains as much as possible [Iyengar 01a]. To balance the wrapper scan chain, several methods can be used [Koranne 02, Marinissen 00]. In this chapter, the *Partitioning of TAM Chain Items* (*PTI*) method is used to balance the wrapper scan chain design [Marinissen 00]. The PTI method is shown as follows (also in Figure 4.3):

PTI_Step_I. Assign the core-internal scan chains into TAM chains such that the maximum sum of scan lengths assigned to a TAM chain is minimized.

PTI_Step_II. Assign the wrapper input cells into TAM chains on top of TAM partition, such that the maximum scan-in time of all TAM chains is minimized. The wrapper input cells have length 1.

PTI_Step_III. Assign the wrapper output cells into TAM chains on top of TAM partition, such that the maximum scan-out time of all TAM chains is minimized. The wrapper output cells have length 1.

To execute PTI_Step_I in the PTI method, the *Largest Processing Time* (*LPT*) algorithm is used to assign the scan lengths [Marinissen 00]. Details of the LPT algorithm are discussed as follows. Let $S = \{S_1, S_2, \ldots S_y\}$ be a set of the core-internal scan chains, where scan chain S_i has a length $l(S_i)$. The Largest Processing Time (LPT) algorithm is detailed as follows (also in Figure 4.3):

LPT_Step_A. Sort S in descending order so that $l[S]_1 > l[S]_2 > \ldots > l[S]_y$.

LPT_Step_B. For i = 1 to *number of TAM partition*:

Put the maximum length of scan-in(scan-out) to the TAM partition *i*,
Delete the maximum length of scan-in(scan-out) from the scan set *S*.

LPT_Step_C. For *i* = *number of TAM partition* +1 to *number of scan chain*:
Put the maximum length of scan-in(scan-out) from the set *S* to the TAM partition that has minimum length,
Delete the maximum length of scan-in (scan-out) from scan set *S*.

After designing the wrapper scan-in and scan-out, the wrapper input cells and wrapper output cells are assigned to partition TAM such that the maximum length of all TAM chains is minimized [Marinissen 00].

Figure 4.3(a) shows a wrapper design example for an embedded core that has two functional inputs, one functional output and four core-internal scan chains with the length of 9, 6, 3, and 3 flip-flops (FF), respectively. In the example shown in Figure 4.3(a), we would like to design a wrapper scan chain of this core for a two-bit width TAM. Figure 4.3(b) shows the procedure of PTI and LPT methods. Using the LPT algorithm to solve PTI_Step_I, first, all the lengths of scan chains are ordered in descending order, which is 9, 6, 3, and 3, shown as LPT_Step_A in Figure 4.3(b). In LPT_Step_B, the longest scan chain SC1 of length 9 is assigned to the first partition, PT_1, of TAM and the next longest scan chain SC2 of length 6 is assigned to another TAM partition PT_2. In the first iteration of LPT_Step_C, the next scan chain SC3 of length 3 is assigned to the minimum length of TAM partition (i.e., PT_2) with a length of 6 bits. In the second iteration of LPT_Step_C, the last scan chain SC4 of 3 is assigned to PT_1 of length 9. After using the LPT algorithm, the length of wrapper scan chains PT_1 and PT_2 are 12 and 9 bits, respectively, as shown in Figure 4.3(b). In PTI_Step_II, the functional inputs (I) are assigned to each TAM partition by assigning first to the partition that has the minimum length of wrapper scan chain (i.e., PT_2). In the PTI_Step_III, the function output (O) is assigned to the minimum length (i.e., PT_2). After finishing all three steps of the PTI method, the maximum length of the wrapper scan chains design for this example core is 12 bits, as shown in Figure 4.3(c).

4.3 Modeling the SOC Test Optimization Problems with Core Wrapper Design

After the core wrapper scan chains have been designed, the testing time for each core and its TAM width is calculated using Equation (4.1). We then proceed to optimize the total testing time and the resource allocation. In the previous chapter and in our previous work presented in [Kloypayan 02c, Kloypayan 02d], a formulation modeling has been proposed for optimizing the SOC testing with SOC cores allocation and TAMs width selection. Let a SOC system consist of I cores and J TAM lines, i.e., ($1 \le i \le I$) and ($1 \le j \le J$). The width of each TAM line TAM_j is represented by w_j, which has to be less than or equal to the maximum width w_{max}. The summation of the width w_j for all TAMs should be equal to the total TAM width W. The power dissipation for each core i is represented by P_i. We assume that the power dissipation per cycle for core i is equal to power dissipation P_i divided by testing time of that core on TAM_j, i.e., $\frac{P_i}{T_i(w_j)}$. In this chapter, a power dissipation P_i and the maximum power Ω of the SOC system are given and considered in the optimization. For simplicity, we assume that the power allocated to each TAM is equal, which is $\frac{\Omega}{J}$. When a core i is tested on TAM_j, the power consumption should not exceed the allocated power of TAM_j, i.e., $\left(\frac{P_i}{T_i(w_j)} \le \frac{\Omega}{J}\right)$. To minimize the SOC testing time with resource allocation, the SOC testing model can be formulated as follows:

$$\text{Objective:} \quad \text{Minimize } \{\max(\sum_{i=1}^{I} T_i(w_j) \cdot x_{ij})\}, \ 1 \le j \le J \tag{4.2}$$

$$\text{Subject to} \quad \sum_{j=1}^{J} x_{ij} = 1, \ 1 \le i \le I \qquad \text{(Resource allocation)} \tag{4.3}$$

$$\sum_{j=1}^{J} w_j = W, \qquad \text{(TAM width distribution)} \tag{4.4}$$

$$w_j \le w_{max}, \tag{4.5}$$

$$\frac{x_i P_i}{T_i(w_j)} \le \frac{\Omega}{J}, \qquad \text{(Power constraint)} \qquad (4.6)$$

$$x_{ij} = 0, \text{ or } 1 \qquad (4.7)$$

where $T_i(w_j)$ is the testing time for core i on TAM_j , as shown earlier in Equation (4.1), $1 \le i \le I$, $1 \le j \le J$;

w_j is the width of TAM_j ;

W is the total width of TAMs;

w_{max} is the maximum width of TAMs;

Ω is the maximum power rating in the SOC system;

P_i is the power consumption of core i;and

x_{ij} is the 0-1 variable defined as $x_{ij} = \begin{cases} 1, & \textit{if Core i is assigned to TAM j} \\ 0, & \textit{otherwise} \end{cases}$.

The SOC testing with resource allocation problem is a NP-complete problem [Kloypayan 02c]. Solving a NP-complete problem with traditional methods such as the Integer Linear Programming (ILP) becomes inefficient when the SOC system becomes more complex. In this chapter, a *maximum neural network* (MNN) is proposed to solve the NP-complete SOC testing problems. Details are presented in the next section.

4.4 Constructing the Maximum Neural Network (MNN) to Solve the SOC Testing Problems

A *neural network* (NN) is a system constructed to imitate the function of a brain [Kartalopoulos 96]. As shown in Figure 4.4, a NN consists of individual neural units (nodes) that link together with the associated weight between each node. Neural networks can provide optimal solutions to difficult optimization problems, which the traditional heuristic methods cannot generally accomplish. Solving the optimization problems require the minimization of the cost functions subject to a set of constraints. The neural network can produce good solutions by minimizing the cost function, also known as *energy functions* [Smeda 99]. In addition, by designing the right neural networks, the computation time for

achieving the optimal solution is usually less than that of traditional optimization techniques [Sellers 96].

In this chapter, after solving the PTI problem (i.e., cores wrapper design for each TAM partition has been balanced and kept in database), an unsupervised *maximum neural network* (MNN) is also proposed to optimize the resource allocation problems for SOC testing design, as shown in Figure 4.4. Neural networks are able to solve NP-complete problems in polynomial time [Kartalopoulos 96, Takefuji 92]. The operation of the proposed MNN is based on the group update. Details of a general model of the maximum neural network have been presented in Chapter 2.

Based on a general model of a maximum neural network in Chapter 3, Figure 4.4 shows the architecture of the proposed MNN. There are $J + 1$ input units: $[Y_{ij}, j=1,\ldots,J]$ and $[w_j, j = 1,..,J]$, and I output units: $[x_{ij}]$ for $i= 1,\ldots,I$. From the previous chapter in Equations (3.1) to (3.3), Y_{ij} is an energy function, which is the testing cost of the SOC testing system design. In Figure 4.4, an output x_{ij} defines that core i is assigned to TAM_j. By using Equations (4.2)-(4.7), the proposed MNN can be constructed so that each core i can be assigned only to one TAM line, TAM_j. In the MNN, only one output is selected (i.e., in Figure 4.4, there is only one output x_{ij}, j=1…J, has a non-zero value and all others are zero). In the proposed SOC test model, an input Y_{ij} and an output x_{ij} are calculated as follows (also see Figure 4.4):

$$x_{ij} = f(Y_{ij}) = \begin{cases} 1 & if\ Y_{ij} = \max\{Y_{ia}\}\ \ (a = 1,...,J) \\ 0 & otherwise \end{cases}, \tag{4.8}$$

$$Y_{ij}(u+1) = Y_{ij}(u) - \Delta Y_{ij}, \tag{4.9}$$

$$\Delta Y_{ij} = \alpha(u) \cdot T_i(w_j) + \gamma(u) \cdot \left(\frac{x_{ij} P_i}{T_i(w_j)} - \frac{\Omega}{J} \right), \tag{4.10}$$

$$w_j(u+1) = w_j(u) - \beta(u) \cdot (w_j(u) - w_{\max}),\ j = 1,..,J\text{-}1, \tag{4.11}$$

$$w_J(u+1) = W - \sum_{j=1}^{J-1} w_j(u+1), \tag{4.12}$$

where $\alpha(u)$, $\gamma(u)$ and $\beta(u)$ are learning rates;

u is number of iterations, $u = 1,\ldots,U$;

$w_j(u)$ is the width of TAM_j at iteration u, $1 \le j \le J$;

w_{max} is the maximum width of TAMs;

W is the total width of TAMs;

Ω is the maximum power rating in the SOC system;

P_i is the power consumption of core i; and

$T_i(w_j)$ is the testing time for core i on TAM_j determined by Equation (4.1) after considering wrapper design, and $1 \le i \le I$, $1 \le j \le J$.

In searching for the optimal solution, the learning rates α, β and γ may be decreased at each iteration or they may be constant throughout the learning process.

As shown in Figure 4.4, the proposed MNN begins the first iteration by generating a set of random numbers assigned to the energy function Y_{ij} and the width w_j for each TAM_j. The output x_{ij} can be calculated by using the function $f(Y_{ij})$ defined in Equation (4.8). As shown in Figure 4.4, in searching for the optimal solution, Y_{ij} and w_j for the next iteration are updated using Equations (4.9) to (4.12). The searching procedure continues until either the optimal solution has been reached or the termination conditions are met.

Figure 4.5 shows the proposed MNN searching algorithm for solving the SOC test problem with resource allocation. The complete MNN searching algorithm of the proposed maximum neural network is shown as follows (also in Figure 4.5):

MNN_Step_1. For each core in a SOC system, minimize the maximum length of wrapper scan chain for the PTI problem by using the LPT algorithm, as discussed in the previous section.

MNN_Step_2. Calculate the SOC testing time cycle $T_i(w_j)$ of each core i for each TAM j with w bits wide, using Equation (4.1), and save the result in a database.

MNN_Step_3. Initialize the proposed neural network MNN, set $u = 0$.

MNN_Step_4. Assign each w_j value by using a uniform random number from 1 to W.

MNN_Step_5. Assign each Y_{ij} value by using a uniform random number.

MNN_Step_6. Calculate x_{ij} by Equation (4.8).

MNN_Step_7. Obtain the testing time $T_i(w_j)$ from the database.

Calculate the testing time for the system.

IF the new testing time for the system is shorter than the old testing time and the power constraint is not violated,

THEN keep the new testing time as the minimum testing time for the SOC system.

MNN_Step_8. Update Y_{ij} and w_j for the next iteration (u+1).

Increment u by 1.

IF the state of the system reaches the equilibrium state or the termination conditions are met,

THEN stop this procedure.

ELSE, repeat to MNN_Step_6.

The termination condition used in this chapter includes the following two situations: (i) the iteration number has reached a predefined number, or (ii) the MNN system has reached a stable point. The procedure stops once either of these conditions is met, as shown in Figure 4.5.

The proposed MNN searching and optimization algorithm is used to find the optimal SOC testing solution. As shown in Figure 4.5, in the beginning the set of scan chains for each core in a SOC system is balanced (PTI problem). Then, the testing time of each core for each TAM width partition is calculated. Next, random numbers for the initial input to the MNN are generated, including a set of the energy functions Y_{ij} and the width w_j of each TAM_j. The MNN finds the testing time for each core i for each TAM width w_j from the database. The output x_{ij} from the MNN is for the SOC test resource allocation. If the MNN reaches an optimal solution, the procedure stops; otherwise, the procedure continues until the predefined number of iterations is reached, as shown in Figure 4.5.

4.5 An Example and Testing Results of the Proposed Modeling and MNN

The proposed modeling, neural network and the optimization algorithm have been implemented on 800 MHz personal computers using MATLAB® software. In this section, one of the results from computer implementation is demonstrated.

In this example, the proposed modeling and MNN is implemented using the test data of SOC chip d695 from the literature in [ITC 02]. Table 4.1 shows the detail of test data for each core of SOC chip d695, which consists of ten cores (including two combinational cores and eight sequential benchmark circuits). Figure 4.6 shows the Test Access Mechanism (TAM) design found by the MNN for this example d695. As shown in Figure 4.6, the cores 1, 2, 7, 8 and 9 are assigned to TAM_2 with width of 5. Cores 3, 4 and 10 are assigned to TAM_3 with width of 7, as shown in Figure 4.6. Table 4.2 shows the resultant best solutions of width allocation and core assignment to each TAM for the first example SOC d695 system with three TAMs ($j = 3$), where the power constraint is not considered. In Table 4.2, the test bus assignment vector shows the core assignment to each TAM. For example, when the total width W is equal to 28 (the fourth row of Table 4.2), the SOC testing time is 25,936 cycles, the optimal width distribution (w_1, w_2, w_3) for the TAM_1, TAM_2 and TAM_3 is (16,5,7) and the test bus assignment vector of the cores is [2,2,3,3,1,1,2,2,2,3], as shown in Table 4.2. In Table 4.2, the test bus assignment vector [2,2,3,3,**1**,**1**,2,2,2,3] shows that core 5 and core 6 (the underlined items in the vector) are assigned to TAM_1, also shown in Figure 4.6. All the MNN computation time of the MNN for this example case d695 is within 30 seconds, as shown in Table 4.2.

More details of the computer implementations and practical examples will be presented in Chapter 5.

4.6 Summary

In this chapter, a maximum neural network (MNN) has been proposed to minimize the total testing time for SOC test automation with core wrapper design. The MNN and the mathematical modeling have been presented to solve the resource allocation and test

scheduling problems for SOC test automation. After the embedded core wrapper scan chains are designed, the best solution of SOC testing time can be solved by the developed MNN algorithms in a polynomial time. Computer implementation and more results of practical examples will be presented in Chapter 5.

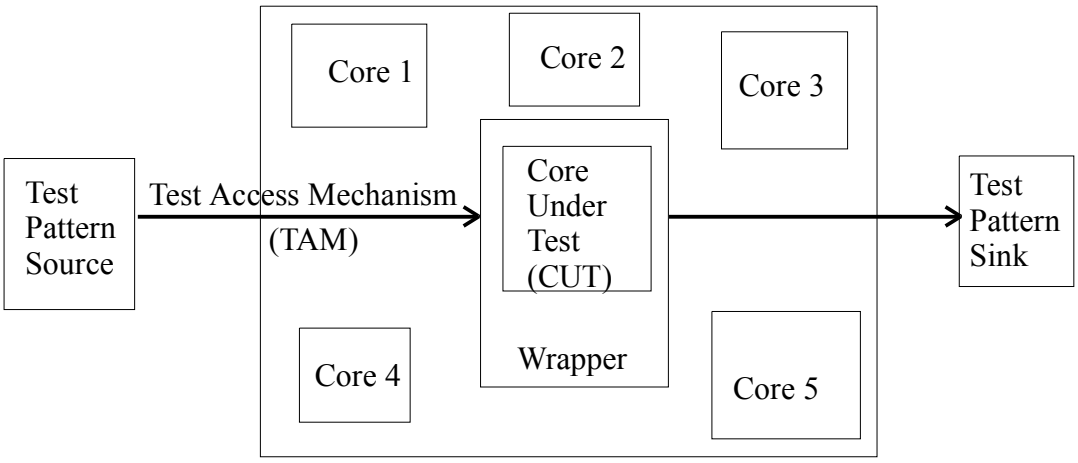


Figure 4.1 Embedded core test infrastructure for SOC testing

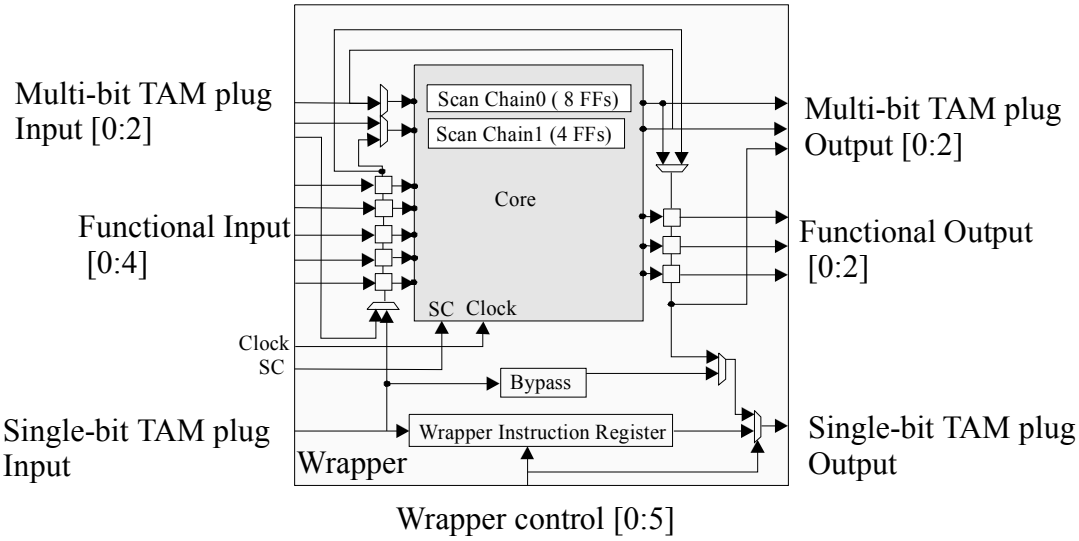
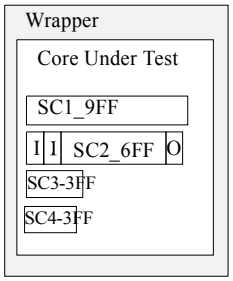
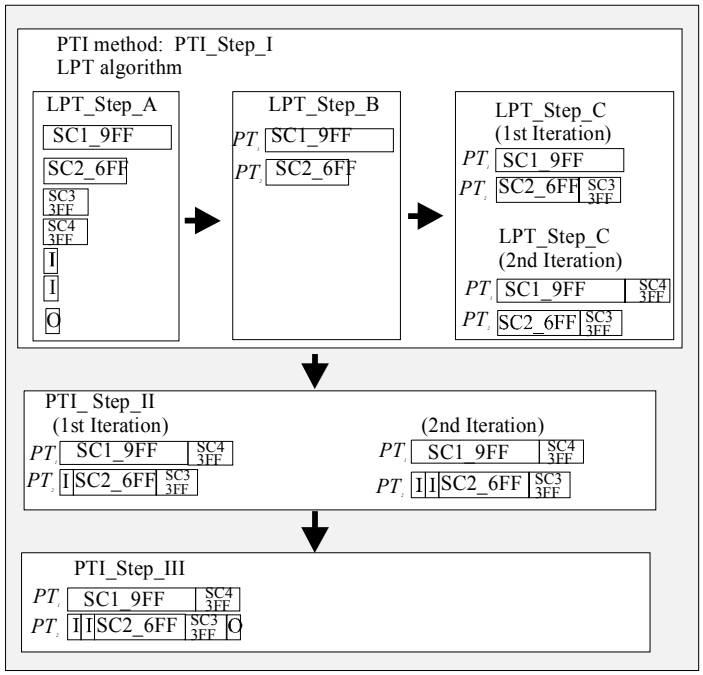


Figure 4.2 An example of a core with wrapper [Marinissen 00]

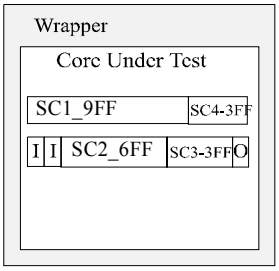


SC : scan chain
I : functional input
O : functional output
PT_i : k^{th} TAM partition

(a)



(b) The PTI and LPT procedures



2-bit width TAM
Longest wrapper scan chain = 12 bits

(c) The final result of core wrapper design

Figure 4.3 An example of wrapper scan chain design by the PTI and LPT algorithms

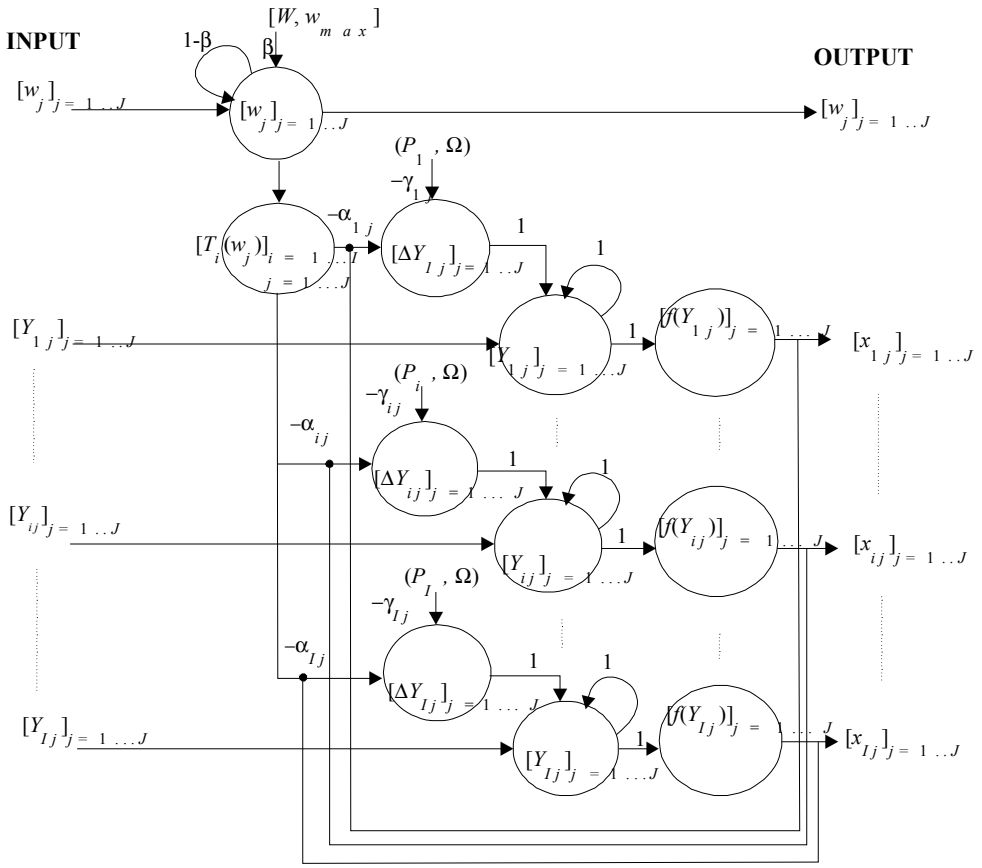


Figure 4.4 The proposed maximum neural network (MNN) architecture

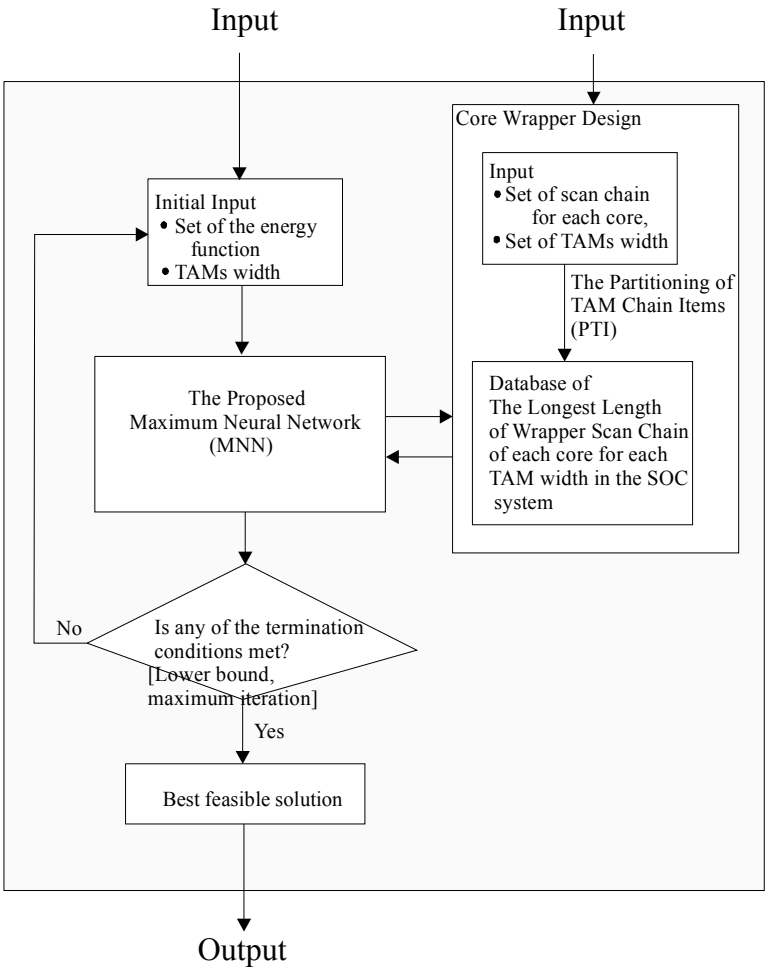
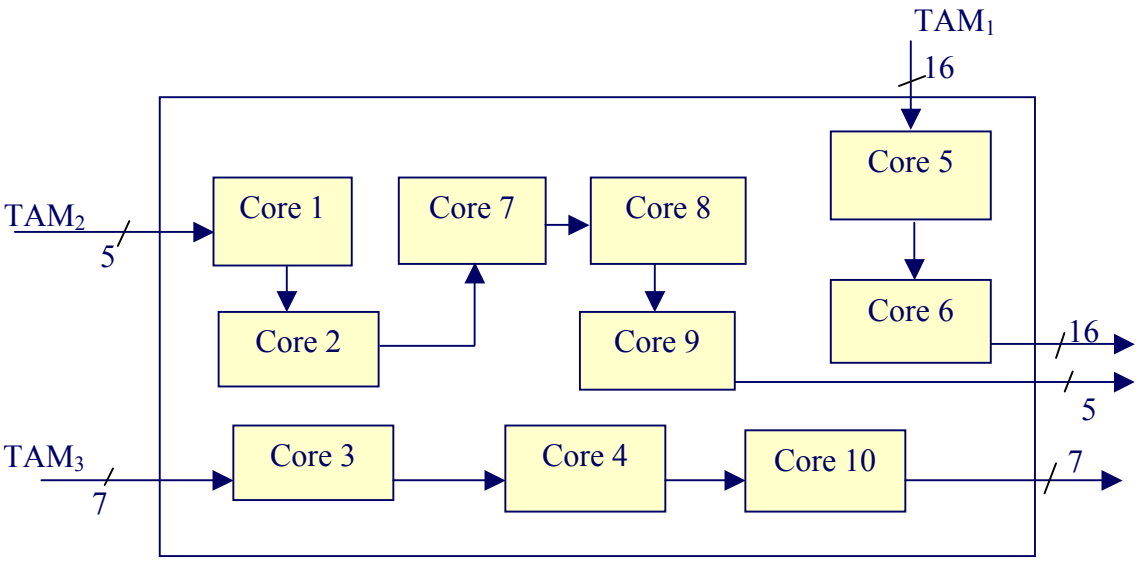


Figure 4.5 The proposed MNN searching and optimizing algorithm for SOC test design

Table 4.1 Test data for each core in SOC chip d695 [ITC 02]

Core No.	No. of Test Patterns	No. of Input	No. of Output	No. of Scan Chains	Scan Chain Length	
					Min	Max
1	12	32	32	-	-	-
2	73	207	108	-	-	-
3	75	34	1	1	32	32
4	105	36	39	4	52	54
5	110	38	304	32	44	45
6	234	62	152	16	39	41
7	95	77	150	16	33	34
8	97	35	49	4	44	46
9	12	35	320	32	54	54
10	68	28	106	32	51	51



SOC testing time (cycles)	25936
TAM width	28
TAM partition	(16, 5, 7)
TAM allocation	[2, 2, 3, 3, 1, 1, 2, 2, 2, 3]
Computation time (min)	0.48

Figure 4.6 Test Access Mechanism (TAM) design for the SOC system d695 with three TAMs and total TAM width *W* is equal to 28 bits

Table 4.2 The best solution of SOC testing time, width allocation and core assignment for the SOC system d695 with three TAMs.

TAM width (bits)	SOC Testing time (cycles)	Width Allocation (bits)	TAM Allocation	Computation time (min)
16	44029	(6,7,3)	[1,2,3,1,2,1,2,1,2,3]	0.52
20	35606	(16,1,3)	[3,2,2,3,1,1,3,2,1,1]	0.49
24	29699	(18,3,3)	[3,2,2,3,1,1,2,3,3,1]	0.48
28	25936	(16,5,7)	[2,2,3,3,1,1,2,2,2,3]	0.48
32	23578	(21,2,9)	[1,1,2,3,1,1,1,2,3,3]	0.48
36	20325	(20,4,12)	[3,3,2,2,1,1,3,2,3,3]	0.47
40	18674	(18,4,18)	[2,3,2,2,1,3,1,2,1,3]	0.49
44	17780	(20,20,4)	[1,2,2,3,2,1,2,3,2,1]	0.48
48	17143	(20,19,9)	[1,1,3,3,1,2,3,2,2,1]	0.51
52	16364	(17,16,19)	[3,3,2,2,1,3,3,1,3,2]	0.48
56	15747	(33,18,5)	[2,3,3,3,1,2,2,1,1,1]	0.48
60	13341	(36,7,17)	[1,2,2,2,1,3,1,2,1,1]	0.47
64	12867	(37,20,7)	[1,2,3,3,1,2,2, 3,1,1]	0.49

CHAPTER 5

COMPUTER IMPLEMENTATIONS AND RESULTS

This chapter presents the computer implementations and results of the proposed methods. The proposed methods have been implemented on 800 MHz personal computers using Matlab software.

5.1 Results for the SOC Test Automation with Resource Allocation Problem Using the Maximum NN

This section shows the implementation results of proposed SOC optimization modeling and the maximum neural network (MNN) presented earlier in Chapter 3. The examples demonstrated in this section use the same testing data of the SOC system S_4 from Table 3.1, mentioned earlier in Chapter 3.

Table 5.1 shows the optimal width allocation and the optimal core assignment to each TAM for the example SOC system S_4 with two TAMs. In Table 5.1, the test bus assignment vector shows the core assignment to each TAM. For example, when the total width W is equal to 20 (the second row of Table 5.1), the optimal width distribution (w_1, w_2) is equal to (2, 18) and the test bus assignment vector of the cores is [1, 1, 2, 2, 2, 1, 1, 2, 1, 2]. The assignment test vector [2, 1, 1, 1, 2, 1, 1, 1, 1, 2] from Table 5.1 means that the cores at columns 1, 2, 6, 7 and 9 are assigned to TAM_1 with width of 2, and other cores are assigned to TAM_1 with width of 18. All the computation time for this example case is less than 60 seconds, as shown in Table 5.1.

Figure 5.1 shows the Test Access Mechanism (TAM) design and test schedule for the system S_4 with two TAMs, and the total TAM width W is equal to 20. For the TAM design for this example SOC system, cores S838, S9234, s38584, S5378 and S38417 are assigned to a TAM_1 with width of 2, and the other cores are assigned to a TAM_2 with width of 18, as

shown in Figure 5.1(a) and Table 5.1 (the second row). Figure 5.1(b) shows the test schedule for the design of the example SOC test. The total testing time is 2429137 cycles on TAM_1 with width of 2. On TAM_2 with width of 18, the total testing time is 2429554 cycles, and it has idle time of 417 cycles. Compared to the traditional ILP methods, the computation times are much shorter (< 1 min for the example case).

Table 5.2 and Figure 5.2 show the comparison of SOC testing time for the SOC system S_4 with two TAMs using the three different methods: Integer Linear Programming [Chakrabarty 00a], Genetic Algorithm [Ebadi 01] and the proposed MNN. The SOC testing time found by the proposed method is close to the reported SOC testing time from ILP [Chakrabarty 00a] and GA [Ebadi 01].

Table 5.3 shows the results when the example SOC system S_4 has four TAM lines, respectively. In this example, when the TAM lines are increased from two lines to four lines, the testing time of the SOC system is decreased, as shown in Tables 5.1 and 5.3. Table 5.4 shows the optimal testing time for the example system S_4 when the number of TAM lines varies. In Table 5.4, a total width of 56 is used for the example SOC system. As shown in Table 5.4, when the number of TAM lines is increased, the total testing time of this system is decreased because an increase in the number of TAM lines means more than one core could be tested at the same time. The decreasing of total testing time is significant when one TAM line changes to three TAM lines. When the number of TAM lines changes from three to seven, the total testing times do not change much The increasing of the number of TAM lines does not always decrease the total testing time. With the same total TAM width, the width of each TAM line becomes smaller when increasing number of TAM lines. When the width of each TAM line is smaller than the core test width, the total testing time of the SOC system is increased.

Figure 5.3 shows the relationship among the total SOC testing time, the number of TAM lines and the total TAM width. When the total width is increased, the overall testing time becomes lower, which is consistent with Equation (3.4). When the width of the TAM line is increased, the testing time for each core becomes lower. As discussed earlier (in Table

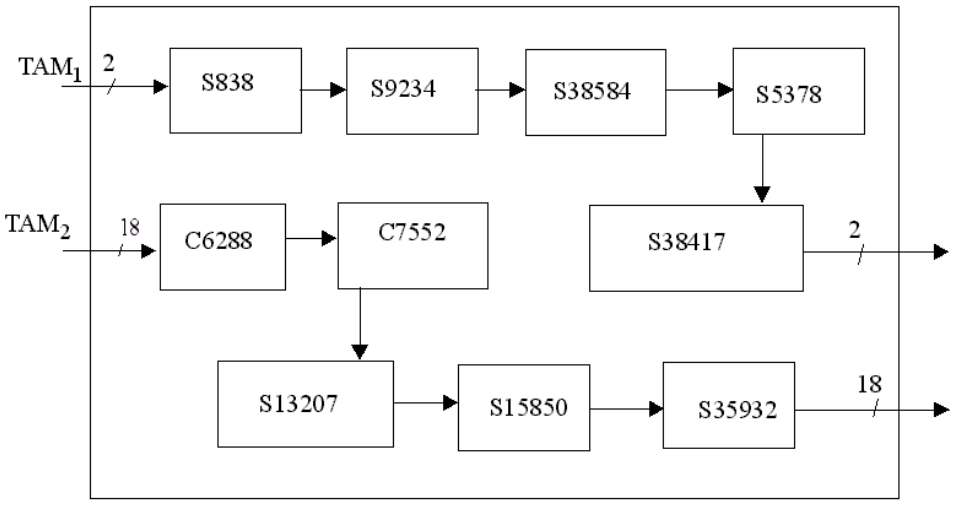
5.4), when the number of the TAM lines is increased, the overall testing time is decreased until it nearly reaches an optimal result. These results will help a SOC core user in TAM design.

In the previous results, the power constraint has not yet been considered. In Table 5.5, the power constraint is considered along with other constraints to minimize the testing times for the SOC system S_2 with two TAM lines. The maximum power of the system equals 100 mW. Similar to the results from Table 5.1, when the total TAM width is increased, the total testing time is decreased. Compared to the best testing time in Table 5.1, the best testing times in Table 6 are longer due to the power constraint. When the power constraint is considered, the computation times are also longer.

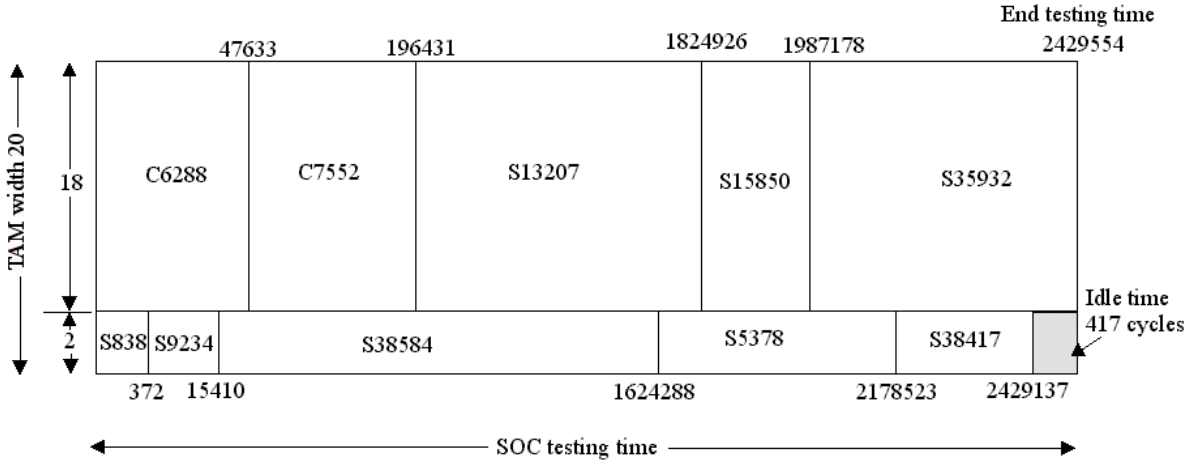
Table 5.6 shows the best testing times for the SOC system S_2 when the maximum power dissipations vary from 100 to 500. The SOC system has two TAM lines with a total TAM width of 28 bits. When the maximum power dissipation Ω is increased from 100 mW to 200 mW, the testing time of the SOC system is decreased. When Ω is increased to 500 mW, the testing time of the SOC system does not change significantly because the power constraints are more relaxed, as shown in Table 5.6.

Table 5.1 The results from the proposed maximum neural network applied to the example SOC system S_4 with two TAMs

Total test width W	Testing time (Best Solution from our method)	Optimal width distribution (w_1, w_2)	Test bus assignment vector	Execution time (min)
16	2456793	(3,13)	[1, 2, 2, 2, 1, 2, 2, 2, 1, 1]	0.70
20	2429554	(2,18)	[1, 1, 2, 2, 2,1, 1, 2, 1, 2]	0.65
24	2361278	(1,23)	[1, 1, 2, 2, 1, 2, 1, 2, 2, 2]	0.58
32	2222247	(1,31)	[1, 2, 2, 2, 1, 2, 2, 2, 2, 1]	0.60
36	2195730	(32,4)	[2, 1, 1, 1, 2, 1, 1, 1, 1, 2]	0.59
40	2169213	(7,33)	[1, 2, 2, 2, 1, 2, 2, 2, 2, 1]	0.57
44	2039183	(1,43)	[1, 1, 2, 2, 1, 2, 2, 2, 1, 2]	0.56
48	1975827	(46,2)	[1, 2, 1, 1, 2, 1, 1, 1, 2, 1]	0.56
52	1949151	(4,48)	[2, 2, 2, 2, 1, 2, 2, 2, 1, 2]	0.56
56	1931694	(7,49)	[2, 2, 2, 2, 1, 2, 2, 2, 1, 2]	0.55



(a) TAM design



(b) Test schedule for system S_4

Figure 5.1 TAM design and test schedule for the system S_4 with two TAMs and total TAM width of 20

Table 5.2 Comparison of the SOC testing time for different methods

Total test width *W*	Testing time (ILP method) [Chakrabarty 00a]	Testing time (Genetic Algorithm) [Ebadi 01]	Testing time (Best Solution from our method)	Testing time difference (compare with ILP) (%)	Execution time (min)
16	2423712	2478822	2456793	1%	0.70
20	2363126	2423284	2429554	3%	0.65
24	2278443	2361278	2361278	4%	0.58
32	2202286	2222247	2222247	1%	0.60
36	2174501	2195730	2195730	1%	0.59
40	2149720	2144192	2169213	1%	0.57
44	2123437	2039183	2039183	-4%	0.56
48	2099390	1966608	1975827	-6%	0.56
52	2086542	1949151	1949151	-7%	0.56
56	2069738	1931694	1931694	-7%	0.55

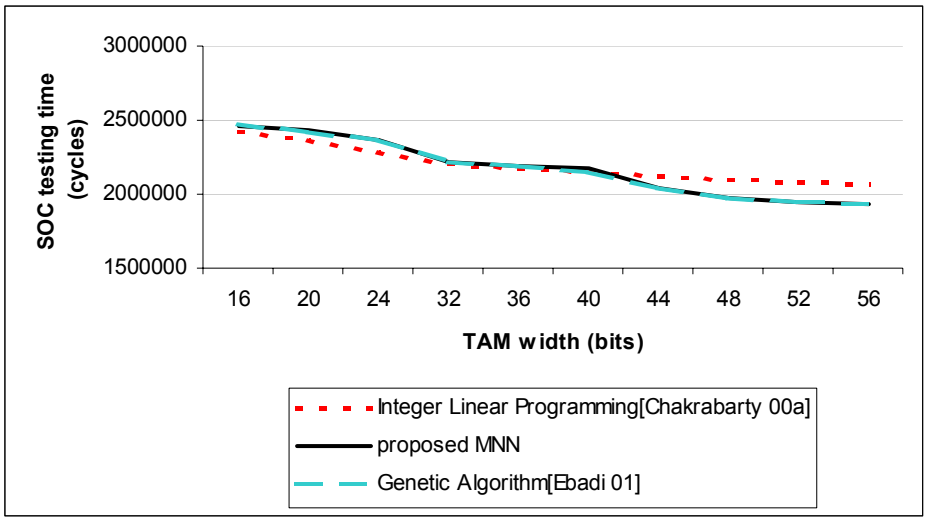


Figure 5.2 The comparison of SOC testing time for SOC system S_4 with two TAMs using different methods

Table 5.3 The results from using the proposed maximum neural network applied to the example SOC system S_4 with four TAMs

Total test width W	Testing Time (on $_W1$, on $_W2$, on $_W3$, on $_W4$)	Testing time (cycles) (Best Solution)	Width distribution (Best Solution) (w_1, w_2, w_3, w_4)	Test bus assignment vector	Execution time (min)
16	(1687233, 491352, 1684650, 1643549)	1687233	(4, 2, 7, 3)	(2, 1, 1, 2, 3, 1, 4, 4, 2, 4)	1.21
20	(1638705, 1608878, 1472908, 796849)	1638705	(16, 2, 1, 1)	(4, 3, 3, 3, 1, 2, 4, 4, 3,3)	0.87
24	(1618441, 1608878, 841651, 1427722)	1649143	(20, 2, 1, 1)	(1, 3, 3, 3, 1, 2, 4, 3, 3, 4)	0.87
32	(1592760, 1604850, 1100836,1145263)	1604850	(25, 4, 2, 1)	(2, 2, 4, 4, 1, 2, 4, 3, 4, 3)	0.62
36	(1541440, 1131196, 1116217,1602970)	1602970	(9, 3, 1, 23)	(2, 2, 2, 3, 4, 1, 2, 2, 2, 3)	0.63
40	(1585088, 1577505, 1045215,1167241)	1585088	(6, 28, 2, 4)	(2, 1, 4, 3, 2, 1, 3, 4, 3, 4)	0.60
44	(988198, 1575381, 1198314, 1572340)	1575381	(5, 7, 3, 29)	(3, 2, 3, 1, 4, 2, 1, 1, 3, 3)	0.58
48	(1539112, 1531806, 803683, 1375822)	1614209	(36, 10, 1, 1)	(4, 4, 1, 3, 1, 2, 3, 4, 4, 4)	0.65
52	(1194583, 1030993, 1526846, 1541710)	1570342	(2, 3, 12, 35)	(2, 3, 1, 1, 4, 3, 2, 2, 2, 1)	0.60
56	(1533627, 1550931, 1279951, 870128)	1550931	(39, 14, 2, 1)	(3, 1, 2 , 3, 1, 2, 3, 3, 3, 4)	0.53

Table 5.4 The best solution of testing time for the system S_4 when the number of TAM lines are varying

No. of TAM line	Testing time (cycles) (Best Solution)	Width distribution (w_j) (Best Solution)	Test bus assignment vector
1	3800947	(56)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
2	2157430	(1, 55)	(2, 1, 1, 2, 1, 2, 2, 2, 1, 2)
3	1633623	(32, 17, 7)	(3, 1, 3, 2, 3, 1, 2, 2, 2, 2)
4	1606119	(13, 3, 15, 25)	(1, 4, 1, 2, 4, 1, 2, 2, 3, 3)
5	1579976	(46, 5, 3, 1, 1)	(5, 4, 5, 4, 1, 2, 4, 3, 3, 5)
6	1589610	(20, 28, 4, 2, 1, 1)	(4, 4, 6, 5, 2, 3, 6, 1, 1, 5)
7	1608878	(12, 28, 9, 2, 2, 2, 1)	(1, 4, 5, 1, 2, 6, 7, 7, 4, 3)

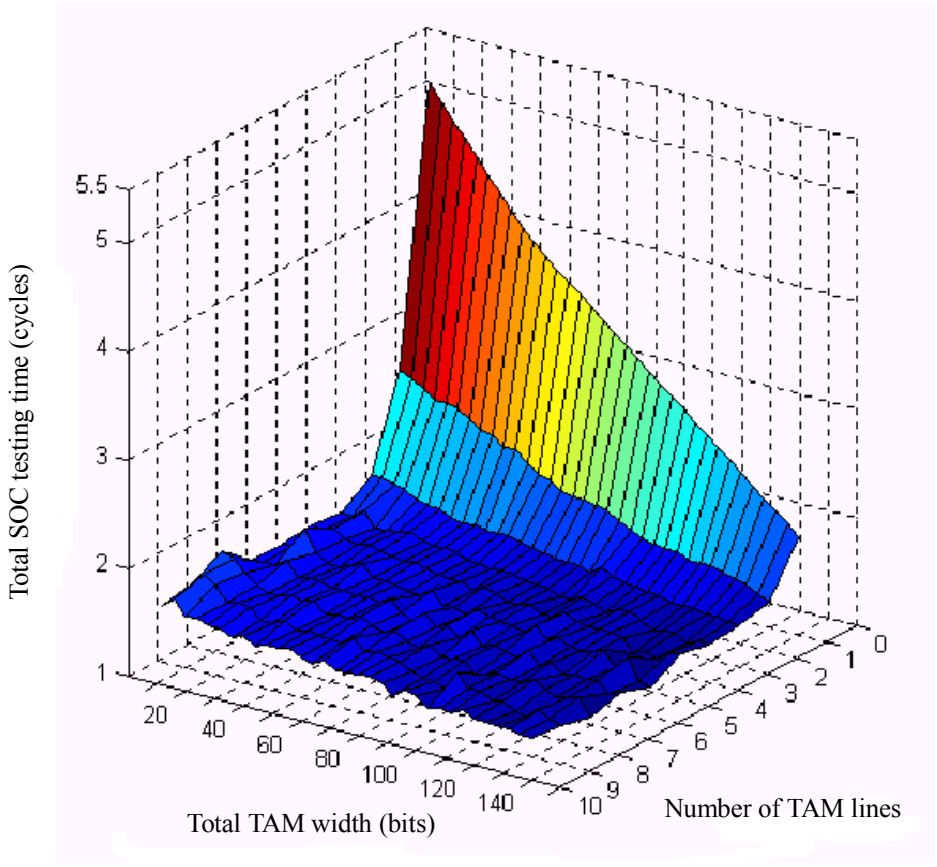


Figure 5.3 The relationship of the total SOC testing time and the number of TAM lines and the total TAM width

Table 5.5 The testing time for the system S_4 with the power constraint when the total TAM widths are varying (the maximum power is 100 mW)

Total test width W	Width distribution (w_1, w_2) (The best solution from our method)	Testing Time (cycles) (on $_W1$, on $_W2$)	testing time (The best solution from our method)	Test bus assignment vector	Execution time (min)
16	(1, 15)	(2600903, 2632430)	2632430	(1, 1, 2, 2, 1, 2, 2, 2, 2, 1)	2.0
20	(3, 17)	(2568246, 2593485)	2593485	(1, 2, 2, 2, 1, 2, 2, 2, 2, 1)	3.45
24	(19, 5)	(2564037, 2568050)	2568050	(2, 1, 2, 1, 2, 1, 2, 1, 2, 1)	4.20
28	(7, 21)	(2544656, 2516851)	2544656	(1, 2, 1, 2, 1, 2, 1, 2, 1, 2)	6.67
32	(22,10)	(2493258, 2509565)	2509565	(2, 1, 2, 1, 2, 1, 2, 1, 2, 1)	8.0
36	(13, 23)	(2474474, 2469665)	2474474	(1, 2, 1, 2, 1, 2, 1, 2, 1, 2)	8.45

Table 5.6 The testing time for the system S_4 with the power constraint when the maximum power dissipation is varying (the total TAM width = 28 bits)

Maximum power dissipation Ω	Width distribution (w_1, w_2) (The best solution from our method)	Testing Time (cycles) (on $_W1$, on $_W2$)	testing time (The best solution from our method)	Test bus assignment vector	Execution time (min)
100	(7, 21)	(2544656, 2516851)	2544656	(1, 2, 1, 2, 1, 2, 1, 2, 1, 2)	6.67
200	(24, 4)	(2478663, 2497016)	2497016	(2, 1, 1, 1, 2, 1, 2, 1, 2, 1)	1.5
300	(4, 24)	(2496668, 2478771)	2496668	(2, 2, 2, 2, 1, 2, 1, 2, 1, 2)	1.5
400	(23, 5)	(2491378, 2502309)	2502309	(1, 2, 1, 1, 2, 1, 2, 1, 2, 1)	0.26
500	(4, 24)	(2497016, 2478663)	2497016	(1, 2, 2, 2, 1, 2, 1, 2, 1, 2)	0.26

5.2 Results for the SOC Test Automation with Core Wrapper Design

The proposed SOC optimization modeling and the maximum neural network (MNN) presented in Chapter 4 have been implemented on 800 MHz personal computers using MATLAB® software. The proposed MNN and the SOC optimization modeling are tested with five SOC benchmarks: d695, g1023, p34392, p22810, and p93791 from the literature in [ITC 02]. Table 4.1 and Tables 5.7 to 5.10 show the detailed testing data for all the cores used in the five practical SOC test examples. The SOC example systems d695, g1023, p34392, p22810 and p93791, consist of 10, 14, 19, 28 and 32 cores, respectively. All these five SOC benchmark are used in the implemented MNN system. For the simplicity of presentation, the first example SOC system d695 (Table 4.1) is used for illustration.

The developed MNN network is used to find the optimal testing time for the example SOC systems. Figure 5.4 compares SOC testing time for the example system d695 with two TAMs using the three different methods: the benchmarking Integer Linear Programming (ILP) [Iyengar 01a], $P_{PAW\_enumerate}$ [Iyengar 01a], and the proposed MNN. As shown in Figure 5.4, the total width W ranges from 16 bits to 64 bits. Table 5.11 compares the SOC testing time and the MNN computation time by different methods. The SOC testing time found by the MNN is close to the reported SOC testing time from ILP and $P_{PAW\_enumerate}$ in [Iyengar 01a]. In Figure 5.4, when the TAM widths are increased, the resulting SOC testing times of the example system are decreased. The computation time of the proposed MNN is less than the computation time of ILP, but slightly higher than the computation time of $P_{PAW\_enumerate}$. Notice that in Table 5.11, unlike for the computation time of ILP and $P_{PAW\_enumerate}$, the computation time of the proposed MNN is not increased with the increment of TAM width.

Figures 5.5 to 5.7 show the comparison of computation time of all the different methods using the five different benchmarking SOC examples. As shown in Figures 5.5 and 5.6, the computation time of the proposed MNN remains stable when the SOC system design becomes more complex (i.e., number of cores and TAM width are increased). When the

SOC system becomes more complex, the best solution (testing time, width allocation, and TAMs allocation) can be found by the proposed MNN within a reasonable time, as shown in Figures 5.5 and 5.6.

Figure 5.7 shows the SOC testing time (objective function) and the computation time of the proposed MNN when the number of searching iterations varies. As shown in Figure 5.7, when the number of iterations increases (that causes higher computation time), the resulting SOC testing time pretty much remains stable. From Figure 5.7, it can be seen that the proposed MNN can converge to good results of SOC testing time very rapidly without the need for a larger number of searching iterations.

Figures 5.8 and 5.9 compare the SOC testing time for the example chip d695 with three TAMs using the different existing methods: ILP [Iyengar 01a], $P_{PAW\_enumerate}$[Iyengar 01a], ECTSPSol [Koranne 02], and the proposed MNN. In Figure 5.8, when TAM width is higher than 48 bits, the ILP has the highest SOC testing time because the optimal solution cannot be found due to the high complexity of the SOC system [Iyengar 01a]. Figure 5.9 shows the comparison of the SOC testing time for the example SOC chip p93791 with 32 cores by using three different methods. For the SOC chip p93791, the SOC testing time (objective function) found by the proposed MNN is better than the results obtained by the other two methods, as shown in Figure 5.9.

The proposed MNN also has the capacity of optimizing the SOC testing system considering the power constraints. Tables 5.10 and 5.13 show the comparison of the SOC testing time (objective function) for the SOC example system d695 with and without power constraints. We assume that each TAM has maximum power consumption of 50 mW and each core consumes power 1500 mW per each pattern. The SOC testing time with power constraint (i.e., subject to more constraints in the optimization) is higher than that without power constraints, as shown in Tables 5.12 and 5.13. The computation time in solving the SOC problems also gets much larger when the power constraint is considered, as shown in Tables 5.12 and 5.13. Figure 5.10 shows the difference of SOC testing time and computation time for the example SOC system d695 with and without power constraints. Figure 5.10 and

Tables 5.12 and 5.13 show that, with the consideration of power and other constraints, the computation time needed to find the optimal solution grows significantly (thirteen times increment). While most of the existing methods (for example ILP or $P_{PAW\_enumurate}$ method) can only solve the SOC testing problems without the power constraints, the developed MNN method is able to find the optimal SOC testing solution with these additional constraints within reasonable computation time.

5.3 Summary

This chapter presented the experiment results of the proposed techniques. The results show that the proposed techniques can be applied to minimize the overall SOC testing time in three SOC test automation problems: test scheduling problem (Chapter 2), test automation with resource allocation and power constraint problem (Chapter 3), and test automation with core wrapper design (Chapter 4). The developed techniques require much less computation time to solve complex SOC testing optimization problems.

Table 5.7 Test data for each core in SOC chip g1023 [ITC 02]

Core No.	No. of Test Patterns	No. of Input	No. of Output	No. of Scan Chains	Scan Chain Length	
					Min	Max
1	134	139	273	14	42	43
2	74	221	215	2	83	84
3	57	192	171	1	53	53
4	268	145	155	4	54	54
5	51	32	27	4	31	32
6	36	20	18	2	47	47
7	34	20	18	2	47	47
8	31	63	80	2	52	52
9	68	56	34	1	64	64
10	29	301	377	1	13	13
11	15	145	191	1	9	9
12	16	157	161	1	13	13
13	512	58	64	-	-	-
14	1024	140	114	-	-	-

Table 5.8 Test data for each core in SOC chip p34392 [ITC 02]

Core No.	No. of Test Patterns	No. of Input	No. of Output	No. of Scan Chains	Scan Chain Length	
					Min	Max
1	210	15	94	1	806	806
2	514	165	263	29	8	570
3	3108	37	25	-	-	-
4	6180	38	25	-	-	-
5	12336	62	25	-	-	-
6	1965	11	8	-	-	-
7	512	9	8	-	-	-
8	9930	46	17	-	-	-
9	228	41	33	-	-	-
10	454	129	207	19	16	519
11	9285	23	8	-	-	-
12	173	7	4	-	-	-
13	2560	12	16	-	-	-
14	432	11	8	-	-	-
15	4440	22	8	-	-	-
16	128	7	7	-	-	-
17	786	15	4	-	-	-
18	745	175	212	14	198	729
19	12336	62	25	-	-	-

Table 5.9 Test data for each core in SOC chip p22810 [ITC 02]

Core No.	No. of Test Patterns	No. of Input	No. of Output	No. of Scan Chains	Scan Chain Length	
					Min	Max
1	785	28	56	10	110	130
2	12324	47	33	-	-	-
3	3108	38	26	-	-	-
4	222	48	64	-	-	-
5	202	90	112	29	27	214
6	712	80	64	-	-	-
7	2632	84	64	-	-	-
8	2608	36	16	-	-	-
9	175	116	123	24	24	122
10	38	50	30	4	2	99
11	94	56	23	8	38	88
12	93	40	23	11	42	82
13	1	68	149	4	32	104
14	108	22	15	3	1	73
15	37	84	42	6	36	80
16	8	13	43	1	109	109
17	25	223	69	4	4	89
18	644	53	11	5	56	68
19	58	38	29	3	17	43
20	124	45	40	4	1	77
21	465	115	76	10	93	186
22	59	54	40	3	13	77
23	40	31	8	7	16	115
24	27	73	23	5	2	101
25	215	58	46	18	108	181
26	181	66	33	31	198	400
27	2	285	94	1	34	34
28	26	48	43	5	40	100

Table 5.10 Test data for each core in SOC chip p93791 [ITC 02]

Core No.	No. of Test Patterns	No. of Input	No. of Output	No. of Scan Chains	Scan Chain Length	
					Min	Max
1	409	109	32	46	1	168
2	192	40	34	-	-	-
3	648	40	29	-	-	-
4	11	15	30	23	4	5
5	6127	102	80	-	-	-
6	218	417	324	46	500	521
7	177	9	32	-	-	-
8	177	9	32	-	-	-
9	192	43	34	-	-	-
10	1164	267	128	-	-	-
11	187	146	68	11	17	82
12	391	289	8	46	92	93
13	194	111	31	46	173	219
14	194	111	31	46	173	219
15	288	44	34	-	-	-
16	396	137	64	-	-	-
17	216	144	67	43	145	150
18	42	79	34	-	-	-
19	210	466	365	44	97	100
20	416	136	12	44	132	181
21	42	79	34	-	-	-
22	42	42	34	-	-	-
23	234	105	28	46	1	175
24	3072	17	4	-	-	-
25	2688	29	16	-	-	-
26	96	42	34	-	-	-
27	916	30	7	46	50	68
28	396	109	50	-	-	-
29	172	117	42	35	185	189
30	192	43	34	-	-	-
31	204	148	70	-	-	-
32	3084	268	128	-	-	-

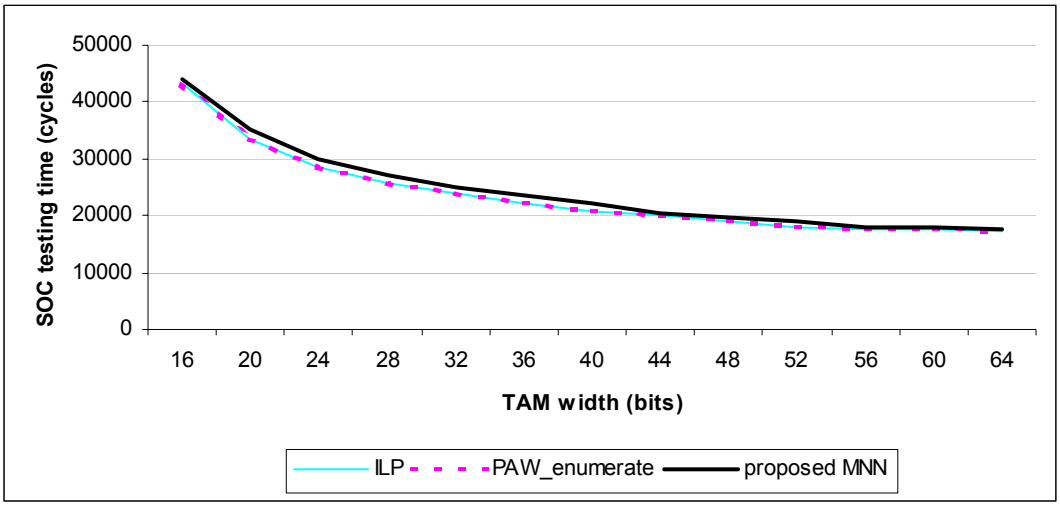


Figure 5.4 The comparison of SOC testing time for SOC d695 with two TAMs using Integer Linear Programming (ILP), $P_{PAW\text{-enumerate}}$ [Iyengar 01a], and the proposed MNN

Table 5.11 Comparison of computation time and the resulting SOC testing time for different methods

TAM width (bits)	**ILP** [Iyengar 01a]		$\mathbf{P}_{PAW\_enumerate}$ [Iyengar 01a]		The proposed MNN		
	SOC testing time (cycles)	Computation time (min)	SOC testing time (cycles)	Computation time (min)	SOC testing time (cycles)	%difference (w enumerative)	Computation time (min)
16	43238	0.7	43238	0.02	44163	2%	0.45
20	33458	0.8	33458	0.02	35289	5%	0.40
24	28626	2.1	28626	0.02	29772	4%	0.36
28	25640	3.9	25640	0.02	27043	5%	0.37
32	24030	5.23	24030	0.02	24890	4%	0.34
36	22246	11	22246	0.02	23570	6%	0.35
40	20815	12.5	20815	0.03	22248	7%	0.34
44	20094	13	20094	0.03	20324	1%	0.33
48	18911	32.1	18911	0.03	19674	4%	0.34
52	17929	50.1	17929	0.03	19026	6%	0.34
56	17671	52.8	17671	0.03	18123	3%	0.35
60	17449	76.7	17449	0.05	17935	3%	0.34
64	17375	158.7	17375	0.05	17778	2%	0.33

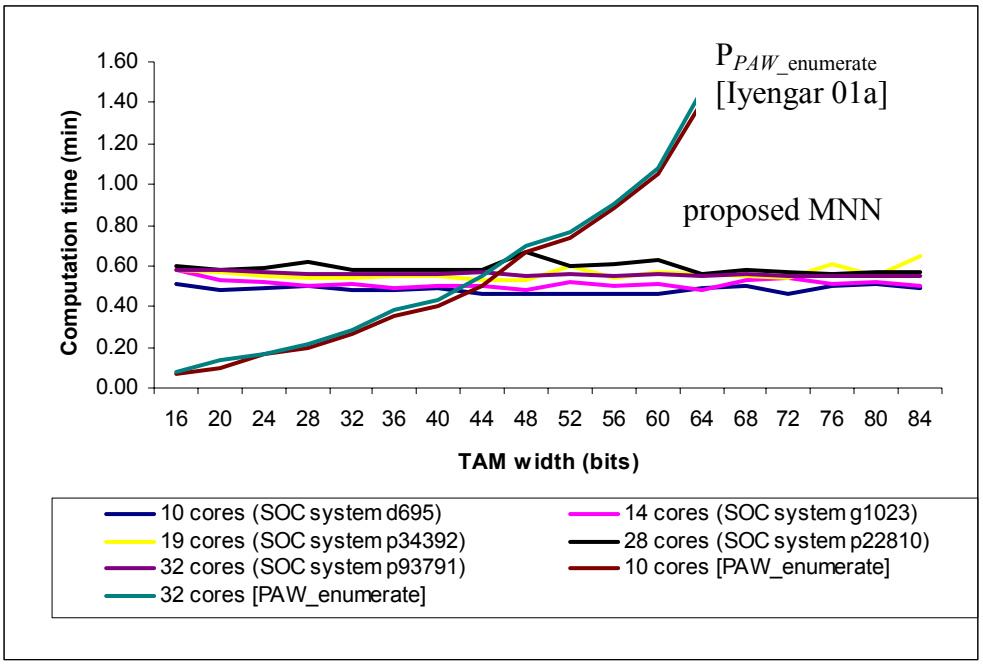


Figure 5.5 The comparison of the computation time of different methods when the number of SOC cores is increased

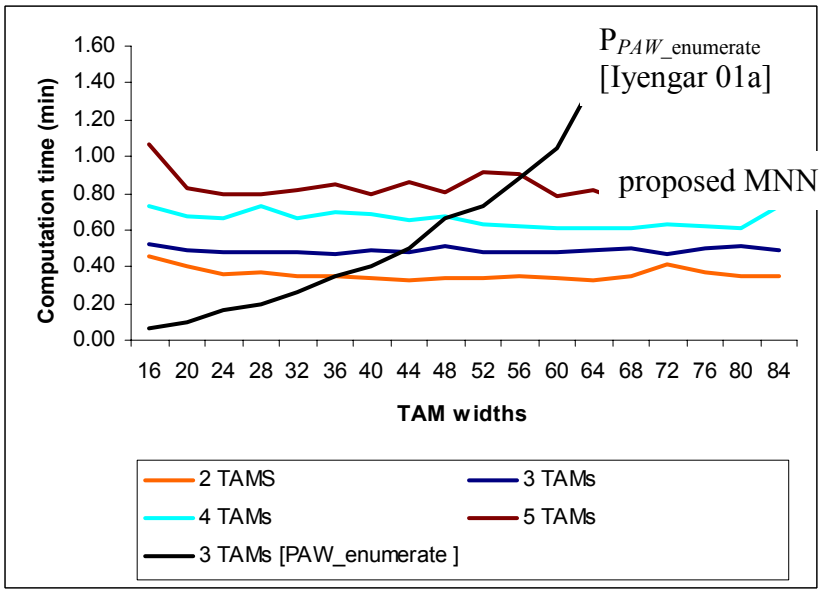


Figure 5.6 The comparison of the computation time of different methods when the number of TAM width is increased

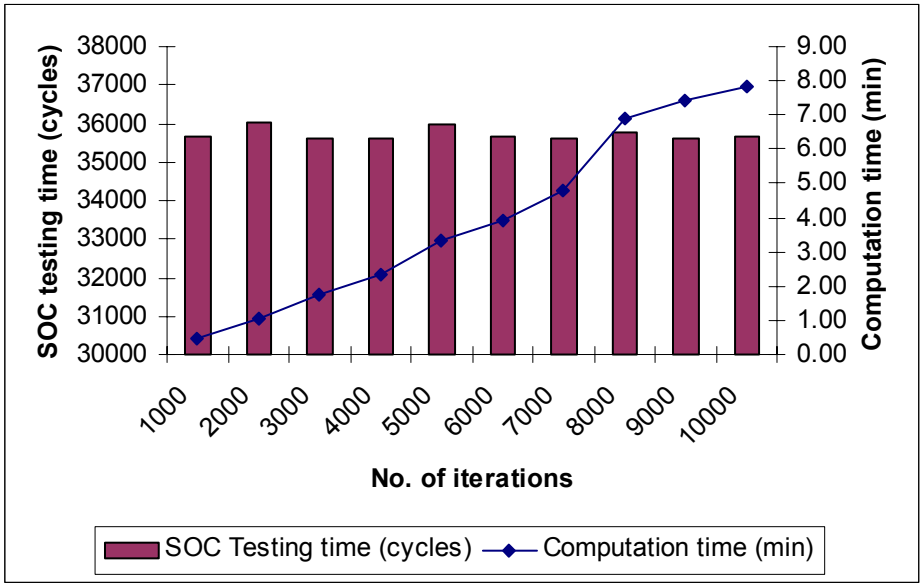


Figure 5.7 The SOC testing time (objective) and the computation time of the proposed MNN when the number of iterations varies

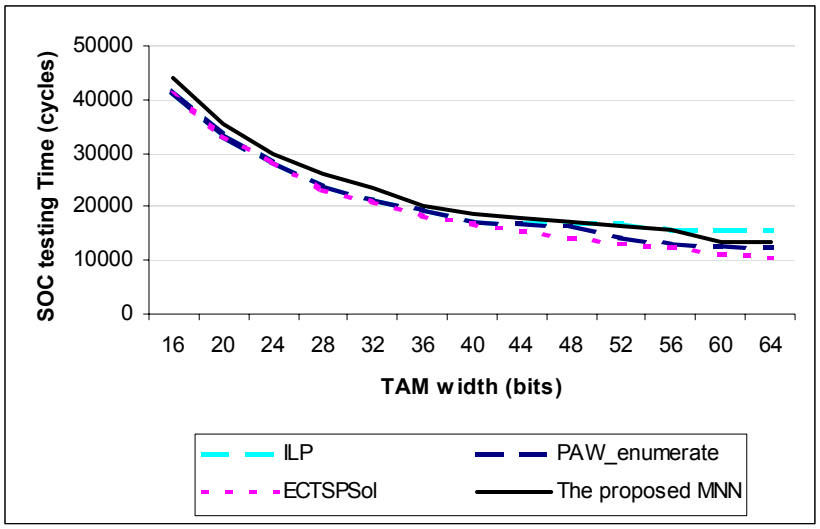


Figure 5.8 The comparison of SOC testing time for the example system d695 with three TAMs using Integer Linear Programming (ILP), $P_{PAW\text{-enumerate}}$ [Iyengar 01a], ECTSPSol [Koranne 02], and the proposed MNN

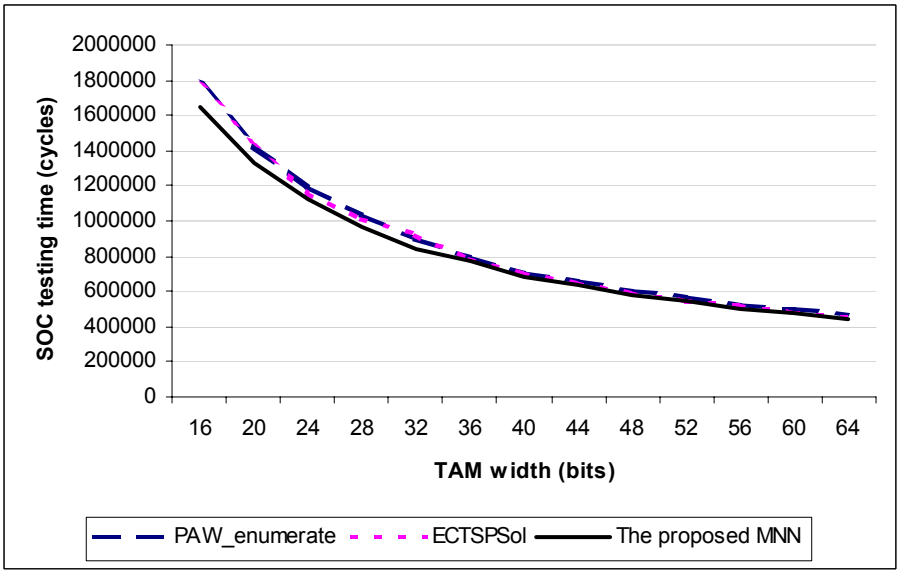


Figure 5.9 The comparison of SOC testing time for the example system p93791 with three TAMs using $P_{PAW\text{-enumerate}}$ [Iyengar 01a], ECTSPSol [Koranne 02], and the proposed MNN

Table 5.12 The comparison of SOC testing time and computation time with and without power constraint for the example system d695 with two TAMs

TAM Width (bits)	No power constraint		With power constraint	
	SOC testing time (cycles)	Computation time (min)	SOC testing time (cycles)	Computation time (min)
16	44163	0.45	45251	5.91
20	35289	0.40	36394	5.28
24	29772	0.36	30518	4.75
28	27043	0.37	27761	4.74
32	24890	0.34	25468	4.44
36	23570	0.35	24188	4.69

Table 5.13 The comparison of SOC testing time and computation time with and without power constraint for the example system d695 with three TAMs

TAM width (bits)	No power constraint		With power constraint	
	SOC testing time (cycles)	Computation time (min)	SOC testing time (cycles)	Computation time (min)
16	44029	0.52	45028	7.23
20	35606	0.49	36579	6.19
24	29699	0.48	30502	6.11
28	25936	0.48	26267	5.86
32	23578	0.48	23426	5.92
36	20325	0.47	21202	6.15

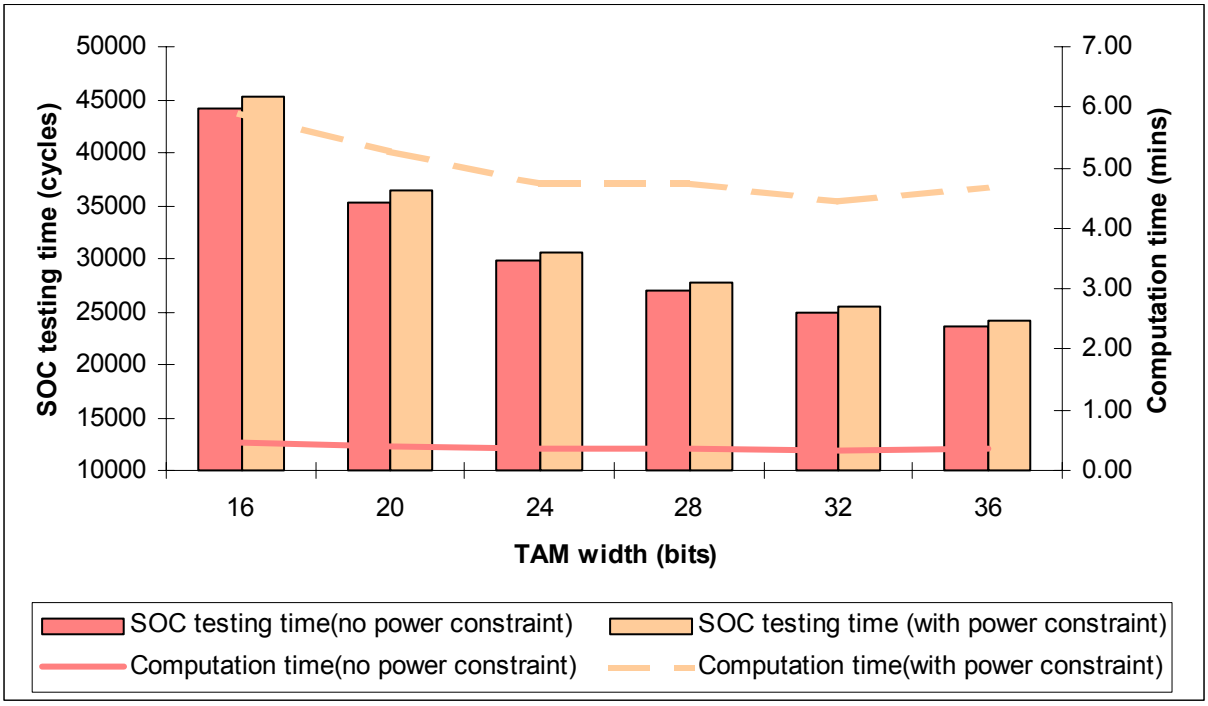


Figure 5.10 The comparison of SOC testing time and computation time for the example system d695 with and without power constraints

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

In this dissertation, different neural networks have been developed to solve the System-on-a-Chip (SOC) test automation problems with the wrapper design and optimal resource allocation. The major contributions of this research are summarized as follows:

1. To solve the NP-hard System-on-a-Chip (SOC) test design problems, a fixed-weight neural network (NN) combined with heuristic algorithms has been proposed. The developed neural network can effectively solve the SOC test scheduling problems with disjunctive constraints. The SOC design test scheduling problems are subject to four different constraints: (i) precedence constraint, (ii) resource constraint, (iii) core constraint, and (iv) power constraint. To prevent the proposed neural network from getting trapped in local optimal, some heuristic algorithms have been developed. The results show that the proposed method can effectively solve a large size SOC test automation problem within a reasonable computational time.

2. The proposed maximum neural network has been developed to solve the resource allocation problem for SOC test automation. The proposed maximum neural network can solve the NP-hard SOC test automation problems in a polynomial time. The results show that the overall testing time of the SOC testing system can be minimized with the optimal resource allocation and the optimal partitioned TAMs width. This method also requires significantly less computation time in solving larger size SOC testing problems compared to the traditional methods.

3. The resource allocation problem for SOC test automation is extended by adding an embedded core wrapper design issue. After embedded core wrapper scan chains are designed, the best solution of SOC testing time can be solved by the

developed maximum neural network algorithms in a polynomial time. Computer implementation and practical benchmark SOC examples were also presented. The results show that the overall testing time of the SOC testing system can be minimized with consideration of the following constraints: (i) the resource allocation, (ii) the power consumption constraint, and (iii) the optimal partitioned TAMs width. Compared with the traditional methods, the presented maximum neural network technique requires significantly less computation time to solve complex SOC testing optimization problems with a large number of cores and TAMs width.

Computer implementation and results show that the developed neural network can minimize the SOC testing time subject to various constraints. The computation time from the developed method is considerably less than the traditional methods (i.e., ILP and heuristic algorithm). The techniques presented in this dissertation can be used in the computer-aided test automation for System-on-a-Chip (SOC) design.

For future research, the following are several suggested directions:

1. Optimize the SOC test scheduling problems by considering more constraints such as overhead area, pre-emption constraint, etc.

2. Optimize the SOC test automation with resource allocation problems by considering more complex systems such as cores embedded in another core, split and merge TAM, etc.

REFERENCES

[Aerts 98] Aerts, J., and Marinissen, E.J., "Scan Chain Design for Test Time Reduction in Core-Based Ics," *Proceedings IEEE International Test Conference*, Washington DC, USA, 18-23 October 1998, pp 448-457

[Aikyo 00] Aikyo, T., "Embedded Tutorial: Issues on SOC Testing in DSM Era," *Proceedings of the Design Automation Conference*, *Asia and South Pacific*, Yokohama, Japan, 2000, pp. 515.

[Bagchi 01] Bagchi, D., RoyChowdhury, D., Mukherjee, J., and Chattopadhyay, S., " A Novel Strategy to Test Core Based Designs," *Fourteenth International Conference on VLSI Design*, Bangalore, India, 3-7 January 2001, pp 122-127.

[Benso 00] Benso, A., Chiusano, S., Di Carlo, S., Prinetto, P., Ricciato, F., Spadari, M. and Zorian, Y., "HD2BIST: a Hierarchical Framework for BIST Scheduling, Data patterns delivering and diagnosis in SoCs," *ITC International Test Conference*, Atlantic City, NJ, USA, 3-5 October, 2000, pp. 892-901.

[Chandramouli 96] Chandramouli, R., Pateras, S., "Testing Systems on a Chip," *IEEE Spectrum*, November 1996, pp. 42-47.

[Chakrabarty 00a] Chakrabarty, K., "Design of System-on-a-Chip Test Access Architectures using Integer Linear Programming," *Proceedings of VLSI Test Symposium*, 30 April- 4 May, Montreal, Quebec, Canada, 2000, pp 127-134.

[Chakrabarty 00b] Chakrabarty, K., "Design of System-on-a-Chip Test Access Architectures under Place-and-Route and Power Constraints," *Proceedings of Design Automation Conference*, 2000, pp 432-437.

[Chakrabarty 00c] Chakrabarty, K., "Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 19, no. 10, October 2000, pp. 1163-1174.

[Chauhan 99] Chauhan, P., Clarke, E.M., Lu, Y., and Wang, D., "Verifying IP-Core based System-On-Chip Designs," *Proceedings of ASIC/SOC conference*, Washington DC, USA, 15-18 September 1999, pp. 27-31.

[Crouch 99] Crouch, A., *Design-for-Test for Digital IC's and Embedded Core Systems*, Prentice Hall, 1999.

[Daeje 98] Daeje C., "Executing System On A Chip: Requirements for A Successful SOC Implementation," *Electron Devices Meeting,* San Francisco, USA*,* 6-9 December 1998, pp. 3-8.

[Dagli 94] Dagli, C.H., *Artificial Neural Networks for Intelligent Manufacturing*, Chapman&Hall, 1994.

[Ebadi 01] Ebadi, Z.S., and Ivanov, A., "Design of an Optimal Test Access Architecture Using a Genetic Algorithm," *Proceedings of 10th Asian Test Symposium*, Kyoto, Japan, 19-21 November, 2001, pp. 205-210.

[Fausett 94] Fausett, L., *Fundamentals of Neural Networks: Architectures, algorithms, and applications,* Prentice Hall, 1994.

[Flores 99] Flores, P., Neto, H., Chakrabarty, K., Marques-Silva, J., "Test Pattern Generation for Width Compression in BIST," *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems,* Orlando, FL, USA, 30 May-2 June, 1999, vol. 1, 1999, pp. 114-118.

[Foo 88] Foo, Y.-P. S., and Takefuji, Y., "Stochastic Neural Networks for Solving Job-shop Scheduling. I. Problem Representation," *Proceedings*

of the 1988 IEEE International Conference on Neural Networks, San Diago, California, 24-27th July 1988, vol. 2, pp. 275-282.

[Galan-Marin 01] Galan-Marin, G. and Munoz-Perez, J., "Design and Analysis of Maximum Hopfield Networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, 2001, pp. 329-339.

[Gallaher 01] Gallagher, P., Chickermane, V., Gregor, S., and Pierre, T.S., "A Building Block BIST Methodology for SOC Designs: A Case Study," *ITC International Test Conference*, Baltimore, MD, USA, 30 October – 1 November 2001, pp. 111- 120.

[Gupta 97] Gupta, R.K. and Zorian, Y., "Introducing Core-Based System Design," *IEEE Design & Test of Computers*, vol. 14, no. 4, 1997, pp. 15-25.

[Irion 01] Irion, A., Kiefer, G., Vranken, H., and Wunderlich, H.-J., " Circuit Partitioning for Efficient Logic BIST Synthesis," *IEEE Proceedings of Design, Automation and Test in Europe,* Munich, Germany, 13-16 March, 2001, pp. 86-91.

[Iyengar 01a] Iyengar, V., Chakrabarty, K., and Marinissen, E.J., "Test Wrapper and Test Access Mechanism Co-Optimization for System-On-Chip," *ITC International Test Conference*, *IEEE*, Baltimore, MD, USA, 2001, pp. 1023-1032.

[Iyengar 01b] Iyengar, V., and Chakrabarty, K., "Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-Chip," *IEEE VLSI Test Symposium*, *19th IEEE Proceedings on VTS, 29 April-3 May, Marina Del Rey, CA, USA*, 2001 pp. 368-374.

[Iyengar 02] Iyengar, V., Chakrabarty, K., and Marinissen, E.J., " On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization,"

Proceedings 20th IEEE VLSI Test Symposium, Monterey, CA, USA, 28 April- 2 May, 2002, pp. 253-258.

[ITC 02] ITC'02 SOC Test Benchmarks examples, http://www.extra.research.philips.com/itc02socbenchm

[Jain 98] Jain, A.S., and Meerran, S., "Job-shop scheduling using neural networks," *International Journal of Production Research*, vol. 36, no. 5, 1998, pp. 1249-1272.

[Kartalopoulos 96] Kartalopoulos, S.V., *Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications*, The Institute of Electrical and Electronics Engineers, Inc., New York, 1996

[Kloypayan 02a] Kloypayan, J., and Lee, Y.S., "Optimizing Test Scheduling for The System-on-a-Chip (SOC) Design by Neural Network Approach," *Proceedings of The 2002 Industrial Engineering Research Conference,* Orlando, FL, May 19-21, 2002, Paper Number (CD): CD#2282.

[Kloypayan 02b] Kloypayan, J., and Lee, Y.-S., "Modeling and Solving The System-on-a-Chip (SOC) Design Test Using Neural Network Approach," (under review), Submitted to *Journal of Manufacturing Engineering*, January 2002, (26 pages).

[Kloypayan 02c] Kloypayan, J., and Lee, Y.-S., "System-On-a-Chip (SOC) Test Automation and Resource Allocation by Maximum Neural Network Approach," (in review) Submitted to *Computers & Industrial Engineering*, May 2002, (24 pages).

[Kloypayan 02d] Kloypayan, J., and Lee, Y.-S., " Optimizing System-on-a-chip (SOC) Test Automation with Core Wrapper Design by Maximum Neural Networks," (working paper), 2002, (31 pages).

[Koranne 02] Koranne, S., “Design of Reconfigurable Access Wrappers for Embedded Core Based SOC Test,” *Proceedings of the International Symposium on Quality Electronic Design*, San Jose, CA, USA, 18-21 March 2002, pp. 106-111.

[Lee 00] Lee, Y.S., Fang, S.C., and Chiu, N.C., "Soft Computing for Optimal Planning and Sequencing of Mill-Turn Machining Operations," *The Handbook of Computational Intelligence in Manufacturing*, Edited by A. Kusiak and J. Wang, (ISBN #0-8493-0592-6), CRC Press LLC, Boca Raton, FL, 2000, pp. 8-1 to 8-33.

[Marinissen 98] Marinissen, E. J., Arendson, R., Bos, G., Dingemanse, H., Lousberg, M. and Wouters, C., “A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores,” *Proceedings of International Test Conference*, Washington DC, USA, 18-23 October 1998, pp. 284-293.

[Marinissen 00] Marinissen, E.J., Goel S.K., and Lousberg, M., “Wrapper Design for Embedded Core Test,” *ITC International Test Conference*, Atlantic City, NJ, USA, 3-5 October, 2000, pp. 911-920.

[Nourani 00] Nourani, M., and Papachristou, C., “An ILP Formulation to Optimize Test Access Mechanism in System-on-Chip Testing,” *ITC International Test Conference*, Atlantic City, NJ, USA, 3-5 October, 2000, pp. 902-910.

[Pino 96] Pino, J.L., Williamson, M.C., and Lee, E. A., “Interface Synthesis in Heterogeneous System-Level DSP Design Tools,” *IEEE International Conference on Acoustics, Speech, and Signal Processing,* Atlanta, GA, May 1996, (4 pages).

[Rajsuman 00] Rajsuman, R., *System-on-Chip: Design and Test*, Artech House Publishers, 2000.

[Sabuncuoglu 96] Sabuncuoglu, I., and Gurgun, B., "A neural network model for scheduling problems," *European Journal of Operation Research*, vol. 93, 1996, pp. 288-299.

[Sellers 96] Sellers, D.W., "A Survey of Approaches to the Job Shop Scheduling Problem," *28th Southeastern Symposium on Proceedings of System Theory, 31 March- 2 April, Baton Rouge, LA, USA*, 1996, pp. 396-399.

[Shaikh 00] Shaikh, S.A., Khare, J., and Heineken, H.T., "Manufacturing and Testability Oriented Synthesis," 13th *International conference on VLSI Design*, 2000, pp. 185-191.

[Shubat 01] Shubat, A. "Moving the Market to Embedded Memory," *IEEE Design & Test of Computers*, vol. 18, no. 3, 2001, pp.5-6

[Smeda 99] Smeda, A.A., and El-Hawary, M.E., " Application of Hopfield Neural Network in Routing for Computer Networks," *Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering, Shaw Conference Center,* Edmonton, Alberta, Canada, May 9-12, 1999, pp. 145- 149.

[Sugihara 98] Sugihara, M., Date, H., and Yasuura, H., "A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem," *Proceedings IEEE International Test Conference*, Washington DC, USA, 18-23 October 1998, pp. 465-472.

[Takefuji 92] Takefuji, Y., Lee, K-C., and Aiso, H., "An artificial maximum neural network: a winner-take-all neuron model forcing the state of the system

in a solution domain," *Biological Cybernetics*, vol. 67, 1992, pp. 243-351.

[Temple 02] Temple University, System Chip Design Center, http://www.temple.edu/scdc/

[Wang 00] Wang, J., and Kusiak, A., *Computational Intelligence in Manufacturing Handbook*, CRC Press LLC, 2000.

[Yang 00] Yang, S., and Wang, D., "Constraint Satisfaction Adaptive Neural Network and Heuristics Combined Approaches for Generalized Job-Shop Scheduling," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, 2000, pp. 474-486.

[Varma 98] Varma, P. and Bhatia, S., "A Structured Test Re-Use Methodology for Core-Based System Chips," *Proceedings of International Test Conference*, Washington DC, USA, 18-23 October 1998, pp. 294-302.

[Yu 97] Yu H., Wang, H., Xu, X., and Xue, J., "A Neural-based Approach to Production Scheduling," *Proceedings of the 1997 American Control Conference,* Albuquerque, New Mexico, USA, June 1997, vol 2, pp. 1027-1031.

[Yu 01] Yu, H., Wu-Tung, C., Chien-Chung, T., Mukherjee, N., Samman, O., Zaidan, Y., and Reddy, S.M., "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," *The Tenth Asian Test Symposium,* Kyoto, Japan, 2001, pp. 265-270.

[Zorian 97] Zorian, Y., "Test Requirements for Embedded Core-Based Systems and IEEE P1500," *Proceedings of International Test Conference*, Washington DC, USA, 1-6 November, 1997, pp. 191-199.

[Zorian 98] Zorian, Y., "System-Chip Test Strategies," *Proceedings of Design Automation Conference*, San Francisco, CA, USA, 15-19 June, 1998, pp. 752-757.

[Zorian 99] Zorian, Y., Marinissen, E.J., and Dey, S., "Testing Embedded-Core-Based System Chips," *Computer*, vol. 32, no. 6, 1999, pp. 52-60.