OPTIMISING SOLUTION OF THE SCAN PROBLEM AT RT LEVEL BASED ON A GENETIC ALGORITHM

Josef Strnadel
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno
Czech Republic
strnadel@fit.vutbr.cz

Zdeněk Kotásek
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno
Czech Republic
kotasek@fit.vutbr.cz

Abstract. *The paper deals with the problem of selecting registers into a scan chain, the problem is solved on RT level. As a result of the methodology, it is not only stated which registers shall be modified into scan registers but also how registers will be organized into sections, namely how registers will be subdivided and ordered in sections. The partial scan problem is defined and seen as a combinatorial problem, a mathematical formula is used to demonstrate it. The problem of selecting registers for scan chain is solved through genetic algorithm. The methodology was implemented and verified on DIFFEQ benchmark circuit. Experimental results are compared with results gained in other approaches.*

1 Introduction

Testability analysis procedures are usually implemented either on behavioural or RT level. The tools based on behavioural level analyse the VHDL source program and identify such constructions which generate complicated designs. It is important to note that the constructions revealed are those which are not recognised during the syntactical and semantic analysis as possibly problematical. From this point of view, these tools can be understood as tools which provide the corrections of the VHDL source file [11].

Research activities have been also targeted towards test pattern generation at the high level. In [5] the VHDL open platform for TPG and fault simulation algorithms at the behavioural level is described. The platform consists of editor and browser.

Techniques that improve testability using high-level descriptions analyse testability during high-level synthesis. However, these techniques are tightly linked to design tools. Generalising them to a large set of commercially available synthesis tools is not possible.

Many approaches to the RTL testability analysis were demonstrated so far. Most of them are based on the UUA structural analysis [4]. Recently, the methodologies whose goal is to analyse VHDL design description and insert scan at the behavioural level were presented. The methodology is based on locating memory elements in the circuit. Then, all located memory elements are inserted in the *scan chain* at the behavioural level. The approach is supposed to be generalised to cover multiple scan chains [1].

Scan approaches fall into two main groups: *full scan* and *partial scan*. In partial scan, FFs are selected in such a way that the remainder of the circuit has certain desirable testability properties. Existing approaches for selecting FFs for partial scan can be classified as *testability analysis based* [2], *test generation based* [3] and *structural analysis based* [4]. All of these techniques suggest testability modifications after the completion of the design and are incapable of suggesting behavioural modifications by identifying testability

bottlenecks in the behaviour during the design process. Some methods exist which are based on inserting test registers in order to obtain self-testable circuits [6], [7]. The methodologies are implemented in the way which guarantees minimum hardware overhead [8]. In [9] an interesting method how to further utilise scan chains is demonstrated - they are used for pattern decompression.

2 Definition of the Problem and Basic Concepts

Many of the design for testability approaches that have been proposed in the past use scan based schemes while others use non-scan techniques to suggest improvements to the design. They are based on the analysis of the circuit structure and searching for parallel paths along which the diagnostic data (test vectors and responses to them) can be transferred. Different aspects (criteria) are taken into account which allow various solutions of the problem to be gained. As a result of such approaches, a subset of registers is identified through which the test will be applied. It is supposed that the registers will be converted either to *scan registers* or *BIST elements*.

Let it be noted now that one possibility how to identify registers for the test application process is through enumerating all combinations of registers which could possibly form the scan chain(s) and evaluate every alternative. These approaches are usually denoted as "rough methods" (although they lead to acceptable solutions, they can be too much time consuming for bigger problem complexity). These methods are based on *exhaustive search*: they simply visit all points in the search space in some order and retain the best solution visited. Other methods only visit part of the search space, albeit the number of points visited may grow exponentially (or worse) with the problem size. To avoid this, it is possible to use heuristics that do not guarantee an optimal solution in general.

In this paper it is presented how *genetic algorithms* can be used for testability improvement at RT level. A genetic algorithm performs a multidirectional search by maintaining a population of potential solutions. The population undergoes a simulated evolution from one generation to another: at each generation the relatively good solutions reproduce, while the relatively bad solutions die. The goodness or badness of the solutions is determined by a *fitness* function. Each solution is encoded as a *chromosome* which is represented as a string of bits from a binary alphabet.

Producing several successive chromosome generations, the average fitness of the solutions is increasing. The algorithm is usually stopped after a certain number of iterations or when no further improvements are produced. The best solution that has been produced is one which is hopefully close to the optimum. To apply GA to a problem, it is necessary to identify: (1) meaningful representation for the candidate solutions; (2) a fitness function to assess different solutions; and (3) a set of useful genetic operators, that can efficiently recombine and mutate candidate solutions.

Not very much attention was devoted so far to the use of evolution algorithms to solve diagnostic problems. This paper is an attempt to show that genetic algorithms can be seen as a tool which is appropriate for these purposes. The goal of the paper is to show a completely new approach to the partial scan problem.

The paper is organised as follows. First, the partial scan problem is defined, attention is paid to the time complexity of solution finding. Then the methodology based on genetic algorithm together with experimental results is presented. Finally, the perspectives of future activities in this research area are discussed. Subsection in the second chapter

3 Partial Scan Problem

Let *UUA* (*Unit Under Analysis*) contain *n* parallel registers. Let a register be marked as $PREG_i$, where $i \in \{1,2,\ldots,n\}$. The set of all parallel registers in *UUA* will be defined as $PREGS = \{PREG_1, PREG_2, PREG_3, \ldots, PREG_n\}$. One of the possibilities how to provide diagnostic data (test patterns and responses to them) transfer through the *UUA* structure and thus improve its diagnostic properties (in terms of principles presented in [10]) is through the identification and modification of the selected subset of registers $SCAN \subseteq PREGS$ into scan registers (each selected $PREG_i$ to *SREGi*).

Then, besides the original function of a parallel register, scan register fulfils also the function of a shift register through which test patterns are applied and test responses observed. The register has then additional extra inputs and outputs which form serial path through the set of registers. It is a well known fact that the conversion of registers into scan chain will certainly cause the decrease of the UUA "quality" in terms of worse dynamic parameters, chip area overhead, lower yield, additional extra primary inputs/outputs, longer test application times, etc. It can be stated that all these negative aspects are strongly affected by the number of registers included into the scan chain.

It should be also noted that in a *UUA* more scan chains can be configured (see p_2 solution in Fig. 3). Shorter test application times are seen as an advantage of such approaches while more primary inputs/outputs needed to implement these methodologies can cause certain problems. In the methodology presented in this paper we take into account the *controllability, observability, testability, area overhead*, *pin overhead* and other parameters of resulting *UUA* structure – including the sequential depth and maximal cycle length, the number of scan chains, the number of scan registers in chains and the organisation of registers in scan chains to evaluate the impact of converting registers into scan registers.

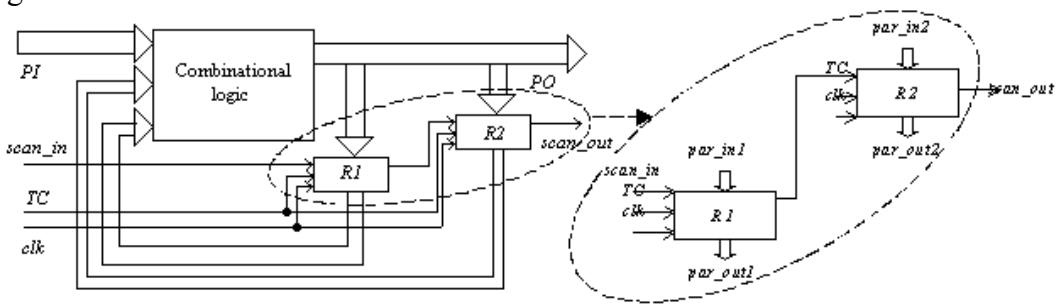

Fig. 1 – Scan chain example in a sequential circuit and its simplified schema

It is evident that many problems exist combined with the implementation of partial/full scan methods. The goal of our research activities is: to define partial scan problem, deal with its analysis and suggest solutions of the problem. For our purposes we defined the partial problem of counting scan alternatives as follows:

What is the number of alternatives in which *n UUA* registers can be organised into scan chains when it is supposed that 1) More than one scan chain may exist in the *UUA* and 2) the organisation of scan registers in scan chains is taken into account and 3) every register can be included into one scan chain.

Let the set of all the alternatives for *UUA* with *n* registers be denoted as $PSPI_n$. Then the partial problem under the above described assumptions is defined as "$|PSPI_n| = ?$".

Thus**,** every element of $PSPI_n$ set represents a particular set of scan chains (each scan chain includes particular scan registers in given order) which will be formed in the *UUA*.

Before we present a solution to the problem of $PSPI_n$ set cardinality, it is necessary to define *scalts(m)* function returning the number of alternatives in which exactly *m* selected (given) *UUA* registers can be organised into scan chains. Let this function be defined as:

$$nscalts(m) = \sum_{i=1}^{m} \left[\frac{m!}{i!} \times \binom{m-1}{i-1} \right] \qquad (1)$$

The formula which represents the answer to the question "What is the size of $PSPI_n$ set?" has the following form:

$$|PSPI_n| = \sum_{m=0}^{n} \left[\binom{n}{m} \times scalts(m) \right] \qquad (2)$$

The alternatives of scan chain configuration from $PSPI_n$ set represent solutions which have different quality level. The most important problem to be solved is still the problem of selecting the most appropriate solution of the scan organisation (i. e. selecting the best alternative from $PSPI_n$ set). It is covered by the *partial scan problem*.

Partial scan problem is defined as follows: Which alternative from $PSPI_n$ set is the most appropriate one when the test application quality and price are taken into account.

Fig. 2 – Definition of partial scan problem

For better understanding the complexity of the $PSPI_n$ set cardinality problem, the formula for evaluating $|PSPI_n|$ is completed with a a table (see Tab. 1). The symbols in the rows of the Tab. 1 have the following meanings: 1) *n* - the number of registers in *UUA*, 2) $|PSPI_n|$ - the total number of alternatives, in which various scan chain configurations can be formed, 3) t_{all} - time needed to identify the best solution in $PSPI_n$ set using "rough method", 4) t_{gen} - time needed to identify the best solution through our methodology based on genetic algorithm. The times written in rows 3) and 4) are valid for PC with PentiumII/333MHz processor, 64MB RAM.

Tab. 1 – Example of a partial scan problem comlexity

n	1	2	3	4	5	6	7	8	9	10
$\|PSPI_n\|$	1	5	25	147	1031	8463	79591	842831	9914335	128162463
t_{all}	1s	5s	25s	2min	17min	2,3h	22h	10 days	4 months	4 years
t_{gen}	5s	10s	25s	90s	3min	7min	30min	90min	9h	1day

4 Partial Scan Methodology Based on Genetic Algorithm

The goal of our research was defined in the previous section. We try to optimise the process of searching for the most appropriate solution of the partial scan chain configuration and thus decrease the time complexity of the searching process. For this purpose we utilised a genetic algorithm which allows to describe the problem by means of a binary string. The advantage of the approach can be primarily seen in the fast convergence of algorithms developed for these purposes.

To be able to utilise a genetic algorithm for the partial scan problem solution, it was necessary to represent the problem by means of a binary string, denoted as chromosome. Let it be reminded that for *UUA* with *n* registers $|PSPI_n|$ alternatives exist how to create various scan chain(s) configurations. Besides, it is required any element from $PSPI_n$ set to

be addressable by means of the chromosome. The chromosome $ch \in \{0,1\}^+$ is defined as a bit string with the length $\lceil log_2 |PSPI_n| \rceil$.

Now a chromosome can be seen as a unique address of an element in $PSPI_n$ set, i.e. a unique address of a possible solution of scan chain(s) organisation; there are $|PSPI_n|$ possible addresses, the address can acquire a value from <0; $|PSPI_n|$-1> interval. It means that a particular chromosome represents concrete solution of partial scan problem. So, a chromosome is a unique identification (address) of a solution (from $PSPI_n$ set) which means unique identification of 1) a number of scan chains that will be inserted into a *UUA*, 2) a set of *UUA* parallel registers that will be modified into scan registers, 3) scan registers that will be included in given scan chain and 4) an order of scan registers in scan chain. Thus, there are $|PSPI_n|$ distinct chromosomes and it can be proved that a bijection (see Fig. 3) from the set of chromosomes to $PSPI_n$ set exists.

It can be derived that if a chromosome represents an address of an element in $PSPI_n$ set then by generating a chromosome *ch* (its bits) we gain an address of $pspi_n \in PSPI_n$ element which corresponds to *ch* contents.

As every element in $PSPI_n$ set represents a concrete set of scan chains which should be configured in *UUA* and a chromosome represents the address of the element then a chromosome can be seen as a prescription how to modify *UUA*. The prescription determines the registers which will be modified into scan registers, how many scan chains will be configured in *UUA* which scan registers will be involved into particular scan chains and what will be their order.

Example for $n = 2$:
$|PSPI_2| = 5 \approx 5$ chromosomes
$PSPI_2 = \{p_0, p_1, p_2, p_3, p_4\}$.

Address	=	chromosome
0. (000)	=	$chromosome_0$
1. (001)	=	$chromosome_1$
2. (010)	=	$chromosome_2$
3. (011)	=	$chromosome_3$
4. (100)	=	$chromosome_4$
5. (101)	}	unused
6. (110)	}	unused
7. (111)	}	unused

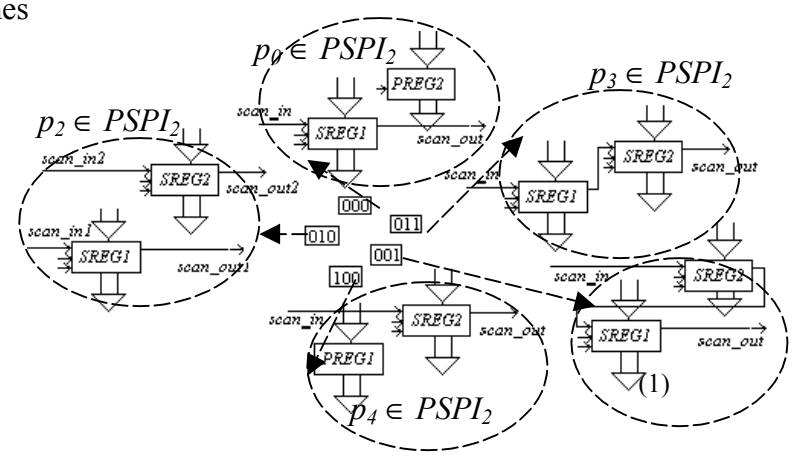

Fig. 3 – Complex example: bijection from the chromosome set to $PSPI_2$ set

It is also necessary to evaluate which chromosome from $PSPI_n$ set represents the *UUA* transformation offering the best diagnostic properties. For these purpose, the fitness function [10] was introduced which assigns a value from <0; 1> interval (fitness value) to the particular chromosome. The fitness value is proportional to the quality of the *UUA* transformation (testability factors, chip area overhead, number of I/O pins, test application time are evaluated in the fitness function). Thus, two chromosomes (*UUA* transformations) can be compared on the basis of their fitness values. The solution we are searching for is represented by a chromosome with a maximal fitness value, i. e. the chromosome with the highest possible fitness value in <0; 1> interval for given *UUA*. A genetic algorithm can be used to find such chromosome for given *UUA*. The genetic algorithm can be understood as

a "competition of chromosomes representing various *UUA* modifications" in which the chromosome with the highest fitness value wins.

As a distinctive advantage of utilising genetic algorithm to investigate the search space of chromosomes (compared with other methods used for these purposes) we see the fact that it enables the problem to be defined as a bit string - chromosome and that the algorithm based on genetic algorithm converges rapidly to a sub-optimal (acceptable) solution. The solution is a chromosome describing the *UUA* modification with the acceptable *UUA* diagnostic properties which can be used by implementing scan approach. Let it be noted that in some cases it is not necessary to identify the best chromosome but as an acceptable solution we can see the *UUA* modification which satisfies the designer requirements.

5 Experimental results

Our method was verified on *DIFFEQ* benchmark circuit. Experimental results for *DIFFEQ* benchmark circuit are shown in the following two tables (Tab. 2, Tab. 3). There are 6 registers in original *DIFFEQ* structure, so $n = 6$. According to Tab. 1, $|PSPI_6| = 8463$ and the time t_{all} to gain results fulfilling given criteria by means of exhaustive search is 2,3 hour. The purpose of our genetic algorithm is to find the same solution in a shorter time t_{gen}.

In Tab. 2, a relation among N (population size), number of generations (iterations) of genetic algorithm and t_{gen} (genetic algorithm CPU time) is shown. It can be seen, that all t_{gen} values gained by the proposed genetic algorithm for n=6 are much lower than t_{all} (=2,3h) in Tab. 1. The higher n the bigger difference between t_{gen} and t_{all} can be seen, but still $t_{gen} << t_{all}$ for $n>>1$. For *DIFFEQ* circuit case (n=6), it can be seen that the number of generations needed to find a solution for different N does not differ substantially, so it can be stated that the number of generations will not decrease considerably with N increasing. Based on experiments with our genetic algorithm, we recommend to use N from 20 to 40 interval, but still such situations can appear in which also different values of N are better than the recommended one. As a solution, two *DIFFEQ* registers (*PREG1* and *PREG6*) were suggested by genetic algorithm for modification to scan registers *SREG1* and *SREG6* and for being involved into one scan chain, where *SREG6* is the first scan register in scan chain and *SREG1* is the second register in scan chain.

Tab. 2 – Time requirements of proposed genetic algorithm and for *DIFFEQ* circuit (n=6)

	***N*=20**	***N*=40**	***N*=60**	***N*=80**	***N*=100**
Number of generations	16	15	14	13	14
t_{gen} (CPU time for genetic alg.)	5min20s	9min	13min	17min	23min

In Tab. 3, columns "Original *DIFFEQ structure*", "Modified *DIFFEQ* structure[1]", "Modified *DIFFEQ* structure[2]" and "*DIFFEQ structure with full scan*" belong to original *DIFFEQ* structure, modified *DIFFEQ* structure based on the solution by means of the proposed genetic algorithm, modified *DIFFEQ* structure based on the solution by means of [12] and *DIFFEQ* structure with built-in full scan (e.g. all 6 *DIFFEQ* registers are modified to scan registers and inserted into one scan chain with this order of scan registers: *SREG6, SREG5, SREG3, SREG1, SREG4, SREG2*).

Tab. 3 – Experimental results gained with different methodologies

	Original *DIFFEQ* structure	**Modified *DIFFEQ* structure[1]**	**Modified *DIFFEQ* structure[2]**	***DIFFEQ structure* (full scan)**
Total number of PI/PO	11/1	13/2	13/2	13/2
No. PI/PO for test purposes	0/0	2/1	2/1	2/1
Total number of nodes	71	77	83	89
No./% of controllable nodes	49/69,0%	77/100%	83/100%	89/100%
No./% of observable nodes	16/22,5%	62/80,5%	66/79,5%	70/78,7%
Average controllability of nodes	0,651	0,978	0,983	0,990
Average observability of nodes	0.197	0.956	0,974	0,983
Average testability of nodes	0.424	0.967	0.979	0.987
Depth (cyclic paths ignored)	12	6	2	2
Structure/max. cycle length	Cyclic/8	Cycle-free/0	Cycle-free/0	Cycle-free/0
Number of combinational gates	599	599	599	599
Number of nonscan FF/gates	48/480	32/320	16/160	0/0
Number of scan FF/gates	0/0	16/224	32/448	48/672
Total number of FF/gates	48/1079	48/1143	48/1207	481271
Gate overhead	0%	5,9%	11,9%	17,8%

[1] modification of *DIFFEQ* structure proposed by genetic algorithm (*SREG1, SREG6* in scan chain)
[2] modification of *DIFFEQ* structure according to [12] (*SREG4, SREG6, SREG1, SREG5* in scan chain)

It can be seen that except full-scan solution presented in the column No. 4, another methods of selecting registers into scan chains exists. Comparing results from the column No. 2 (proposed by our genetic algorithm) with the results presented by similar methods presented e.g. in [12] (see the column No. 3 of Tab. 3), [13] or in other one existing approaches it can be said that our method based on genetic algorithm has at least the same properties of selecting the best solution of this problem. As an advantages of our method we see: 1) much lower time complexity for increasing n (compare t_{all} and t_{gen} values in Tab. 1) and 2) the ability to identify solutions fulfilling demanded or max available criteria. It can be stated that the methodology presented in this paper can be seen as a completely new approach to the partial scan.

6 Conclusions

The reason for this research was to develop a completely new approach to the scan problem at RT level, implement and verify the appropriateness of the approach. In the beginning of this paper, the partial scan problem was defined and identified as a combinatorial problem and a mathematical formula was used to demonstrate the time complexity of this problem. For the time complexity of this problem was found too high,

there was a need of an optimisation technique. Thus, the optimising solution based on genetic algorithm was developed and presented in the next sections of this paper. In "Experimental results" part of this paper, the CPU time needed for finding solution by our method was shown to present the big difference between the time complexity of a non-optimised solution and our optimising solution based on genetic algorithm. Also, some of the testability properties of different circuit structures were presented in a table to show a difference between testability properties of a circuit structure based on our genetic approach and the circuit structures based on some other approaches. For the future research, there is a plan for 1) an extension of our optimising method to more test techniques than only the scan technique, 2) experimental results with more benchmark circuits and 3) an improvement of a fitness function which affects a time for solution finding the most. It is also planned to experiment with different approaches to *crossover* and *mutation* algorithms.

Acknowledgments

This work has been financially supported by the Czech Ministry of Education – FRVŠ No. 1754/2002/G1 "Application of Evolution Approaches for Digital Circuit Testability Enhancement" and the Research Intent of FEI, Brno University of Technology, Czech Republic, grant No. CEZ: J22/98:262200012 "Research of the Information and Control Systems".

References

[1] Aktouf, Ch. - Fleury, H. - Robach, Ch.: Inserting Scan at the Behavioural Level, IEEE Design & Test of Comp

[2] Agrawal, V. D.- Cheng, K. - Johnson, D. D. - Lin, T..: A Complete Solution to the Partial Scan Problem, Proc. of the 1987 International Test Conference, September 1--3, 1987, Washington, pp. 44--51

[3] Higami, Y. - Kajihara, S. - Kinoshita, K.: Partial Scan Design and Test Sequence Generation Based on Reduced Scan Shift Method, Journal of Electronic Testing: Theory and Applications, 7, Kluwer Academic Publishers, 1995, pp. 115--123

[4] Flottes, M. L. - Pires, R. - Rouzeyre, B. - Volpe, L.: A Fast and Effective Technique for Partial Scan Selection at RT Level, Proc. of IEEE ETW 1997, May 28--30, 1997, Cagliary, Italy, pp. 36--42

[5] Štefanovič, J. - Mikloš, P. - Gramatová, E.: A New Open Platform for VHDL Modelling on Behavioural Level, Proc. IEEE DDECS 2001 (Design and Diagnostics of Electronic Circuits and Systems), April 2001, Gyor, Hungary, pp. 141 - 144

[6] Abadir, M.S. - Breuer, M.A.: A Knowledge-Based System for Designing Testable VLSI Chips, IEEE Design & Test of Computers, vol. 2, No. 3, 1985, pp. 56 - 68

[7] Stroele, A. P. - Wunderlich, H. J.: Hardware-Optimal Test Register Insertion, IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, vol. 17, No. 6, 1998, pp. 531 - 539

[8] Stroele, A. P. - Wunderlich, H. J.: Test Register Insertion with Minimum Hardware Cost, Proc. ICCAD 95, San Jose, California, USA, pp. 95 -101

[9] Dorsch, R. - Wunderlich, H. J.: Reusing Scan Chains for Test Pattern Decompression, Proc. ETW 2001, Stockholm, May 2001, pp.307 – 315

[10] Kotásek, Z. – Růžička, R. – Strnadel, J.: Formal and Analytical Approaches to the Testability Analysis - the Comparison, Proceedings of IEEE DDECS 2001 (Design and Diagnostics of Electronic Circuits and Systems) Workshop, Györ, Hungary, 2001, pp. 123-128

[11] Kotásek, Z. – Růžička, R. – Sochůrek, M.: Behavioural Analysis for Testability on VHDL Source File, Proceedings of the IEEE DDECS 2000 (Design and Diagnostics of Electronic Circuits and Systems), Smolenice, Slovakia, 2000, pp. 209 - 212

[12] Bukovjan, P.: Allocation en vue de la testabilité dans le cadre de la synthèse de haut niveau, PhD thesis, Institut National Polytechnique de Grenoble, 2000

[13] Xinli Gu: RT Level Testability Improvement by Testability Analysis and Improvements, PhD thesis, Department of Computer and Information Science, Linköping University, Sweden, 1996, 149 pp.