A Survey of Testability Measurements at Various Abstraction Levels

Naghmeh Karimi
Rutgers, The State University of New Jersey
38 PUBLICATIONS **373** CITATIONS
SEE PROFILE

Zainalabedin Navabi
Worcester Polytechnic Institute
340 PUBLICATIONS **2,066** CITATIONS
SEE PROFILE

A Survey of Testability Measurements at Various Abstraction Levels

Naghmeh Karimi[+], Pedram Riyahi[*] and Zainalabedin Navabi[+]
Electrical and Computer Engineering
[+]*University of Tehran*
[*]*Northeastern University*
naghmeh@cad.ece.ut.ac.ir, priahi@ece.neu.edu, navabi@ece.neu.edu

Abstract: The new EDA tools such as high level automatic synthesis and design analysis programs require measurement of testability at one or more levels of abstraction. Depending on the application and the level of the input hardware description, we may need to measure testability of a design at the gate, RTL or behavioral level. This paper presents a survey of various testability methods at these levels. In the last section of this paper, we compare these methods for their applications, speed and complexity of algorithms.

1. Introduction

With the fast growth of the VLSI technology, VLSI testing problem has gained critical significance. By applying a Design For Test (DFT) technique on a design, its testability will be improved, although it's area and delay will be increased. In order to balance these metrics the testability measurement techniques must be considered carefully. In this paper the approaches have been classified according to design abstraction levels. We will discuss testability analysis methods at the gate, Register Transfer and behavioral levels of abstraction. The last section in this paper compares various methods and presents application of these methods in various EDA tools.

The reminder of the paper is organized as follows. Section 2 presents an overview of testability analysis methods and measures. Then Section 3 presents some gate level testability analysis methods. The testability analysis techniques at the RT level are described in Section 4; Section 5 discusses the corresponding methods at the behavioral level. Finally Section 6 presents applications of various testability analysis methods and compares the testability methods at different levels of abstractions.

2. Testability Analysis

Testability analysis is a way of showing how easy or how hard it is to test a circuit. Testability analysis is not data dependent (test data) and is only determined by the circuit structure or description. This makes the testability an intrinsic property of a circuit. The testability metrics are related to the fault coverage of a design and they are used to measure the fault sensitization and fault propagation cost during test generation.

Testability analysis can be done at the gate, Register Transfer (RT), or behavioral levels of abstraction. At the gate level, testability of a design is measured by controllability and observability. Controllability reflects the cost of setting up a specific value on a line, and observability reflects the cost of observing a specific value of a line. At the RT level, testability is mainly measured by controllability and observability at the level of vectors and RT level components. Behavioral testability is a measure of how various parts of a high-level code, or blocks of code, can be reached.

Testability measurement at the lower levels of abstraction is more accurate with a higher processing cost. On the other hand, testability measures at the higher levels involve approximations based on the topology of the circuit or coding style.

3. Gate Level Testability

Gate level testability is measured using a netlist of the circuit being analyzed. Several techniques have been proposed to measure testability at the gate level. The early work in testability analysis at this level can be seen in References 1 to 3. In these approaches the testability of each line of a design is related to its distance from the primary inputs or the primary outputs. The controllability is related to the distance of that line from the primary inputs and the observability is related to the corresponding distance from the primary outputs [4]. Although these metrics are not precise enough to demonstrate the design testability, they are used very frequently because of simplicity of computations.

Some known approaches in this level are SCOAP [2], TMEAS [5-6], COP [7], LEVEL and CAMELOT [8-9], VICTOR and TESTSCREEN [11-12].

3.1 Generic Methods

Generally, many gate-level testability techniques are based on measuring 0-controlability, 1-controllability and observability parameters. In these methods *v*-controllability is the probability of setting a line to the value *v*, and the observability is the probability of observing the value of the line on an observable point.

Figure 1 is a simple OR gate with the inputs *X*, *Y* and the output *Z*. The corresponding testability metrics can be measured as discussed below.

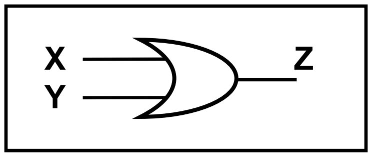

Figure 1:A simple OR gate

Where *C0 (x)* and *C1 (x)* are 1 and 0 controllability of line *x* and *C2 (x)* is the observability of that line.

$$\mathrm{C_0(Z) = C_0(X) + C_0(Y) \; + 1}$$
$$\mathrm{C_1(Z) = \min\{C_1(X)\,, C_1(Y)\} + 1}$$
$$\mathrm{C_2(X) \; = C_2(Z) + C_0(Y) + 1}$$

The testability metrics for the other gates can be computed similarly. In this approach the circuit testability is defined as:

$$TST = \sum_{i=0}^{2} k_i (\sum_{l \in L} C_i(l))$$

Where C_i *(l)* is the corresponding testability metric of Line *l*, *l* is the total number of the circuit lines and $k_i s$ are the weights assigned to the controllability and observability values.

Some techniques consider the sensitivity analysis. The sensitivity measure ranks the flip-flops relative to each other based on detectability, which is composed of controllability and observability costs [14]. Here the controllability cost *($C_v(l)$)* is the minimum number of clock cycles required to set Line *l* to value *v* and the observability cost *(O(l))* is the number of clock cycles required to set Line *l* to the value *v*. *TCF* (shown below) is the sum of the detectability of all the faults and the detectability of fault *f* stuck at *v* of Line *l* (*DET(f)*) is computed as:

$$TCF = \sum_{f \in F} DET(f)$$
$$DET(f) = \max\{C_{\bar{v}}(l), E(l)\} + D(l)$$

Where *F* is the set of the faults of the circuit, *E (l)* is the minimum controllability cost required to propagate a fault in a path from a primary input to Line *l* and *D (l)* is the number of the flip-flops in the most observable path from Line *l* to a primary output.

3.2 TMEAS

The TMEAS (Testability MEASurement) [5-6] algorithm has been developed for register-transfer-level circuits but it can also be applied at the gate level. The testability metrics are controllability and observability of the individual lines of a design. These metrics are between 0 & 1 and reflect the ease of controlling and observing the internal nodes. The controllability and observability of all lines in this method are determined by solving a system of simultaneous equations with the *CY*s and *OY*s as unknowns as discussed below. Assume a gate has x_1, $x_{2\ldots}x_n$ as its inputs and z_1, z_2… z_m as its outputs. For this element,

$$CY(z_j) = CTF \times \frac{1}{n}\sum_{i=1}^{n} CY(x_i)$$

Where *CTF* is the Control Transfer Factor of the gate and is computed as:

$$CTF \;\; = \frac{1}{m}\sum_{j=1}^{m}(1 - \frac{|N_j(0) - N_j(1)|}{2^n})$$

And $N_j(h)$ is the number of input combinations for which z_j has the value *h*.

In this method observability metric is computed as shown below.

$$OY(x_i) = OTF \times \frac{1}{m}\sum_{j=1}^{m} OY(z_j)$$

Where,

$$OTF \;\; = \frac{1}{n}\sum_{i=1}^{n}\frac{NS_i}{2^n}$$

And NS_i is the number of input combinations for which a fault on input x_i can be propagated to an observable point.

The TMEAS algorithm considers the fanout effect on testability. If there is a fanout stem *S* with *K* branches, then the controllability and observability of each branch is changed to:

$$CY = \frac{CY(s)}{1 + \log k}$$

$$OY(s) = 1 - \prod_{i=1}^{k}(1 - OY(b_i))$$

This algorithm treats all fanouts regardless of them being reconvergent or not the same way. While, in testability measures, only reconvergent fanouts can cause dependency of testability metrics.

3.3 SCOAP

In the SCOAP (Sandia Controllability Observability Analysis Program) algorithm the controllability and observability metrics reflect the difficulty of controlling and observing the internal nodes. Here we have the combinational metrics and the sequential ones. The combinational controllability reflects the distance to the PIs (Primary Inputs) and the sequential controllability provides an estimate of the number of the time frames needed to provide a 0 or 1 at a particular node [15].

The corresponding metrics for a positive edge D flip-flop with an active low asynchronous reset (*R*) and clock (*C*) is defined as:

$$CC^1(Q) = CC^1(R) + CC^1(D) + CC^0(C) + CC^1(C)$$

$$CC^0(Q) = \min\{CC^0(R), CC^1(R) + CC^0(D) + CC^0(C) + CC^1(C)\}$$

$$SC^1(Q) = \{SC^1(R) + SC^1(D) + SC^0(C) + SC^1(C)\} + 1$$

$$SC^0(Q) = \min\{SC^0(R), SC^1(R) + SC^0(D) + SC^0(C) + SC^1(C)\} + 1$$

In this method reconvergent fanouts are ignored and fanouts and their reconvergence points are treated as all other circuit lines.

Several testability analysis algorithms have been derived from SCOAP. These methods are basically the same as SCOAP but with different improvement [15]. COMET [16], ITTAP [17], ARCOP [18], DTA [19], FUNTAP [20] are some of these methods.

3.4 CAMELOT

As in TMEAS, in the CAMELOT [8-9] (Computer-Aided MEasure for LOgic Testability) testability measurement method the testability metrics are controllability and observability of individual lines of a design. These metrics are between 0 and 1 and reflect the ease of controlling and observing the internal nodes. Here the controllability of an output is defined as :

$$CY(output) = CTF \times f((CY_s(inputs))$$

Where *CTF* is the Control Transfer Factor and is computed as:

$$CTF = 1 - \left|\frac{N(0) - N(1)}{N(0) + N(1)}\right|$$

where *N(0)* and *N(1)* are the number of the input combinations that set a value 0 or 1 on the output respectively.

4. RT Level Testability

Due to the large number of the components of a gate level circuit, measuring testability at the gate level is time consuming process. RT level testability solves this problem for the price of less accuracy.

4.1 General Approach

At the RT level, a common approach to testability analysis is based on the probabilities of data. Some of the existing algorithms propose measuring the combinational and sequential controlling and observing of the individual lines of a design. Thus there are 6 parameters associated with each line of a circuit [21].

4.2 Probability Based

Another approach based on probability has been proposed in Reference 22. In this approach, the controllability and the observability factors are based on the underlying state probability distributions for the registers of a circuit and the testability factors are related to the entropy of the state distribution. Here the current state of a register is the value being stored in that register and the controllability of the register is measured by the randomness of its value. For Register *x* this parameter shown by *MR(x)* is computed as:

$$MR(x) = \frac{I_x}{|x|} = \frac{1}{|x|}\sum_{i=0}^{2^{|x|}-1} P_{x,i} \log_2 \frac{1}{P_{x,i}}$$

In the above equation, *Px* is the state probability distribution vector of Register *x*, $P_{x,i}$ is the probability that Register *x* is in state *i*, and |*x*| is the bit width of the register.

In this method, the observability of Register *x* is measured by its transparency defined by *(MT(x))*. This parameter is the probability that a state error in Register *x* can be propagated to an observable point in the circuit. Thus to compute the transparency of a register, state errors must be considered. We can say that a state error occurs at Register *x* if under fault free conditions, the register is in state *i*, while in the presence of a fault it is in state *i′# i*. This parameter is calculated as follows:

$$MT(x) = \vec{t}_x \cdot \vec{P}_x$$

As shown, *MT* is the inner-product of t_x and P_x, where t_x is an state based transparency vector and $t_{x,i}$ (its i^th^ element) is the transparency of Register *x* given that its fault free state is *i*. To compute the transparency vectors of the individual registers we have to do our computing starting with the primary outputs and move towards the primary inputs. It must be considered that all the t_x elements of a primary output are 1.

The transparency of Register *x* depends on the probability of propagating a state error through a logic unit for which *x* is an input to the next register, *z*, and the probability of propagating the state error from that register to an observable point.

Both of these factors depend on the sensitivity of the combinational logic unit to its second input (y). Therefore $t_{x,i}$ elements can be computed as:

$$t_{x,i} = \left[\sum_{j=0}^{2^{1\times1}-1} S_{ij}^{\otimes} . P_{y,j} \right] \left[\sum_{j=0}^{2^{1\times1}-1} t_{z,i\otimes j} . P_{y,j} \right]$$

Here $S^{\otimes}$ is the sensitivity matrix of the logic unit ($\otimes$ is the operation performed by this unit). Thus $S_{i,j}$ is the probability that a state error at the left hand input of a logic unit from i to i' causes a state error in its output given that its second input is in state j. This method is relatively accurate but it is time and memory consuming.

4.3 Pattern Based

Another approach has been proposed in Reference 23. In this method the controllability of Register x ($C(x)$) depends on the number of the patterns that can be set on the register and the observability of Register x ($O(x)$) is computed from the number of different pattern pairs for which switching from one to the other has a different effect on the circuit output.

$$C(x) = \frac{y_1}{2^n}$$
$$O(x) = \frac{y_2}{(2^{n-1} \times (2^n - 1))}$$

In this equation, y_1 is the number of patterns, which can be set on Register x, $y2$ is the number of the different pattern pairs in Register x that have different effects on the output. Parameter n is the bit width of the register. The number of different pairs that can be set on a register is ($2^{n-1} \times (2^n-1)$).

This method considers reconvergence fanouts at the vector level. For example, if the width of the inputs of Register x is n and the number of shared bits between these inputs is p, then controllability of Register x is defined as:

$$C(x) = \frac{(C_2(x) - C_1(x)).p}{n} + C_1(x)$$

In this equation, $C1(x)$ and $C2(x)$ are controllability of Register x given that its inputs are fully independent or fully dependent on each other respectively. The number of shared bits between these two inputs is 0 or n. In this method loops are not allowed in the data path and the controller is assumed to be testable.

4.4 Dataflow Approach

Another method, proposed in Reference 24, is based on the testability analysis in terms of the dataflow and control structure extracted from an RTL design. Here the testability of a circuit is evaluated by sufficiency and smoothness of dataflow. Sufficiency is measured by the amount of data, while smoothness is evaluated by the implication cost to activate the dataflow. The RTL operations are classified into 2 classes. One class is the exclusive operation class and the other is the intersection operation class. An exclusive operation is an operation whose output is controlled only by one of its inputs and the intersection operation is an operation whose output is controlled by all its inputs. By evaluating the number of patterns a register can take as its value, the behavior of the registers is examined.

By definition, the data amount of a word is the number of bits required to express the values it can take. For example, for an n bit word, which can take only p patterns the data amount is $log_2\ (p)$.

In this method, controllability of a register depends on control data amount, control implication data amount, and control step count. Control data amount is the estimated number of patterns that the output word of a register or operation can take as its value. Control implication data amount is the sum of the word length of registers that control this register. Control step count is the ratio of control implication data amount to the sum of word length of primary inputs, which are used to feed the control implication data amount.

The observability of a register depends on the observation data amount, observation implication data amount, observation step count, and observation path activation ratio. Observation data amount is the minimum word length of the observation path through which the value of output word can be propagated to an output. Observation implication data amount is the sum of word length of registers whose values need to be determined to observe the output word of this register in an observable point. Observation step count is the ratio of the observation implication data amount to the sum of word lengths of the primary inputs that are used to feed the observation implication data amount.

Observation path activation ratio is the ratio of the data amount of an input of the observation path to the corresponding parameter at the outputs of the observation path (primary outputs). By considering these parameters the testability metrics of the individual registers of a design are computed.

4.5 Simulation Based

A proposed RT level testability analysis method is based on using simulation information to compute the controllability and observability of the individual lines of a design [25].

In this method first we apply N test vectors to the circuit and then quantify the testability parameters based on the number of times 1 or 0 appears on a line. The

One-Controllability of a module external Line l is defined as:

$$C_1(l) = \frac{ones\_count}{N}$$

N is the number of test vectors and *ones_count* is the number of times a 1 appears on Line l after applying N test vector. Zero-Controllability and observability are similarly defined.

4.6 BDD Based

A BDD based testability measurement is proposed in Reference 26, in which the BDD diagram of a design is considered and the number of required test vectors to test the design is evaluated considering the number of the nodes of this BDD. Here, the testability of a design is related to the number of the test vectors required to test the design.

If G_f is the BDD representation of Function f, then the minimum number of the test vectors required to test any implementation of f is:

$$V(G_f) = n - 1$$

In the above, n is the number of the nodes in the BDD representation of Function f. Similarly, Variable Testability Measure *(VTM)* of Line x in the RTL representation of a design is defined as the minimum number of test vectors required to test the function represented by this line. The sum of the *VTM*s at the primary outputs presents the testability of the entire circuit.

4.7 Discrete Mathematics Approach

Another approach in RTL testability is using discrete mathematics concepts for testability analysis [27]. Here the circuit elements are classified into sets according to their function in the design and their role during the test application. Other sets are defined to reflect the role of circuit registers. This method is based on *ipaths* [33, 34] and their structures. The *ipaths* are the path through which the test vectors can be transferred without being modified. *ipaths* can be classified into two groups. One is the *ipath* through an element structure and the other is the *ipath* from the output of an element to the input of another. In this method, by considering the set theory the controllable and the observable registers of the design are determined.

4.8 TAO

TAO is another approach for high-level testability analysis. In this method, by considering the algebra of regular expressions the controllable (observable) paths to (from) a node and consequently the hard-to-test points are determined [28].

5. Behavioral Testability

Improving the testability during the early stage of a design has several benefits, including improved fault coverage, reduced test hardware overhead and design iteration time. However performing testability analysis at the behavioral level involves some approximations and is even less accurate than that performed at the RT level since in this level the allocation of the structural components is not yet determined.

5.1 Probability Based

One method proposed for the behavioral level is using randomness and transparency metrics to quantify the controllability and observability of signals embedded within a behavior. This method is the same as the entropy-based algorithm, which was described in Section 4.2. The difference is that it assumes that each signal is associated with a specific register [29].

5.2 Controllability Based

Another method in this level has been proposed in Reference 30. In this method that is controllability based, variables are classified into two groups, Completely Controllable (CC) and Non-Completely Controllable (NCC). The classification is done based on the sensitivity analysis, i.e., a variable is CC if there exists a sequence of executable paths such that after the execution of these paths the content of that variable can take any possible value by adjusting the input values, otherwise the variable is NCC. In other words, if all bits of a variable are controllable the variable is Completely Controllable.

5.3 Variable Range Based

Some methods extract testability properties by analysis of Variable Range and some other parameters. Two such methods are described here.

5.3.1 Including Statement Reachability

A variable range based method proposes analysis of Variable Range, Operation Testability and Statement Reachability [31]. If a line of code puts a limit on the Value Range of a variable, testing the corresponding hardware becomes more difficult.

In this method the *VR(l,v)* reflects the variable range of value v on Line l in a behavioral design $(l \varepsilon L)$. Where L is the total number of the behavior specification. The Relative Value range for variable v at Line l is defined as:

$$RVR(l,v) = \frac{|VR(l,v)|}{|defVR(v)|}$$

Where *defVR(v)* is the defined value range of variable *v*, i.e., the range of the values it can take regardless of the behavioral specification.

The second metric, Operation Testability, reflects the change in distribution of test vectors in the output of an operation assuming all possible test vectors on its inputs. The optimum case occurs when the complete and uniformly distributed test vectors in the inputs of an operation can be transmitted uniformly and completely to its output.

In the following expression *Q(op)* reflects the difference between the distribution on the output of an operation and a uniform distribution.

$$Q(op) = \sum_{i=1}^{r} \frac{(x_i - n \times p_i)^2}{n \times p_i}$$

In the above, x_i is the number of occurrence of value *i*, *n* is the total number of outputs, p_i is the expected probability of value *i* when each *i* is assumed to occur with the same frequency and *r* is the number of possible values in the output. Then, the Operation Testability is defined as:

$$OpT(op) = \left(\frac{1}{Q(op)}\right)^{(1/b)}$$

In the above, *b* is the word-length of the inputs. By using Operation Testability, Test Hardness (*TH(l)*) and Line Operation Testability (*LOT(l)*) of Line *l* can be computed as shown below:

$$TH(l) = \sum_{op \in Op_l} (1 - OpT(op))$$

$$LOT(l) = 1 - \frac{TH(l)}{TH_{\max}}$$

The parameter Op_l is the set of operation on Line *l* and TH_{max} is the maximal Test Hardness in the original specification.

The third parameter is Statement Reachability. Some testing problems of a design are due to unreachability of its statements in the control flow. Therefore, Statement Reachability is considered here. Assuming $cs_t(c_i)$ and $cs_f(c_i)$ of condition c_i are the set of lines in the behavioral specification which will be executed if the condition c_i is true or false respectively, then Statement Reachability of Line *l* is defined as:

$$SR(l) = \prod_{c_i \in C} \begin{cases} p_t(c_i) & l \in cs_t(c_i) \\ p_f(c_i) & l \in cs_f(c_i) \\ 1 & otherwise \end{cases}$$

Where $p_t(c_i)$ and $p_f(c_i)$ are the probability of condition c_i being true or false respectively and *C* is the set of conditions in the behavioral specification.

5.3.2 Including Statement Hardness

Another method proposes analysis of Variable Range and Statement Hardness to achieve the testability of a design [32]. Statement Hardness is defined for every line of a behavioral code, and it depends on the number of the code lines that a given line of code controls. This parameter also depends on the specific instruction contained in a line of code. For example, multiplication is more complicated than addition, which makes it harder to test. Variable Range is the same as in the previous method.

In this method it is assumed that the estimated area is a function of an operation and width of its operands. This is represented by *area(op, w)* for operation *op* with word length *w*. Parameters for the individual operations are stored in a module library. By means of this library and the corresponding formulas, the testability values of the individual variables of a design are extracted.

For this evaluation, Statement Hardness of Line *l* of code (*SH(l)*) is defined as:

$$SH(l) = \sum_{op \in Op} area(op, w) \times control(l)$$

In the above, $l \in L$ and *Op* is the set of operations in Line *l* and *control(l)* is the number of the statements controlled by this instruction. The Relative Statement Hardness (*RSH(l)*) for Line *l* in a behavioral specification is:

$$RSH(l) = \frac{SH(l)}{\sum_{l \in L} \frac{SH(l)}{L}}$$

And finally the testability of the variable *v* is as shown below:

$$T(v) = \sum_{l \in L} \frac{n}{m} \times (RSH(l) + (1 - RVR(l, v) \times L)$$

In the above expression, *n* is the number of times the variable *v* occur in Line *l* and *m* is the total number of lines where variable *v* occur.

6. Applications and Comparisons

This section discusses applications of various testability analysis methods. Advantages and disadvantages of the methods at the three abstraction levels as well as complexity of the algorithms involved will be discussed.

Generally, gate level analysis methods are accurate but time consuming. Such methods are usually based on probability or the propagation strength of values from primary inputs through circuit lines. The challenge in these methods is handling reconvergent fanouts.

RT level testability analysis has been given some attention because of the large number of the components at the gate level. RT level testability methods are generally based on the gate level methods, but they are vector based. Instead of gate level components, RT level components are processed. RT level testability is less accurate than that at the gate level.

Behavioral testability measure methods are based on lines of code and behavioral variables. Since register allocation and binding are not known at this level, testability parameters are applied to variables instead of registers or lines. Performing testability analysis at this level involves some approximations and is less accurate than the RT level. This is due to the fact that allocation of the structural components is not determined at this level.

References

[1] L. H. Goldstein, SCOAP: Sandia Controllability/ Observability Analysis Program, IEEE Trans Circuits and Systems, Vol.CAS-26, No.9, pp.685-693, September 1979.

[2] L. H. Goldstein and E. L. Thigpen, Controllability/Observability Analysis of Digital Circuits, in Proc. Computer-Aided Design, pp.190-196, Minneapolis, June 1980.

[3] R. Gupta and M. A. Bruer, The Ballast Methodology for Structured Partial Scan Design, IEEE Trans Computers, Vol.39, No.4, pp.538-544, April 1990.

[4] E. Larson, An Integrated System-Level Design for Testability Methodology, PhD Dissertation No.660, Department of Computer and information Science, Linköping University, Sweden, Dec.2000.

[5] J. E. Stephenson and J. Grason, A Testability Measure for Register Transfer Level Digital Circuits, in Proc. Int. Symposium. Fault Tolerant Computing (FTCS), Pittsburgh, PA, June 1976, pp.101-107.

[6] J. Grason, TMEAS-a Testability Measurement Program, in Proc. IEEE/ACM Design Automation Conf. , 1979, pp. 156-161.

[7] F. Brglez, P. Pownall and R. Hum, Applications of Testability Analysis: From ATPG to Critical Delay Path Tracing, 1984 International Conference, pp. 705-712, September 1984.

[8] R. G. Bennetts, C. M. Maunder, and G. D. Robinson, CAMELOT: A Computer-aided Measure for Logic Testability, IEE Proc. Pt. E, vol. 128, no. 5, pp. 177-189, 1981.

[9] R. G. Bennetts, Design of Testable Logic Circuits, Addison-Wesley, Reading, MA, 1984.

[10] I. M. Ratiu, A. Sangiovanni-Vincentelli, and D. O. Pederson, VICTOR: a Fast VLSI Testability Analysis Program, in Proc. Int. Test Conf., pp. 397-401, 1982.

[11] P. G. Kovijanic, Testability Analysis, in Proc. IEEE Semiconductor Test Conf., pp.310-316, 1979.

[12] P. Kovijanic, Computer-Aided Testability Analysis, in Proc. IEEE Auto test conf., pp. 292-294, 1979.

[13] R. A. Rutman, Fault Detection Test Generation for Sequential Logic by Heuristic Tree Search, IEEE Computer Group Repository, pp.172-187,1972.

[14] P. Parikh and M. Abramovici, Testability-Based Partial Scan Analysis, Journal of Electronic Testing: Theory and Applications, No.7, pp. 61-70, 1995.

[15] C. W. Wu, VLSI Testing and Design for Testability//class notes//chapter3//Lab for Reliable Computing (LaRC), EE, NTHU, spring 2002.

[16] W. BERG, R. HESS, COMET: A Testability Analysis and Design Modification Package, IEEE International Test Conference, Philadelphia, November 1982.

[17] D. GOEL, R. Mc DERMOTT, An Interactive Testability Analysis Program: ITTAP, 19th Design Automation Conference, Las Vegas, June 1982.

[18] E. Archambeau, C. Coupal, Arcop: A Rapid Controllability Observability Program - User's manual, Rapport MATRA Design Systems, Santa Clara, August 1983.

[19] L. WAN, E. LAW, DTA: Daisy Testability Analyzer, IEEE International conference on computer aided design, November 1984.

[20] J. FONG, On Functional Controllability and Observability Analysis, IEEE International Test Conference,Philadelphia, November 1982.

[21] C. H. Chen and P. R. Menon, An Approach to Functional Level Testability Analysis, in Proc. International Test Conference, pp. 373-380, Washington, 1989.

[22] J. Carletta and C. Papachristou, Testability Analysis and Insertion for RTL Circuits Based on Pseudo random BIST, in Proc. International conf. Computer Design, pp.162-167, October 1995.

[23] M. L. Flottes, R. Pires and B. Rouzeyre, Analyzing Testability from Behavioral to RT Level, in Proc. European Design and Test Conf., pp.158.165, Paris, 1997.

[24] M. Takahashi, R. Sakurai, H. Noda and T. Kambe, A Testability Analysis Method for Register Transfer Level Description, in Proc. ASP-Design Automation Conf., pp.307-312, 1997.

[25] C. P. Ravikumar, G. S. Saud and N. Agrawal, STAFAN-like Functional Testability Measure For Register-Level Circuits, in Proc. The 4th Asian Test Symposium, pp.192-198, 1995.

[26] M. Jamoussi, A New Variable Testability Measure: A Concept for Data- Flow Testability Evaluation, in Proc. Int. Conf. VLSI Design, pp.239-244, India, 1991.

[27] Z. Kotasek and R. Ruzicka, The Implementation of RTL Testability Analysis Algorithms through the Discrete Mathematics Concepts, in Proc.4th International Scientific Conference on Electronic Computers and Informatics, pp. 177-182, 2000.

[28] S. Ravi, G. Lakshminarayana and N. K. Jha, TAO: Regular Expression Based High-Level Testability Analysis and Optimization, in Proc. Int. Test Conf., pp. 331-350, 1998.

[29] J. E. Carletta and C. A. Papachristou, Behavioral Testability Insertion for Data Path/ Controller Circuits, Journal of electronic testing: Theory and applications, Vol.11, pp.9-28, 1997.

[30] C. -H. Chen, T. Karnik and D. G. Saab, Structural and Behavioral Synthesis for Testability Techniques, IEEE Trans.CAD,13(6),pp.777-785,1994.

[31] E. Larsson and Z. Peng, Testability Analysis of Behavioral-Level VHDL Specifications, IEEE European Test Workshop, pp.120-121, Barcelona, Spain, May 1998.

[32] E. Larsson and Z. Peng, Early Prediction of Testability by Analyzing Behavioral VHDL specifications, in Proc. NOR-CHIP 97, pp 259-266, 1997.

[33] M. S. Abadir and M. A. Bruer, A Knowledge Based System for Designing Testable VLSI Chips, IEEE Design & Test of Computers, Vol.2, No.4, pp.56-68, 1985.

[34] M. S. Abadir and M. A. Bruer, Constructing Optimal Test Schedules for VLSI circuits Having Built-in Test Hardware, in Proc. Int. Fault-Tolerant Computing Conf., pp.165-170, 1985.