TESTABILITY ANALYSIS DRIVEN DATA PATH MODIFICATION AND CONTROLLER SYNTHESIS

Josef Strnadel, Richard Růžička

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 66 Brno, Czech Republic
phone: +420 5 4114 1211, +420 5 4114 1208, fax: +420 5 4114 1270
e-mail: strnadel@fit.vutbr.cz, ruzicka@fit.vutbr.cz

ABSTRACT:

In the paper, it is shown how testability analysis can be used both to modify digital data path for maximize testability at minimal costs and to offer information applicable during automated synthesis of a controller used to apply a test to the modified data path.

Keywords: analysis, digital, circuit, controller, synthesis, testability

1 INTRODUCTION

Many *testability analysis* (TA) approaches have been developed in the past. They can be classified according to several aspects. On the basis of abstraction level, they can be divided, e.g., to gate-level [4][8], *register-transfer level*, RTL, [3][5][10][13], *functional level* [2][16], *behavioral level* [12] or *multilevel* [6] approaches. According applicability of results, they can be divided to *general-purpose*, e.g., [2][4][12][13] and *special-purpose* (e.g., with results strongly tied to application of particular DFT technique as partial scan in [10]). According to *data path* (DP) analysis utilized, they can be divided to simpler *probability-based* (e.g., [3][5][8]) dealing with stochastic behavior of data flow, and exact *structural-analysis* based approaches (e.g., [4][10][13]). In most of the approaches, testability is evaluated by means of *controllability* (C) and *observability* (O) parameters; but, the approaches differ in the way, how C and O are defined and measured. TA method used in the paper can be classified as multilevel, general-purpose, structural-based TA approach with library-driven behavior and accuracy. During our previous research activities, we tried to take advantage of so called *transparency principles* utilized for enhancement of hierarchical TPG methods and to utilize in TA area [13] with several application areas like DFT area and synthetic benchmark-generation area, all over the class of RTL digital circuits [9][13]. In [11], principle of a test controller synthesis method on basis of a special grammar was presented.

During our last activities, we have decided to connect two important parts (DP modification, test controller synthesis) developed separately into one complex methodology. Because significant integer of our research and development work is completed now, we have decided to present a view of a framework in the paper as well as to make a comparison with similar approaches.

The structure of the paper is as follows. First, transparency principles are presented. Next, language used to describe transparency information for each component is presented together with principle of automated extraction of the information. After that, TA principle is presented in brief together with applicability of its results in DP modification and test controller synthesis areas. At the end of the paper, brief summary is presented together with possible future research perspectives.

2 Key-Concepts Related to Our Approach

2.1 Introduction to Transparency Conceptions

A lot of research efforts have been dedicated to the importance of modeling data flow in digital DP in order to estimate diagnostic parameters of the circuit more precisely using *I/T-Path* model [1] or its derivative: as an example to one-to-one *i-paths* over equal-bit-width ports, see Fig. 1a) – i-path from port *a* to port *y* of multiplexer MX exists iff selection input *sel* is set to 0. Alike in Fig. 1b) – i-path from *d* to *y* of register R exists iff a rising edge appears on *clk*. In Fig. 1c) i-path from *a* to *y* of adder ADD exists iff *b* is set to all 0's.

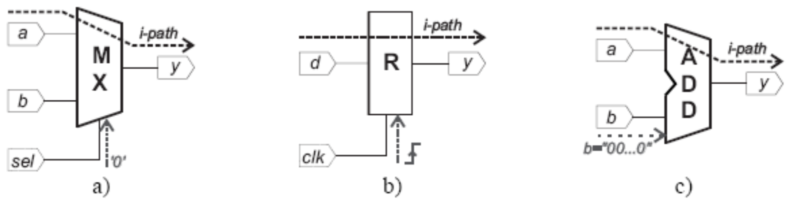

Fig. 1. Illustration to i-paths

Other works tried to enhance properties of above-mentioned model. E.g., in conception referred to as *S/F-Path* conception [3], it was shown I/T-Path's strict "bijective mapping requirement" can be soften by *analyzing DP separately* for transferring test vectors/patterns (responses) between *x* and *y* ports: *test vectors/patterns* (*responses*) can be transferred from *x* to *y* iff a *surjective* (*injective*) mapping exists between *x*-data and *y*-data. Using this less-strict principle, much greater DP portion can be considered suitable for diagnostic data flow comparing to I/T-Path conception. Several variations of the approaches have been used in the area of generating so-called *hierarchical tests* [5, 6, 7, 11]. All of the approaches are often referred to as *transparency conceptions*, because they deal with modeling of situations, in which DP portion is "transparent" to transported data.

2.2 Transparency Extraction Method

Suppose a function of each component is described by means of a truth table. For general case of *k*-bit output functions ($k \geq 1$), we have developed a transparency extraction method which takes a set $F = \{f_{k-1},\dots,f_0\}$ of *k* *n*-input Boolean functions f_i: $\{0,1\}^n \rightarrow \{0,1\}$, $i = k-1,\dots, 0$ as its input (function f_i is related to i^{th} output bit) and generates transparency information at its output.

Tab. 1: FA truth-table

FA inputs			FA outputs	
x	y	c_{in}	z	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tab. 2: Surjective/Injective mappings found for FA

Member of	In	Out	Test	Inner Encoding	Information stored in the library
S_f	xyc_{in}	zc_{out}	-	(111,11,0)	$x(0)y(0)c_{in}(0)\|z(0)c_{out}(0)\|$ -
S_f, I_f	x	z	yc_{in}	(100,10,011)	$x(0)\|z(0)\|y(0)c_{in}(0)$
S_f, I_f	y	z	xc_{in}	(010,10,101)	$y(0)\|z(0)\|x(0)c_{in}(0)$
S_f, I_f	c_{in}	z	xy	(001,10,110)	$c_{in}(0)\|z(0)\|x(0)y(0)$
S_f, I_f	c_{in}	c_{out}	xy	(001,01,110)	$c_{in}(0)\|c_{out}(0)\|x(0)y(0)$
S_f, I_f	y	c_{out}	xc_{in}	(010,01,101)	$y(0)\|c_{out}(0)\|xc_{in}(0)$
S_f, I_f	x	c_{out}	yc_{in}	(100,01,011)	$x(0)\|c_{out}(0)\|y(0)c_{in}(0)$

For illustration, let us present in Tab. II all information about surjections (S_f set containing 7 mappings) and injections (I_f set contains 6 mappings) sets for FA in both binary (Tab. 1) and textual forms (Tab. 2). The extraction method is based on so-called *horizontal line test* [15].

2.3 Library Language

Let us illustrate now how the extracted information that will be stored in a textual form (to be processed, e.g., by TASTE [14]) could look like:

```
MODULE_TYPE FA1
INTERFACE in@x(0) in@y(0) in@cin(0) out@z(0) out@cout(0)
SUR x(0)y(0)cin(0)|z(0)cout(0)|-
INJ
BIJ x(0)|z(0)|y(0)cin(0) y(0)|z(0)|x(0)cin(0) cin(0)|z(0)|x(0)y(0)
    cin(0)|cout(0)|x(0)y(0) y(0)|cout(0)|x(0)cin(0)
    x(0)|cout(0)|y(0)cin(0)
```

Except information related to one particular component type only (1-bit FA in our case), it is possible put more general information (covering entire class of circuits types – e.g. *n*-bit registers, adders, multiplexers) in the library.

2.4 TA Principle

After transparency-related information is stored in the library for each of particular component type or class of component types, it can be assigned to particular components design consists of. Structure of particular design can be described by means of a net-list. Afterwards, it is possible to construct two special digraphs for the DP:

- *test pattern data-flow digraph* $G_S = (V_S, E_S)$ and
- *test response data-flow digraph* $G_I = (V_I, E_I)$.

Set of vertices of G_S (G_I) consists of ports. An oriented edge exists between two vertices iff surjection (injection) exists between the start-vertex and end-vertex data (i.e., iff it is possible to transfer test vectors (responses) from start-vertex to end-vertex). Also, each edge in G_S (G_I) is evaluated by a "transfer-condition" function μ_E: $E \rightarrow 2^V$, where $E = E_S \cup E_I$, $V = V_S \cup V_I$. Using μ_E, set of ports necessary to control an edge $e \in E$ is assigned to the edge.

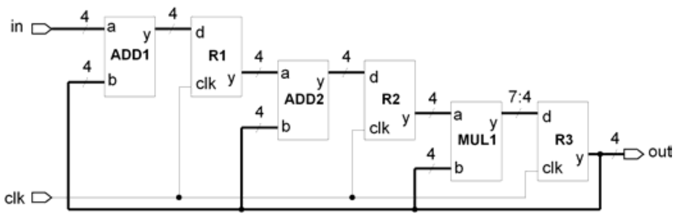

Fig. 2. NL circuit

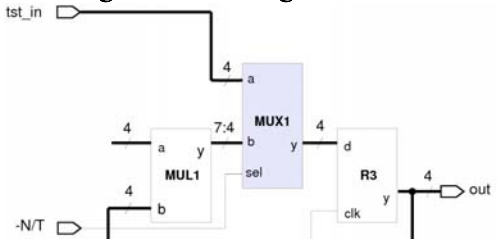

Fig. 3. Modification of NL circuit

For illustration to the graphs, let us suppose now NL circuit (see Fig. 2) is modified as depicted in Fig. 3 (multiplexer MUX1 is added to DP between MUL1.y(7:4) and R3.d in order to enhance testability of NL by breaking the most-nested loop). In Fig. 4, portion of G_S (Fig. 4a) for adjusting test data from PI *tst_in* to input *b* of *ADD1* (*ADD1.b*) is presented together with portion of G_I (Fig. 4b) for observing test data from output *y* of *MUL1* (*MUL1.y*) at PO out. In Fig. 4, following graphical notation is used. In full-line circles, ports of in-circuit components are depicted, in dash-line circles, PIs/POs are depicted and in a double-line circle, port the digraph portion belongs to is depicted. Circles connected by a full-line represent test path for the double-lined port and circles connected by dash-line represent paths to be controlled in order to ensure the data flow through full-line path. Each edge is evaluated by means of μ_E. For more information about TA principle, see e.g. [13] or [14].

2.5 Test Controller Synthesis

Above-presented TA method can be utilized to detect and localize, which parts of the circuit DP cause low testability of the circuit. Consequently, the information can be used by

DFT/SFT process taking TA result as a measure of DP quality from diagnostic point of view in order to modify DP in such a way testability is maximized at minimal costs. After DP is modified, it is necessary to appropriately modify original circuit controller about test controller part. Below, it will be shown how TA results can be utilized for automated construction of the test controller. First research in this area we have started in [11].

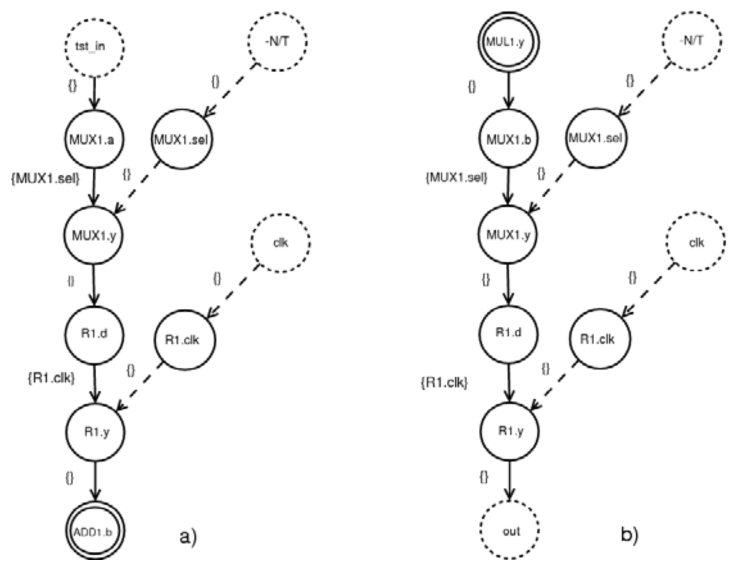

Fig. 4. Illustration to G_S

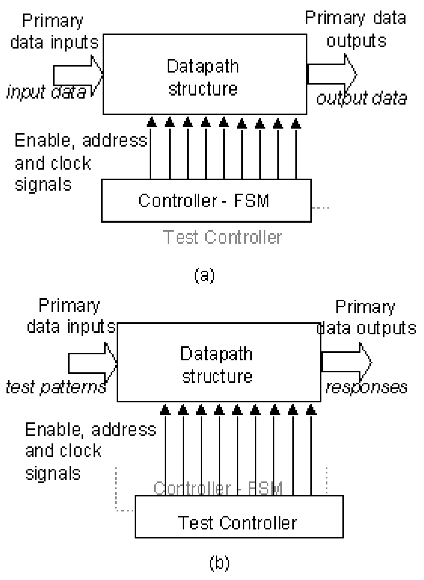

Fig. 5. Datapath/controller in (a) normal and (b) test modes of operation

The controller can be described as a *finite state machine* (FSM) controlling data flow by consecutive enabling of DP portions. In a test mode, original circuit controller is disabled and tested DP is controlled by the test controller. Its task is to enable DPs in order to control flow of test patterns and corresponding responses to particular DP parts. For each path within the DP a FSM, generating all required enable, address and clock signals in right succession must exist. The right succession depends on the trace of the path (on succession of elements and their ports on *control* (*pc*) or *observe* (*po*) paths related to particular in-circuit node *p*). It is evident that the FSM which enables one particular path is a sub-FSM of the test controller. The arrangement of sub-FSMs in the test controller is the object of test scheduling discipline.

A test controller can be formally described by means of FSM. Because of equivalence of FSMs and regular grammars, it is possible to create a grammar for a language, describing a path portion. On basis of the grammar, corresponding FSM can be generated. It is possible to construct automaton M, where $L(M) = L(H)$ and H is a grammar generating a language of all paths in the circuit.

Let us imagine a language L, which describes all possible diagnostic DPs in the circuit structure. As an alphabet of L, set of all ports of the circuit can be used. Each string is a sequence of ports on the path. It is evident that it is regular language – the set of all paths is finite. It can also be seen that L is a prefix language. Such a language can be very easily created from test-vector/response transportation sets S_c/S_o:

$$x_1x_2\dots x_n \in L \leftrightarrow (x_1,x_2,\dots x_n) \in S_c \vee (x_1,x_2,\dots x_n) \in S_o$$

Let us have a grammar $H = (N, T, R, B)$, generating L. The meaning of symbols from the 4-tuple is as follows:

- $N = \{B\} \cup \{<a,X> \mid a \in T, X \subseteq T\}$ is the set of *nonterminals* (each of them is a pair of a port and a port set. The meaning of the set X is a trace of the path, which leads to the port a. X helps to eliminate cycles in the grammar due to feedbacks in the circuit. If these cycles were not eliminated, language would be infinite),
- $T = \{PI \cup PO \cup V\}$ is the set of *terminals*,
- $R = R_1 \cup R_2 \cup R_3$ is the set of *production rules*, where

- $R_1 = \{B \rightarrow a < a, \{a\}> \mid\ < a, \{a\}> \in N \wedge a \in PI \cup V\}$ includes starting rules. Because each path can start at some in-circuit port or PI, each derivation can start from such a port,
- $R_2 = \{<b, X> \rightarrow\ \varepsilon \mid\ X \subseteq T\ \wedge b \in PI \cup V\}$ represents ending rules. Because each path can end at some in-circuit port or PO, each path can end at such a port.
- $R_3 = \{<a, X> \rightarrow b <b,\ X \cup b> \mid X \subseteq V \wedge a \in T \wedge b \in T\ \wedge\ (a\ , b) \in E \wedge b \notin X\}$ represents all other rules: the rule in form of $<a, X> \rightarrow b <b,\ X \cup b>$ will be added to the rule-set iff two main conditions will be fulfilled: 1. $(a\ , b) \in E$, i.e. it must exist an edge from the port a to the port b in the graph G and 2. $b \notin X$, i.e. the path, which leads to the port b must never pass through the port b anytime before. The second condition assures that no loop occur in the grammar. Thus, the language described by grammar will be finite.

- $B \in N$ is a *start symbol.*

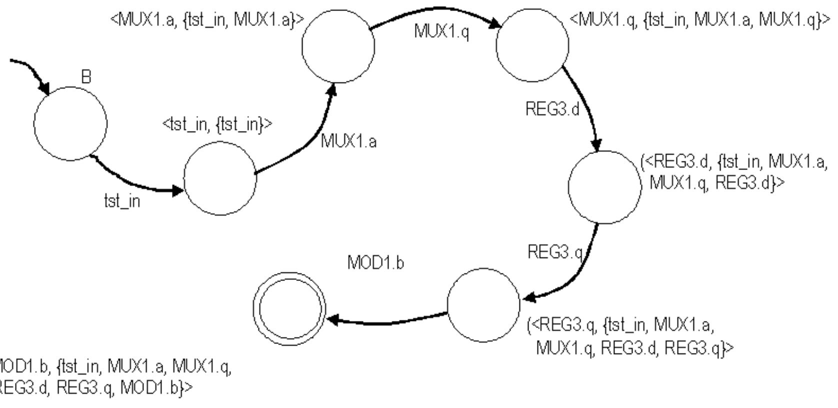
B
tst_in
<tst_in, {tst_in}>
MUX1.a
<MUX1.a, {tst_in, MUX1.a}>
MUX1.q
<MUX1.q, {tst_in, MUX1.a, MUX1.q}>
REG3.d
(<REG3.d, {tst_in, MUX1.a, MUX1.q, REG3.d}>
REG3.q
(<REG3.q, {tst_in, MUX1.a, MUX1.q, REG3.d, REG3.q}>
MOD1.b
<MOD1.b, {tst_in, MUX1.a, MUX1.q, REG3.d, REG3.q, MOD1.b}>

Fig. 6. Sub-FSM example for test-pattern part of Fig. 4a

In Fig. 6, a sub-FSM diagram corresponding to test pattern flow digraph depicted in Fig. 4a is shown. The path begins at primary input tst_in and ends at the input b of MOD1 module. The language describing such one particular path then consists of one string only: $L_{MOD1.b}$ = {tst_in MUX1.a MUX1.q REG3.d REG3.q MOD1.b}. $H_{MOD1.b}$ is is defined as follows:

- N = {B, <tst_in, {tst_in}>, <MUX1.a, {tst_in, MUX1.a}>, <MUX1.q, {tst_in, MUX1.a, MUX1.q}>, …, <MOD1.b, {tst_in, MUX1.a, MUX1.q, REG3.d, REG3.q, MOD1.b}>}
- T= {tst_in, MUX1.a, -N/T, MUX1.sel, MUX1.q, REG3.d, clk, REG3.clk, REG3.q, MOD1.b}
- R = {B → tst_in <tst_in, {tst_in}>, <tst_in, {tst_in}> → MUX1.a <MUX1.a, {tst_in, MUX1.a}>, …, <MOD1.b, {tst_in, MUX1.a, MUX1.q, REG3.d, REG3.q, MOD1.b}> → ε }

The automaton created by means of above-mentioned method can be enriched about information needed to enable paths through circuit elements – there is not only need to trace a path but mainly to enable it. Thus, the FSM must be extended to a Mealy automaton: for the automaton M, where $L(M) = L(H)$ is a language describing a path, Mealy automaton $M' = (Q, T, 2^{V \times D}, \gamma, \lambda, B, F)$ can be created with the output function $\lambda = \{\ (s, b, \{(cp, x), ...\}) \mid s \in Q \wedge s=(<a, X>, b, <b,\ X \cup b>) \wedge a, b \in V \wedge \mu_E(a, b)=\{(cp, x), ...\}\ \}$.

It can be seen that the output alphabet of the Mealy automaton is $2^{V \times D}$. It means that output of the automaton are pairs like (cp, x), where $cp \in V$ is a port to which a value $x \in D$ must be assigned in order to enable a path through some edges of G_S (G_I).

For example, if the FSM depicted in Fig. 6 will be extended to the Mealy automaton, the output function will be *λ = {(B, tst_in, -), (<tst_in, {tst_in}>, MUX1.a, -), (<MUX1.a, {tst_in, MUX1.a}>, MUX1.q, {(MUX1.sel,"1")}), (<MUX1.q, {tst_in, MUX1.a, MUX1.q}>, REG3.d, -), (<REG3.d, {tst_in, MUX1.a, MUX1.q, REG3.d}>, REG3.q, {(REG3.clk, "↑")}), (<REG3.q, {tst_in, MUX1.a, MUX1.q, REG3.d, REG3.q}>, MOD1.b, -)}.*

3 CONCLUSION

In the paper, principle of utilizing TA results for automated data path modification (resulting in higher testability) as well as automated generation of corresponding synthesizable test controller were presented. There are two inputs TA requires: a net-list and a library of components (affecting TA accuracy). Also, principle of extracting information stored in the library was presented in the paper – so, significant integer of our research and development work could be introduced. Our further research will be dedicated especially to TA and S/DFT of hierarchical and system-on-a-chip digital and mixed-signal designs, which belong to the most popular approaches at present. Also, further experiments and comparisons are planned.

ACKNOWLEDGMENTS

The work related to the paper has been financially supported by the Research Plan No. MSM, 0021630528 – *Security-Oriented Research in Information Technology*.

REFERENCES

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

[2] C. H., Chen, P. R., Menon: An Approach to Functional Level Testability Analysis. In: Proceedings of the International Test Conference, 1999, pp. 373—380.

[3] J., Fernandes, M. B. Dos Santos, A. Oliveira, J. P. Teixeira and R. Velazco: Sensitivity to SEUs Evaluation using Probabilistic Testability Analysis at RTL, In: Proceedings of 8th LATW, 2007.

[4] L. H. Goldstein and E. L. Thigpen. SCOAP: Sandia controllability/observability analysis program. In DAC '80: Proceedings of the 17th conference on Design automation, pp. 190–196. ACM Press, 1980.

[5] V. I. Hahanov et al.: *Testability Analysis of the VHDL Structure for Fault Coverage Improving, Electronics and Electrical Engineering*, Vol. 74, No. 2, 2007, pp. 29-32

[6] J., Hlavička et al.: Interactive Tool for Behavioral Level Testability Analysis, In: Proceedings of the IEEE ETW 2001, Stockholm, 2001, pp. 117-119.

[7] D., H., Lee, S., M., Reddy: On Determining Scan Flip-Flops in Partial Scan Designs. In: Proc. Of Int. Conf. on Computer Aided Design, 1990, pp. 322 – 325.

[8] C. M., Maunder, R. G., Bennetts and G. D., Robinson: CAMELOT: A Computer-Aided Measure for Logic Testability. In Proc. of Intenational Conference on Computer Communication, 1980. pp. 1162–1165.

[9] T., Pečenka: Tools and Methods for Automated Generating of Benchmark Circuits. PhD thesis, Brno University of Technology, 2007. 124 p.

[10] R., Růžička: Formal approach to the Testability Analysis of RT Level Digital Circuits. PhD thesis, Brno University of Technology, 2002, 102 p.

[11] R., Růžicka, P., Tupec: Formal Approach to Synthesis of a Test Controller. In: Proc. of the ECBS Conference and Workshop, Brno, IEEE CS, 2004, pp. 348-355.

[12] S. Seshadri and M. Hsiao. *Behavioral-level dft via formal operator testability measures*. JET Vol. 18, No. 6, pp. 596 – 611, 2002.

[13] J., Strnadel: *Testability Analysis and Improvements of Register-Transfer Level Digital Circuits*, Computing and Informatics, Vol. 25, No. 5, Bratislava, 2006, pp. 441-464.

[14] J., Strnadel: TASTE – Testability Analysis SuiTE. Available on-line at http://www.fit.vutbr.cz/~strnadel/diag/taste

[15] J., Strnadel: On Encoding and Utilization of Diagnostic Information Extracted from Design-Data for Testability Analysis Purposes, In: Proc. of 6th ECS Conference, Bratislava, 2007, pp. 333-338.

[16] V., Vishakantaiah, J. A., Abraham and M. S., Abadir: Automatic Test Knowledge Extraction From VHDL (ATKET), In: Proceedings of DAC, 1992, pp. 273-278.