VLSI Design, 2002 Vol. 15 (2), pp. 491–498

Taylor & Francis
Taylor & Francis Group

Improving Datapath Testability by Modifying Controller Specification

M.L. FLOTTES*, B. ROUZEYRE and L. VOLPE

Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier, U.M.R. 5506 CNRS/Université de Montpellier 2, 161 rue Ada, 34392 Montpellier Cedex 5, France

(Received 15 November 2000; Revised 28 March 2001)

A digital circuit includes two main parts: a controller and a datapath. After connection of these two parts, both are subject to a sharp fall in testability due to the lack of controllability and observability at the interface. In this paper, we propose a method for specifying the control part in order to restore the testability of the datapath to a level close to the initial one, in other words its testability before connection. This testability driven specification affects the next state logic as well as the decoder part of the controller but does not make use of any scan-based element. Based on the finite automata theory and on results of a testability analysis performed on the datapath, the proposed method entails very little area penalty.

Keywords: Datapath; Testability; Test control; DFT; FSM; Controller specification

INTRODUCTION

Most of digital circuits are composed of a datapath and of a controller (Fig. 1). This dichotomy is particularly apparent when the circuit is obtained through a High-Level Synthesis flow. The datapath performs computation on data applied on its primary inputs. The controller sequences the normal flow of execution of the datapath. Even if the datapath is fully testable when considered in isolation, particularly if generated using a High Level Synthesis for Testability tool (see Refs. [1,2] for a survey), its testability can be strongly affected after connection to the control part.

In fact, the controller implements only the normal flow of execution. As a consequence, (1) the actual set of control words and (2) the word sequencing in system mode may limit the possibilities of faults testing in the datapath.

The first point is shown in Fig. 2. Let us assume that in system mode, only the multiplications $PI_1 \times Cst2$ and $PI_2 \times Cst1$ are exercised. In order to achieve full controllability on the output of the multiplier, it may be necessary to perform the operation $PI_1 \times PI_2$. Unfortunately the decoder is not specified to generate a control word containing (0, 1) on (sel_mux1, sel_mux2).

The limitations due to the sequencing are shown with the help of the datapath in Fig. 3a. Let us consider the faults in the adder. One way of testing these faults is given in Fig. 3b (load R1, then write data on output PO). Unfortunately, in system mode, the adder response is never observed directly on the primary output. Adder results are first shifted then subtracted with data stored in R3 (Fig. 3c). Consequently, some faults cannot be tested due to the presence of other operations in the propagation path. In this example, the required "words" for testing the adder exist in the controller, but their sequencing does not allow the observation of some adder faults.

One solution for solving the problem of datapath test control is to insert a scan chain at the datapath/controller interface. Such scan chain provides full controllability of the datapath in such a way that any control sequence can be used during the test mode. However, the drawbacks of this approach are well known: the need of test data serialization, the area overhead, the extra delays between controller and datapath.

The methods proposed in Refs. [3,4] tackle each of the above-mentioned problems (lack of control words and limitations due to the sequencing) but not both. Conversely, the technique presented in Ref. [5] targets both problems. It is based on high-level testability results [6] and consists in re-synthesizing the controller. This method may entail some datapath modifications.

*Corresponding author.

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/1065514021000012101

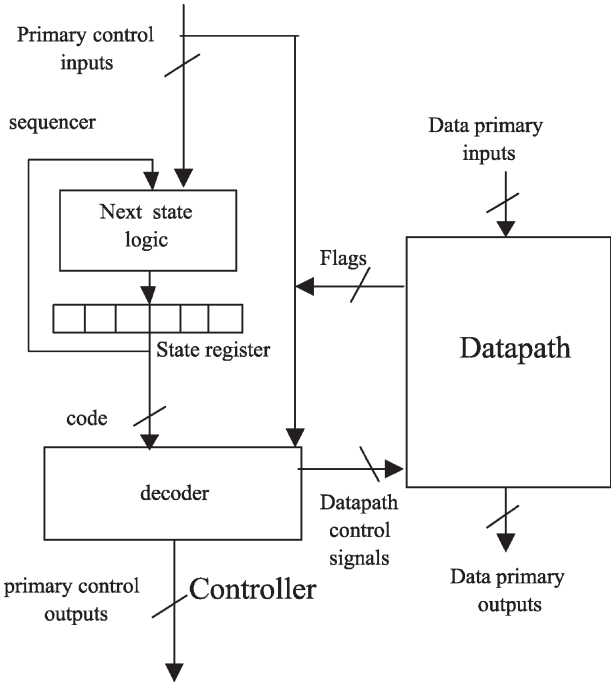

FIGURE 1 Circuit's architecture.

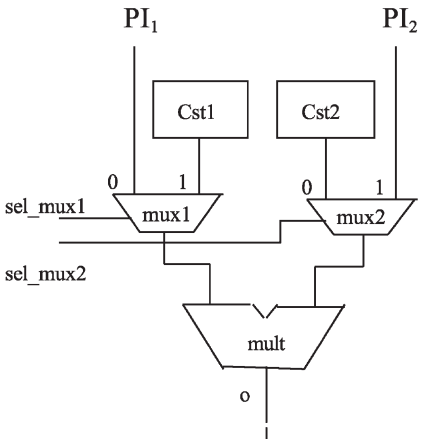

FIGURE 2 Fragment of a datapath.

PRINCIPLE

We propose here to modify the controller specification for enhancing the testability of the datapath. Neither the testability of the controller itself nor the observation of datapath faults through the controller is addressed in this paper. The method consists of adding new control words and new transitions to the controller specification in order to maintain the testability of the datapath as high as possible.

The structural modifications of the controller are shown in Fig. 4: they concern the next state logic, the decoder and the introduction of new primary control signals for activating the added transitions. New control words and transitions are intelligently chosen and added to the controller while minimizing the area overhead.

First, a register transfer (RT) level testability analysis [7] is performed on the datapath. This analysis delivers a "test path" for each datapath module, i.e. a path along which as many patterns as possible can be propagated to test this module. Test data must be propagated through this path from the primary data inputs to the module under test, and from the module to the primary data outputs. Activation of a path may involve the use of non-existent control words or control sequences because during analysis, the testability of the datapath is examined without taking into account the controller. On the other hand, it must be noted that several test paths may exist for a given module but a single one is returned by the analysis step. This point will be further discussed in the "Discussion" section. A test path returned by the testability analysis can be scheduled in different ways. Scheduling of such a path gives a sequence of instructions (control words) for testing the module. The second step of our approach consists in including such sequences in

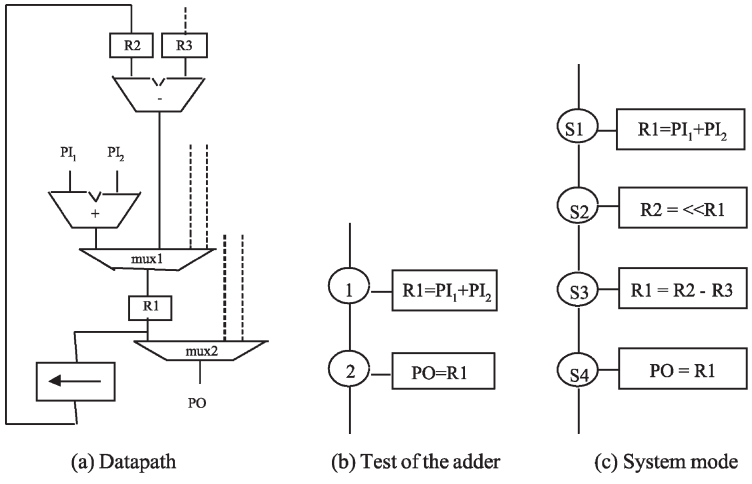

FIGURE 3 Test limitations due to sequencing.

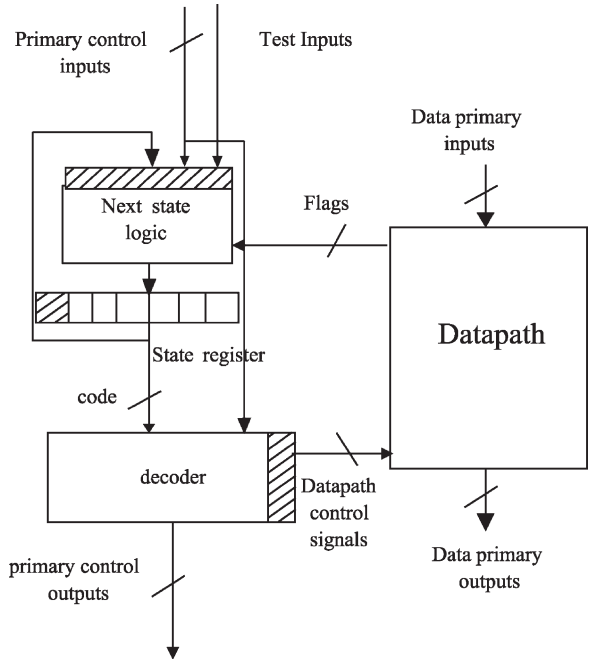

FIGURE 4 Circuit's architecture after modification.

the controller specification. This inclusion is done while keeping in mind the cost related to extra instructions, transitions and test pins.

PROBLEM STATEMENT

For each module, the test path can be scheduled in many different ways (an infinity in fact), giving rise to several sequences of instructions, each being a candidate for inclusion in the controller with a related area overhead. Such a sequence of instructions is called a *test plan* in the remainder.

For instance, let us consider the test of the register R3 in the datapath shown in Fig. 5a. The initial control is shown in Fig. 5b. The R3 test path, depicted by bold lines in Fig. 5a, can be scheduled in at least three ways as given in Fig. 5c–e. The first two require the addition of 1 state, 2 transitions and 2 control words for their inclusion in the controller specification. The third one can be activated without any modification.

Thus, the problem is to choose for all modules the right test plan in such a way that the overall area overhead is minimized. The method we developed is detailed in the next section. It exploits the similarities between (1) the control words and the transitions to add and (2) the ones that already exist in the initial controller. It also relies on the possibility of fixing unspecified values on some control bits (do not care) for improving testability. The method is based on automata theory [9].

METHOD

Modeling

Before selecting the best test plan for a module, we are first faced with the problem of modeling all the schedules of its test path. Rather than explicitly enumerate all the schedules, we represent them implicitly with a help of a finite automaton. Let us recall that a finite automaton is a 5-uple (A, Q, I, T, E) in which A is an alphabet, Q is a finite set of states, $I \subset Q$ a set of initial states, $T \subset Q$ a set of terminal states and E a set of edges. An automaton allows recognizing a language L and a language allows recognizing some words, i.e. some sequences of letters from A. In our context, A is the set of control words (existing in the initial controller or not). A* represents any sequence of any value.

Formally, the finite automaton for a test path is constructed following the data stream from the inputs to the outputs. For instance, let us consider the example shown in Fig. 6 in which the test of the register R3 is addressed. As in the example of Fig. 3, R3 cannot be fully tested in the system mode because of the shift operation that is planed after R3 loading. Thus, the testability analysis derives the test path depicted in bold lines in the datapath. We build up the corresponding automaton in the following way. First, two elementary finite automata (P1, P2), (Q1, Q2) are constructed for the registers connected to the inputs as shown in Fig. 7. Then, the automaton corresponding to the different ways of loading R1 and R2 is built up as the cross product* of the two previous automata. It is represented by the graph fragment (P1Q1, P2Q1, P1Q2, P2Q2) in Fig. 8. Finally, this automaton is completed by the automaton corresponding to the R3 (state T1) and R5 loading (state T2). With this formalism, any valid test plan (i.e. one schedule of a test path) corresponds to a graph traversal from the initial state (P1Q1) to the terminal one (T2). For instance, the path P1Q1, P1Q1, P2Q1, P2Q1, P2Q2, T1, T2, T2 corresponds to the following test plan:

- do anything
- load R1 from I1
- do anything any number of times but loading R1
- load R2 from I2
- load R3, while selecting input 1 of the mux
- load R5
- do anything any number of times but loading R5.

In this model, the infinite number of test plans is represented in a compact way. The initial controller can be modeled in the same way.

Principle

The principle of the proposed approach consists in including a test plan for every module in the controller

*A modified version of the cross product is used in order to avoid test data reconvergence. For instance, if R1 and R2 are connected to the same input, the transition from P1Q1 to P2Q2 (Load R1 and R2 at the same time) is not generated.

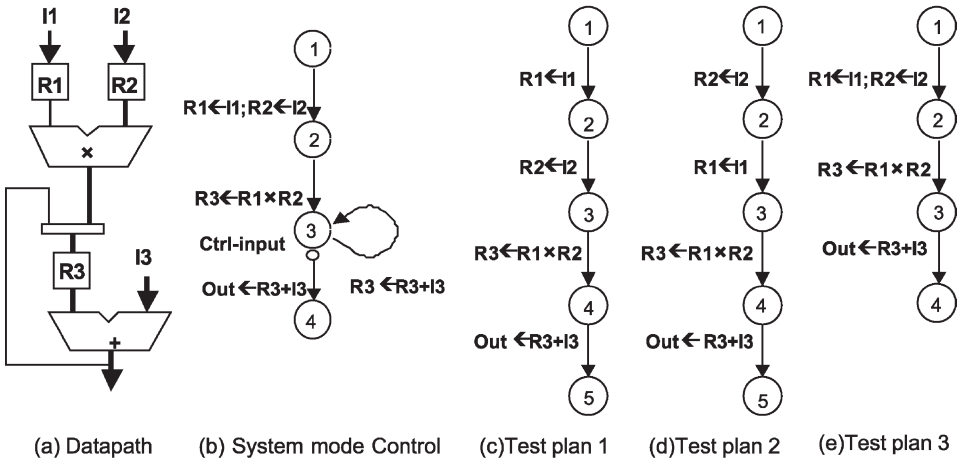

FIGURE 5 Test plans.

specification. This inclusion does not induce additional cost when the initial controller already implements a test plan for the module under consideration. Otherwise, we have to add new control words and/or transitions. The strategy consists in selecting the less expensive test plan, i.e. the one that is the closest to the initial controller.

For instance, in the previous example, the test plan P1Q1, P2Q2, T1, T2 is selected for the test of register R3. This test plan is simply embedded in the initial controller specification by modifying the control word from state S4 to S5 (Fig. 9). From a bit representation point of view, the initial do not care value assigned to the R5w signal is replaced by a 1. This modification may impact the decoder logic.

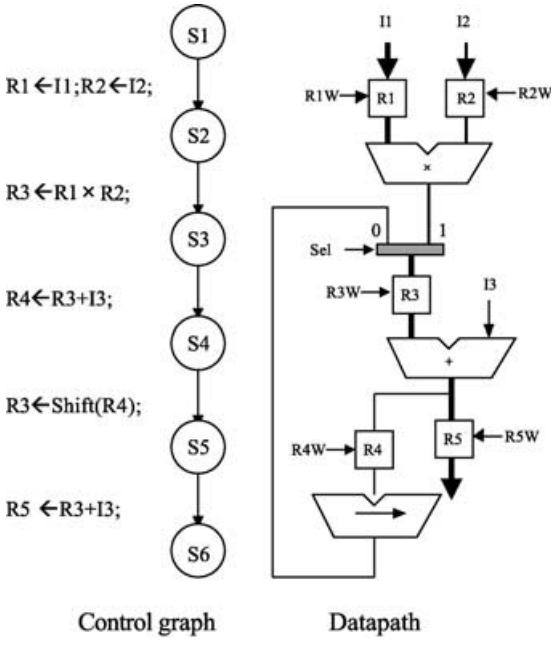

FIGURE 6 Illustrative example.

Implementation

The first goal of the method is to check whether the controller already contains one test plan of the datapath module under consideration. If not, the method includes a test plan in the controller with a minimal cost.

The test plan selection is done according to a cost function taking into account (1) the number of added transitions, (2) the number of added control words, (3) the number of added test pins, (4) the number of do not care values fixed and (5) the number of modules this test plan can be used for.

Formally the algorithm for test plan selection is the following:

1. MinCost $= +\infty$
2. For each module M
3. For each test plan P of M
4. Word = FirstWord(P)//Word represents the label attached to the transition
5. Cost = 0;
6. (Cost, ModifiedController) = BestImplementation (Controller, P, Word, MinCost, Cost);
7. If Cost $<$ MinCost then
8. MinCost = Cost;
9. Pmin = P;
10. Return(Pmin, ModifiedController).

M is a module under test, P of M is a test plan extracted from the M test path and P_{min} is the minimal cost test plan.

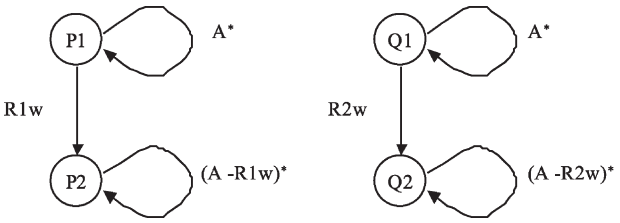

FIGURE 7 Automata for R1 and R2 loading.

BestImplementation returns the lower cost solution for implementing P in the current controller specification (Controller).

In which the BestImplementation is the following:

1. If P is already implemented in Controller
2. If Cost $<$ MinCost then
3. MinCost = Cost
4. ModifiedController = Controller;
5. If there is no transition in Controller compatible with Word then
6. CreatState(Word);
7. For each transition t of Controllercompatible with Word
8. Cost + = CostAnd Update (t, Word, P, Controller);
9. If Cost $>$ MinCost
10. then Cost − = CostAndUpdate (t, Word, P, Controller);//Pruning and backtrack
11. else BestImplementation (Controller, P, Next(P, Word), MinCost, Cost);
12. Cost − = CostAndUpdate (t, Word, P, Controller)//backtrack

Return (Cost, Controller).

For instance, let us consider the example in Fig. 10. Among all test plans, the path depicted in bold p1, p4, p5, p6 is selected. Transition p1−p4 is mapped onto transition s1−s2 on the initial controller; transition p4−p5 is mapped on transition s2−s3. None of the transition p5−p5 or p5−p6 of the test plan can be mapped on a transition issuing from s2. One solution is to add a transition from s3 to s6, this extra transition will implement the test mode transition p5−p6.

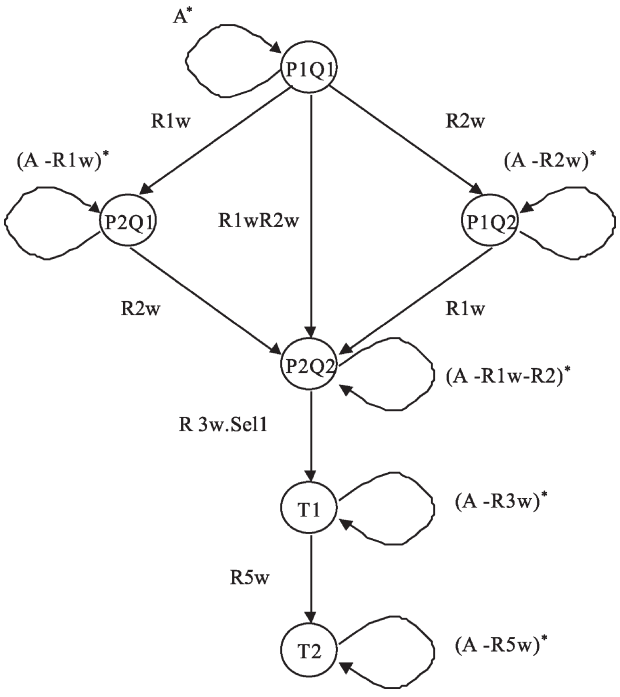

FIGURE 8 Finite automata of test path for R3 in Fig. 6.

RESULTS

This method has been applied to five HLS benchmarks circuits. The RTL descriptions have been obtained with our HLS tool. Table I gives the fault coverage (FC), test efficiency (TE) and ATPG CPU time (CPU) for the datapath considered in isolation, the datapath connected to the initial controller and the datapath with the modified controller. These results have been obtained using the Synopsys suite. They show that the testability level have been raised very close to the maximum.

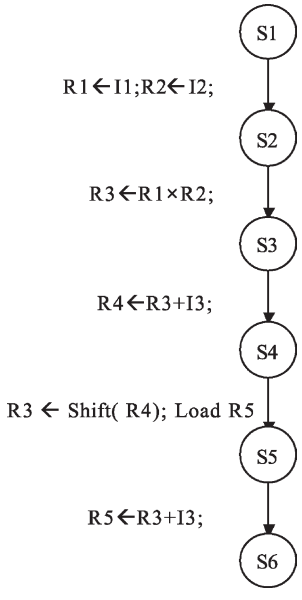

FIGURE 9 Modified controller.

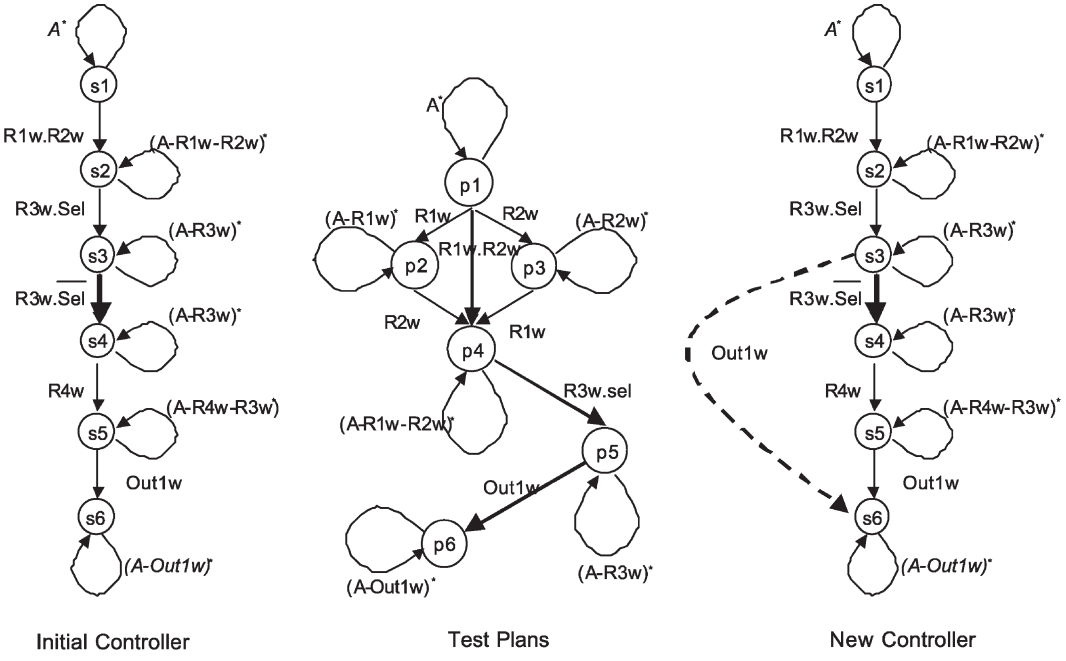

FIGURE 10 Test plan mapping.

Another way of solving the problem is to use scan chains in the controller either on the controller outputs (Scan-dec) or the next state flip-flops (Scan-sec). We compared these strategies with ours. For each strategy, Table II reports the fault coverage (FC), the test efficiency (TE), the area overhead (Over) and the test application time (in number of clock cycles). These numbers shows that the proposed technique compares favorably with scan insertion techniques with the advantage of shorter test time.

While the proposed method targets testability improvement of the datapath only, the results reported in Table III concerning the faults in the controller show that it improves a lot the testability of the whole circuit.

To have a fair comparison with the method proposed in Ref. [5] which in the general case involves modification of the controller and of the datapath, we used the example "Simple-RTL" for which only the controller is modified by TAO. Table IV reports the results in terms of fault coverage and area overhead. It can be seen that at least on this example, our method leads to better results.

DISCUSSION

As presented here, this method leaves some room for further improvements. In particular, the command words and transitions to add are the results of *one* test path for each module obtained from the datapath RT level testability analysis. This path can be insufficient for fully testing the module, even in the case of regular structures like datapaths. The testability analysis could be modified to produce several test paths and all these paths could be specified in the controller. This approach should lead to more testable designs but at the cost of a higher computation complexity and higher number of controller modifications. This last point gives room for trading off testability improvements vs. area overhead. Concerning area overhead, it must be noticed first that this method in its current version entails very little area penalty (cf. "Results"). Secondly, the larger the number of instructions of the original controller, the fewer test-specific instructions is needed. Furthermore, the larger the number of transitions, the easier it is to solve the test

TABLE I Test results

		Datapath			Datapath + initial control			Datapath + modified control		
	# Faults	FC%	TE%	CPU	TC%	TE%	CPU	TC%	TE%	CPU
Tseng	1848	100	100	22.9 s	71.45	76.89	1.66 h	98.11	99.68	7.8 min
Arfil	2910	100	100	5.5 s	27.01	41.24	8.31 h	92.23	99.48	40.8 min
Diffeq	2320	100	100	10.9 s	80.73	81.08	3.47 h	96.25	99.83	4.53 min
Simple-RTL	6178	99.87	99.89	3.42 min	96.10	96.23	3.4 h	98.98	99.81	6.4 min
Complex-RTL	1994	100	100	22.05 s	33.30	40.57	4.94 h	97.63	97.69	7.5 s

TABLE II Comparison with other strategies

	Scan_dec.				Scan_seq				Proposed method			
	FC %	TE %	Over %	Test time	FC %	TE %	Over %	Test time	FC %	TE %	Over %	Test time
Tseng	100	100	12.8	2440	96.37	98.13	0.5	1091	98.11	99.68	14.2	803
Arfil	100	100	17.8	2237	77.97	91.25	0.7	1113	92.23	99.48	15.9	2717
Diffeq	100	100	14.3	4305	95.78	100	0.5	802	96.25	99.83	12.4	1683
Simple-RTL	99.87	99.89	1.8	3935	97.03	97.04	0.5	5370	98.98	99.81	1.5	2366
Complex-RTL	100	100	12.3	3600	86.76	92.93	0.5	803	97.63	97.69	13.3	1026

TABLE III Controller testability

	# Faults	TC %	TE %	# Faults	TC %	TE %
Tseng	334	88.92	99.10	1000	92.50	96.60
Arfil	720	59.86	69.16	1982	88.39	97.97
Diffeq	212	75.71	80.18	808	89.42	99.25
Simple-RTL	400	92.00	93.00	432	94.90	99.53
Complex-RTL	898	79.62	81.73	1606	89.78	92.02

TABLE IV Comparison with TAO

	TAO		Our method	
	Initial circuit	Modified circuit	Initial circuit	Modified circuit
FC %	98.20	99.37	92.31	99.79
Area Overhead	3.6%		1.5%	

sequencing problem. In summary, the more complex the initial controller in terms of its transition density, the smaller the required overhead.

CONCLUSION

The controller modification method presented here raises the testability of a datapath to a level close to the achievable maximum. It is mainly based on a RTL testability analysis and does not require ATPG. When used in conjunction with the synthesis for testability of datapaths method presented in Ref. [8], highly testable circuits can be directly obtained at the cost of small area overhead.

As confirmed by the results, this method compares favorably with scan techniques while avoiding their drawbacks (additional delays, test time, number of test pins).

References

[1] Avra, L.J. and McCluskey, E.J. (1994) "High-Level Synthesis of Testable Designs: An Overview of University Systems", International Test Conference, Test Synthesis Seminar, Digest of Papers.

[2] Wagner, K.D. and Dey, S. (1996) "High-level synthesis for testability: a survey and perspective", *Proc. 33rd ACM/IEEE Design Automation Conf.*, 131–136.

[3] Dey, S., Gangaram, V. and Potkonjak, M. (1995) "A controller-based design-for-testability technique for controller-data path circuits", *Proc. Int. Conf. Comput.-Aided Des.*, 534–540.

[4] Hsu, F.F., Rudnick, E.M. and Patel, J.H. (1996) "Enhancing high-level control-flow for improved testability", *Proc. Int. Conf. Comput.-Aided Des.*, 322–328.

[5] Ravi, S., Lakshminarayana, G. and Jha, N.K. (1998) "TAO: regular expression based high-level testability analysis and optimization", *Proc. ITC*, 331–340.

[6] Bathia, S. and Jha, N.K. (1994) "Genesis: a behavioral synthesis system for hierarchical testability", *Proc. Eur. Des. Test Conf.*, 272–276.

[7] Flottes, M.L., Pires, R. and Rouzeyre, B. (1997) "Analyzing testability from behavioral to RT level", *Proc. Eur. Des. Test Conf.*, 158–165.

[8] Flottes, M.L., Pires, R. and Rouzeyre, B. (1998) "Alleviating DFT cost using testability driven HLS", *Proc. ATS*, 46–51.

[9] Raynaud Smith, V.J. (1983) A First Course in Formal Language Theory (Blackwell Scientific Publications, Oxford).

Marie-Lise Flottes received her Ph.D. degree in Electrical Engineering in 1990 from the University of Montpellier. She is currently a researcher for the French National Scientific Research Center. Since 1986, she has been conducting research in the domain of test at LIRMM laboratory. M.-L. Flottes' interests include Design For Testability, BIST, High Level Synthesis For Testability and test synthesis.

Bruno Rouzeyre received his M.S. degree in Mathematics in 1978, Ph.D. degree on CAD in 1984, all degrees from the University of Montpellier. Currently, he is a

Professor at the University of Montpellier and conducts his research at LIRMM. His main fields of research include High Level Synthesis, Test and Testability and Verification.

Laurent Volpe received his M.S. and Ph.D. degrees in Electrical Engineering from Montpellier University in 1996 and 1999, respectively. His research focused on the testability improvement from behavioral level (during High Level Synthesis) to RTL level (before logic optimization). Since 1999, he has been working for Cadence Design Systems.