How to Avoid Random Walks in Hierarchical Test Path Identification

Yiorgos Makris, Jamison Collins and Alex Orailoğlu
Computer Science & Engineering Department
University of California, San Diego, USA

Abstract

Hierarchical test approaches address the complexity of test generation through symbolic reachability paths that provide access to the I/Os of each module in a hierarchical design. While transparency behavior suitable for symbolic design traversal can be utilized for datapath modules, control modules do not exhibit transparency, and therefore require exhaustive search algorithms or expensive DFT hardware. In this paper we introduce a fast hierarchical test path identification methodology for circuits with no DFT at the controller-datapath interface. We introduce the concept of Influence Tables, modeling the impact of control states on the datapath, based on which appropriate state sequences for accessing each module are identified. Imposition of such sequences on a hierarchical test path identification algorithm, in the form of constraints, results in significant speedup over alternative non-DFT based approaches.

1. Introduction

Hierarchical test methodologies [1, 11, 13, 16] address large designs in a *divide-&-conquer* fashion, wherein test is generated for each module and subsequently translated and applied from the primary design I/Os. Within such approaches, the complexity of vector-by-vector test translation is alleviated through symbolic vector justification and response propagation over reachability paths for each module, as depicted in Figure (1). Composition of symbolic test translation paths is facilitated by the notion of modular transparency [5, 10, 14, 15], commonly exhibited by datapath components. In controller-datapath pairs, however, the exact datapath functionality is determined through precise sequences of controller-provided signals. Controllers do not exhibit bulk transparency behavior, thus limiting the effectiveness of hierarchical test path composition and the applicability of hierarchical test approaches. Since exact reasoning on control signal values requires expensive FSM search algorithms, the controllability and observability of the controller-datapath interface is typically enhanced through DFT hardware [6, 9]. Alternatively, controller redesign/resynthesis is used to provide additional control behavior, in support of hierarchical test path composition [2, 4, 8]. Due to area and performance considerations, however, such DFT modifications may not always be feasible.

In order to devise a viable hierarchical test path identification methodology for controller-datapath pairs without DFT, a trade-off concept similar to datapath transparency is required for the controller. Transparency behavior allows symbolic design traversal, reducing the complexity of datapath test translation, at the cost of sacrificing some of the inherent test translation capabilities of the design. Similarly, a simplification of the controller, preserving most of the behavior required for establishing reachability paths for each module, is required. An analogous approach is employed in the area of verification, where simplified control machines [3, 12] are used to represent the functionality of a design in a symbolic fashion. The reduced functionality is subsequently verified against a golden model, avoiding the exponential path explosion problem of verifying the complete design functionality.

In this paper, we introduce the concept of *Influence Tables*, a mechanism that captures the impact of each control state on the datapath portion of the design. *Influence Tables* are subsequently combined in order to derive valid control state sequences that are appropriate for testing each module in the design. These control state sequences substitute the controller and are used as constraints during the hierarchical test path search algorithm, in order to reduce backtracking and speed up convergence. The intricacies of the controller and the implications on hierarchical test path identification are discussed in section 2, along with a number of alternative approaches for combined controller-datapath search. The concept of *Influence Tables* is introduced in section 3 and its application for identifying appropriate control state sequences for each module is analyzed in section 4. The combined controller-datapath search scheme is presented in section 5 and experimental results are provided and compared to alternative search approaches in section 6.

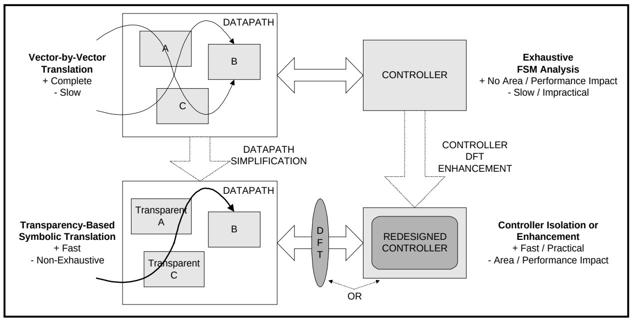


Figure (1): Handling Controller-Datapath Pairs

2. Motivation

Reachability paths from the primary I/Os of a design to the boundary of each module are required in order to apply test in a hierarchical fashion. While the transparency behavior typically exhibited by datapath modules can be easily composed into symbolic reachability paths, symbolic traversal of control modules is not feasible. Controllers produce precise sequences of signals, determining datapath functionality. In order to utilize control modules on hierarchical test paths, an exact value-based control signal analysis is required. However, exhaustive FSM analysis is expensive even for simple controllers. Loops in the controller, along with feedback signals from the datapath, complicate the process of reasoning on exact control state sequences and the corresponding control signal values. The problem is equivalent to the exponentially growing path enumeration problem on the controller FSM. In addition, exact value-based analysis on the feedback variables from the datapath requires exhaustive reasoning on the functionality of the datapath as well, further complicating combined controller-datapath search algorithms.

Unlike the datapath where transparency serves as a trade-off mechanism between complexity and accuracy, no similar concept has been devised for the controller. Due to the difficulty of the problem, solutions rely on DFT in order to either isolate the datapath from the controller, or to provide additional control behavior suitable for composing hierarchical test paths. However, DFT is not always an available option due to the incurred area and performance impact. In this case, reachability path construction for a module in a controller-datapath design requires that one of the following alternative search strategies be employed:

(i) The search is performed only on the datapath, considering control signals as primary inputs. If a solution is found, it is checked for compliance against the controller. If no appropriate control state sequence exists the datapath search is repeated for an alternative solution. The scheme is repeated until a combined solution is found or no more datapath solutions exist.

(ii) All possible state sequences of the controller are enumerated. A control state sequence is selected and the associated control signal values are imposed as constraints to the datapath search. If no solution is found, another control state sequence is selected and the scheme is repeated until a combined solution is found or no more control state sequences exist.

(iii) An intertwined search is performed, wherein each decision of the search algorithm on the datapath is checked immediately for compliance against the restrictions imposed by the controller.

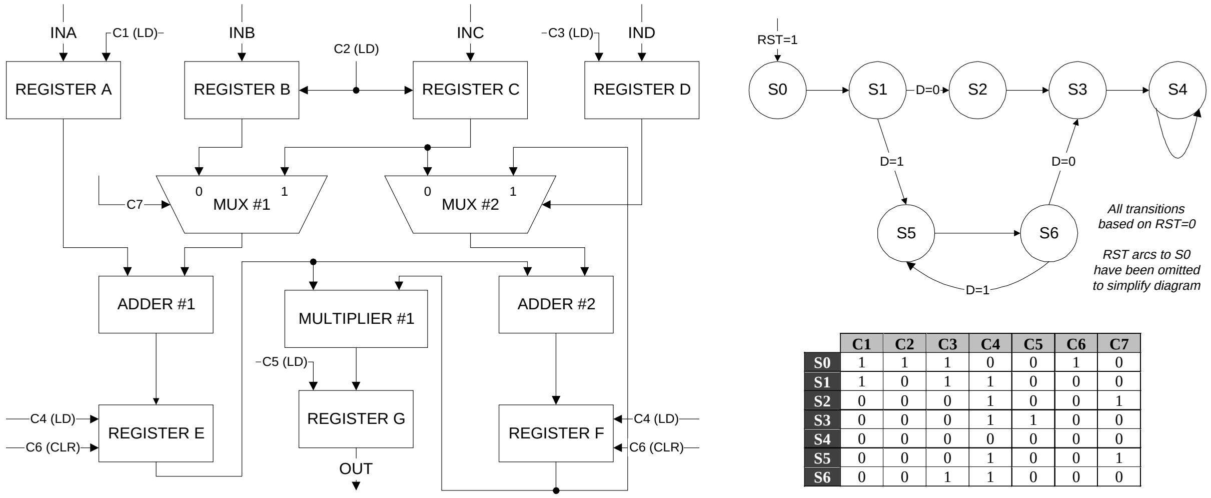
In the first two cases, the search is performed on one portion of the design and is only checked at the end for compliance against the other part. Such blind *trial & error* approaches are easy to implement; however, they are rather inefficient and result in long search times. The third approach is much harder to implement, since a list of appropriate control state sequences needs to be kept and updated each time a decision is made by the datapath search algorithm. However, each invalid path is only examined once, so this scheme is expected to converge much faster than the first two, since backtracking is significantly reduced. Nevertheless, it still requires an exhaustive FSM analysis that is overly expensive, making it hardly a viable alternative to DFT.

In contrast to these approaches, we introduce a fast methodology for identifying control state sequences appropriate for reaching each datapath module, based on the notion of *influence tables.* The proposed method avoids exhaustive examination of the controller functionality and therefore the resulting control state sequences are only potential but not guaranteed solutions. The identified control sequences are provided as constraints to the datapath hierarchical test path identification algorithm, effectively reducing the search space and speeding up convergence. In essence, influence tables provide an informed mechanism for substituting the controller with state sequences that are meaningful for establishing reachability paths for each module, while discarding the rest of the controller behavior.

3. Influence tables

The objective of the proposed scheme is to examine the controller and identify control state sequences that are appropriate for testing each module in the design. In order to model the datapath behavior under the impact of the controller, we introduce the concept of *influence tables* that capture the structural interaction between datapath state variables, for each state of the controller. Influence tables are consequently combined in order to identify control state sequences establishing reachability paths from the primary inputs to the inputs of the module under test and from the outputs of the module under test to the primary outputs. In support of the complexity vs. completeness trade-off discussed in section 2, influence tables capture only topological and not functional interaction between the primary inputs, the primary outputs and the state registers of the design. Therefore, the resulting control state sequences guarantee the existence of a reachability path but cannot reason on its test translation capabilities. The latter will still need to be evaluated through a datapath hierarchical test path identification algorithm, under the guidance of the identified control state sequence, as explained in section 5.

Influence Tables for Control States: The concept of *influence tables* is demonstrated through the controller-datapath pair example of Figure (2). The datapath portion of the circuit is given in Figure (2)(a) and the controller is described in Figure (2)(b). The influence tables for states *S0* and *S3* are given in Figure (2)(c). The top row of the influence table contains the primary inputs, state registers and constant values that during this particular state may influence the primary outputs or state registers noted on



	C1	C2	C3	C4	C5	C6	C7
S0	1	1	1	0	0	1	0
S1	1	0	1	1	0	0	0
S2	0	0	0	1	0	0	1
S3	0	0	0	1	1	0	0
S4	0	0	0	0	0	0	0
S5	0	0	0	1	0	0	1
S6	0	0	1	1	0	0	0

(a) Datapath Example

(b) Controller FSM

S0	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A								1				
B									1			
C										1		
D											1	
E												1
F												1
G							1					
OUT							1					

S3	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A	1											
B		1										
C			1									
D				1								
E	1	1										
F			D		1							
G					1	1						
OUT							1					

(c) Influence Tables for States S0 and S3

S1a	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A								1				
B		1										
C			1									
D											1	
E	1	1										
F			D		1							
G							1					
OUT							1					

S1b	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A								1				
B		1										
C			1									
D											1	
E	1	1										
F					1	D						
G							1					
OUT							1					

(d) Data-Dependent Alternative Influence: Splitting State S1 into S1a and S1b

S0- S1a	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A								1				
B									1			
C										1		
D											1	
E								1	1			
F										IND		1
G							1					
OUT							1					

S6- S3	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A	1											
B		1										
C			1									
D											1	
E	1	1										
F	1	1	IND									
G	1	1			1	D	1					
OUT							1					

(e) Influence Tables for Sequences of Control States S0-S1a and S6-S3

S5-S6	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A	1											
B		1										
C			1									
D											1	
E	1	1										
F	1	1			D	D						
G							1					
OUT							1					

S5-S6- S5-S6	A	B	C	D	E	F	G	IN	IN	IN	IN	‘0’
								A	B	C	D	
A	1											
B		1										
C			1									
D											1	
E	1	1										
F	1	1			D	D						
G							1					
OUT							1					

(f) Influence Tables for Controller Loop between States S5 and S6

Figure (2): Influence Tables Example

the leftmost column. A ‘1’ in a table location indicates that the signal entity of the corresponding row is influenced during this particular state by the signal entity of the corresponding column. For example, in the influence table of state *S0*, register *A* is influenced by the primary input *INA*, since LD=‘1’. Similarly, registers *E* and *F* are influenced by the constant value ‘0’, since CLR=‘1’ and both register *G* and the output *OUT* are influenced by register *G*, since LD=‘0’.

Conditional Influence: The influence table for state *S3* demonstrates how conditional influence is captured by the proposed scheme. In state *S3*, register *F* of the example circuit is influenced through the ADDER #2 by register *C*, under a condition on register *D*. In this case, register *D* does not directly influence register *F*, but it does control the potential impact of register *C* on register *F*. Therefore, the entry in the corresponding table location is not a ‘1’ but rather a ‘D’, indicating the conditional influence of register *C* on register *F*, based on a condition on register *D*.

Data-Dependent Alternative Influence: In Figure (2)(d) we demonstrate how the influence tables handle situations where alternative sets of signal entities may influence the signal entity under examination, based upon conditions on adjacent signal entities. In state *S1* for example, register *E* and – depending on register *D* – either register *C* or register *F* will influence register *F* through MUX #2. In order to model this “exclusive OR” type of behavior, the influence table for state *S1* is split into *S1a* and *S1b*, each capturing one of the possible influence behaviors, as depicted in Figure (2)(d). This is required only when datapath registers such as *D* are used to decide between alternative influence behaviors. In the case of MUX #1, which is controlled by a controller signal, the resolution is automatically performed, since distinct values on signal *C7* indicate distinct control states.

Influence Tables for Sequences of Control States: The effect of a control state sequence on the datapath is obtained by combining the influence tables. The entry (M,N) is filled in the combined table if M at the beginning of the control sequence influences N at the end of the control state sequence. As an example, the combined influence tables for state sequences S0-S1a and S6-S3 are demonstrated in Figure (2)(e). In the influence table of state *S0* the primary input *INA* influences register *A* which in turn influences register *E* in the influence table of state *S1a*. Therefore, in the combined influence table *S0-S1a*, the primary input *INA* influences register *E*. In a similar fashion, in the influence table of state *S0* the primary input *INC* influences register *C*, which in turn influences itself and upon a condition on register *D* also influences register *F* in state *S1a*. Taking into account that register *D* is influenced by *IND* in state *S1a*, in the combined influence table for the state sequence *S0-S1a* the primary input *INC* influences register *C* and upon a condition on the primary input *IND*, it also influences register *F*. The influence table for the control state sequence *S6-S3* is derived in a similar manner. However, in this case the transition from state *S6* to state *S3* is dependent on datapath feedback, which more specifically is a condition on register *D*.

Controller Loops: In this scheme, loops do not impose the same complexity burden as in the combined controller-datapath search approaches described in section 2. Since the influence tables perform a structural progress analysis and not a value progress analysis of the search algorithm, loops are only allowed when the corresponding influence table provides a distinct impact pattern. For example, as shown in Figure (2)(f), the influence table for the state sequence *S5-S6* is identical to the influence table of the sequence *S5-S6-S5-S6*. Since no additional structural influence is obtained when loop iterations are allowed, no additional influence tables are devised. However, since additional loop iterations may enhance the functional capabilities on the datapath, controller loop information is provided to the datapath search algorithm hat will have the option to follow the loop if required.

4. Control state sequence identification

The introduced concept of influence tables captures the impact of the controller states and sequences of states on the datapath portion of the design. This information is subsequently utilized in order to identify control state sequences that are appropriate for establishing reachability paths for testing each datapath module. Initially, the structural requirements for testing a module need to be defined. These requirements represent the primary inputs, registers and primary outputs that need to be controlled or observed in order to test a particular module. For example, testing ADDER #2 of Figure (2)(a) requires observing register *F* and controlling register *E*, register *D* and one of registers *F* and *C*. The next step is to identify a state *Sk* at which test may be applied to the module. A state *Sk* is a valid candidate for this purpose, if the following two conditions hold:

(i) There is a sequence of states with an influence table on which the registers that need to be controlled are only influenced by primary inputs and the sequence ends in a predecessor state of *Sk*. This denotes the *justification control state sequence* for the module.

(ii) There is a sequence of states with an influence table on which the registers that need to be observed influence at least one primary output and the sequence starts from a successor state of *Sk*. This denotes the *propagation control state sequence* for the module.

In the previous example, state *S2* qualifies for testing the ADDER #2, with *S0-S1a* being the justification and *S3-S4* being the propagation control state sequence. As shown in Figure (3), during *S0-S1a*, registers *C, E* and *D* are only affected by primary inputs and during *S3-S4*, register *F* influences the primary output. Since *S1a* is a predecessor state of *S2* and *S3* is a successor state of *S2*, the control state sequence S*0-S1a-S2-S3-S4* is appropriate for testing the ADDER #2 module. Additional checking is

Justification Control State Sequence

S0-S1a	A	B	C	D	E	F	G	IN A	IN B	IN C	IN D	'0'
A								1				
B									1			
C										1		
D											1	
E								1	1			
F										IND		1
G							1					
OUT							1					

Propagation Control State Sequence

S3-S4	A	B	C	D	E	F	G	IN A	IN B	IN C	IN D	'0'
A	1											
B		1										
C			1									
D				1								
E	1	1										
F			D		1							
G					1	1	1					
OUT					1	1	1					

Figure (3): Control State Sequence Identification for ADDER#2

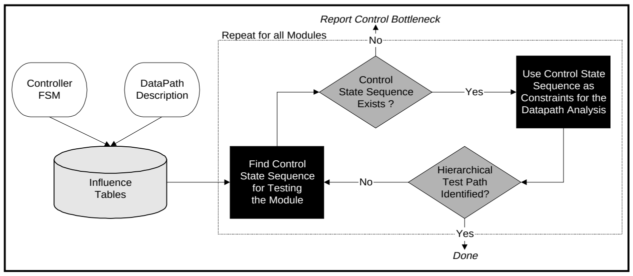

Report Control Bottleneck
Repeat for all Modules
No
Controller FSM
DataPath Description
Control State Sequence Exists ?
Yes
Use Control State Sequence as Constraints for the Datapath Analysis
Influence Tables
Find Control State Sequence for Testing the Module
No
Hierarchical Test Path Identified?
Yes
Done

Figure (4): Combined Controller-Datapath Search

performed in order to make sure that no conflict exists between the conditions and requirements on the path. For example, state transition *S1a-S2* requires a specific datapath feedback value on register *D* that is checked for possible conflicts against conditions required in the corresponding influence tables. Appropriate control state sequences for testing each module in the design can be similarly obtained. These control state sequences guarantee the existence of a reachability path for the module under test and are further combined with the datapath search scheme that will evaluate the justification and propagation capabilities of the corresponding path, as explained in the following section.

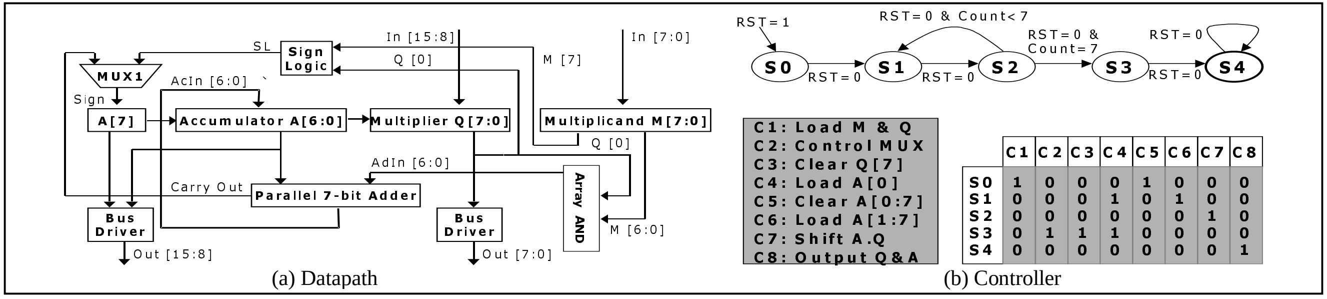
5. Combined test path identification

The influence table analysis of section 4 provides valid control state sequences that are meaningful for establishing reachability paths from the primary inputs to the module under test and from the module under test to the primary outputs. A datapath hierarchical test path composition methodology is still required in order to identify the exact reachability paths and to evaluate local to global test translation capabilities. However, knowing the appropriate control state sequence in advance, helps in guiding the datapath reachability analysis in a pruned search space. The control signals associated with these control state sequences are provided as *constraints* to the datapath search algorithm, reducing the number of alternative choices during hierarchical test path identification and thus speeding up the search process.

The combined scheme for controller-datapath hierarchical test path identification is depicted in Figure (4). Initially, the influence tables are derived from the controller-datapath description. Subsequently, a datapath module is selected and an appropriate control state sequence for testing this module is identified using the method of section 4. If no such sequence exists, a control testability bottleneck is reported and no solution exists unless DFT hardware is incorporated in the design. Otherwise, the identified sequence is provided in the form of constraints to the datapath hierarchical test path identification algorithm introduced in [10]. These constraints reduce the backtracking of the search algorithm, effectively speeding up convergence. If the testability requirements of the module are not satisfied, a new control state sequence is requested from the controller analysis scheme and the process is repeated until either a hierarchical test path is identified, or no more appropriate control state sequences can be found. In an effort to keep the hierarchical test application time low, control state sequences are examined in order of increasing length, so that the resulting reachability paths incur the shortest delay possible.

6. Experimental results

The efficiency of the combined controller-datapath search scheme is demonstrated on an 8-bit *shift-&-add* multiplier described in [7]. For the purpose of the experiment we employ the MULTIPLIER and ADDER modules of the controller-datapath implementation of this circuit, shown in Figure [5].



C1: Load M & Q
C2: Control MUX
C3: Clear Q[7]
C4: Load A[0]
C5: Clear A[0:7]
C6: Load A[1:7]
C7: Shift A.Q
C8: Output Q&A

	C1	C2	C3	C4	C5	C6	C7	C8
S0	1	0	0	0	1	0	0	0
S1	0	0	0	1	0	1	0	0
S2	0	0	0	0	0	0	1	0
S3	0	1	1	1	0	0	0	0
S4	0	0	0	0	0	0	0	1

(a) Datapath (b) Controller

Figure (5): Example Circuit

	Multiplier (Control Set A)	Multiplier (Control Set B)	Multiplier (Control Set C)	Multiplier (Control Set D)	Adder
Approach (i)	0.583 sec	4.25 sec	0.467 sec	N/T	N/T
Approach (ii)	17.97 sec	2.5 min	17.52 sec	N/T	N/T
Approach (iii)	0.277 sec	0.257 sec	0.271 sec	N/T	14.94 sec
Proposed Scheme	0.031 sec	0.052 sec	0.053 sec	2.98 sec	3.88 sec

Table (1): CPU Time Comparison for Hierarchical Test Path Identification Approaches

The three search approaches introduced in section 2 and the proposed methodology of section 5 have been implemented and applied on these modules. The search is expected to be successful on the MULTIPLIER where reachability paths exist for each of the four sets of values on the control inputs that are applied during normal functionality. However, the adder inputs are not independently controllable and the search should report that no reachability path exists for the ADDER.

The CPU time spent by each of the four search approaches on a 266 MHz Pentium™ II with 64 MB of RAM is provided in Table (1). The N/T entries in the table signify that the tool did not terminate within the time limit of 10 CPU minutes imposed throughout the experiment. The *trial-&-error* approaches (i) and (ii) spent the most time before converging. Although they were able to find solutions for the simple MULTIPLIER cases, they did not terminate for the *Control Set D* case which has a complicated solution, or for the ADDER where no solution exists. As expected, approach (iii), the intertwined controller-datapath exhaustive search, performed better than the first two approaches due to reduced backtracking. Nevertheless, it spent a significant amount of time in handling the ADDER case and did not terminate on the complicated MULTIPLIER case. Finally, the proposed methodology identified the expected solutions for all the MULTIPLIER cases and reported the lack of reachability paths for the ADDER. The time spent is almost an order of magnitude less than the best of the alternative approaches, verifying the power of the proposed scheme in identifying hierarchical test paths in controller-datapath pairs with no DFT.

7. Conclusions

Identification of hierarchical test paths in controller-datapath circuits without DFT requires efficient algorithms, capable of addressing the large search space of the design. In order to avoid the excessive backtracking of exhaustive search approaches, modular transparency is commonly used for symbolic datapath reasoning. The introduced *influence tables* concept complements such approaches by providing a judicious mechanism for pruning the control search space, thus facilitating an efficient combined search scheme. The impact of control states on the datapath is captured via influence tables, based on which appropriate control state sequences for testing each module are derived. Substitution of the controller by a judiciously selected set of state sequences results in significant convergence speedup of the controller-datapath search algorithm over alternative approaches.

References

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

[2] J. E. Carletta, C. A. Papachristou, “Behavioral Testability Insertion for Datapath/Controller Circuits”, *Journal of Electronic Testing: Theory & Applications*, Kluwer Academic Publishers, vol. 11, no. 1, pp. 9-28, 1997.

[3] K. T. Cheng, A. S. Krishnakumar, “Automatic Functional Test Generation using the Extended Finite State Machine Model”, *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pp. 86-91, 1993.

[4] S. Dey, V. Gangaram, M. Potkonjak, “A Controller Redesign Technique to Enhance Testability of Controller-Data Path Circuits”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 2, pp. 157-168, 1998.

[5] S. Freeman, “Test Generation for Data-Path Logic: The F-Path Method”, *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 421-427, 1988.

[6] I. Ghosh, A. Raghunathan, N. K. Jha, “A DFT Technique for RTL Circuits using Control/Data Flow Extraction”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 706-723, 1998.

[7] J.P. Hayes, *Computer Architecture and Organization*, McGraw-Hill, 3rd Edition, 1998.

[8] F. F. Hsu, E. M. Rudnick, J. H. Patel, “Enhancing High-Level Control-Flow for Improved Testability”, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 322-328, 1996.

[9] Y. Makris, A. Orailoğlu, “DFT Guidance Through RTL Test Justification and Propagation Analysis”, *Proceedings of the International Test Conference*, pp. 668-677, 1998.

[10] Y. Makris, A. Orailoğlu, “RTL Test Justification and Propagation Analysis for Modular Designs”, *Journal of Electronic Testing: Theory & Applications*, Kluwer Academic Publishers, vol. 13, no. 2, pp. 105-120, 1998.

[11] Y. Makris, J. Collins, A. Orailoğlu, P. Vishakantaiah, “TRANSPARENT: A System for RTL Testability Analysis, DFT Guidance and Hierarchical Test Generation”, *Proceedings of the Custom Integrated Circuits Conference*, pp. 159-162, 1999.

[12] D. Moundanos, J. A. Abraham, Y. U. Hoskote, “A Unified Framework for Design Validation and Manufacturing Test”, *Proceedings of the International Test Conference*, pp. 875-84, 1996.

[13] B. T. Murray, J. P. Hayes, “Hierarchical Test Generation Using Precomputed Tests for Modules”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 6, pp. 594-603, 1990.

[14] B. T. Murray, J. P. Hayes, “Test Propagation through Modules and Circuits”, *Proceedings of the International Test Conference*, pp. 748-757, 1991.

[15] P. Vishakantaiah, J. A. Abraham, M. S. Abadir, “Automatic Test Knowledge Extraction From VHDL (ATKET)”, *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 273-278, 1992.

[16] P. Vishakantaiah, J. A. Abraham, D. G. Saab, “CHEETA: Composition of Hierarchical Sequential Tests using ATKET”, *Proceedings of the International Test Conference*, pp. 606-615, 1993.