Observing Test Response of Embedded Cores through Surrounding Logic

Praveen K. Jaini
Synopsys, Inc.
Mountain View, CA 94041, USA

Nur A. Touba
Computer Engineering Research Center
University of Texas, Austin, TX 78712, USA

ABSTRACT

This paper addresses the problem of observing the test response of an embedded intellectual property core. For the core test set specified by the core vendor, the logic surrounding the core can mask errors at the output of the core such that faults in the core may go undetected. For intellectual property cores where there is no knowledge of the internal structure of the core (i.e., the core is a black box), no assumptions can be made about what errors the faults in the core may cause at the core outputs. All possible errors at the core outputs must be observable. Existing observation point insertion techniques (all of which are based on a fault model and require knowledge of the circuit structure) cannot be used. The conventional solution to this problem is to directly observe the core outputs by either multiplexing them to chip pins or placing a boundary scan around the core. This paper describes necessary and sufficient conditions (assuming no knowledge of the internal structure of the core) for guaranteeing that all errors at the outputs of the core can be observed through logic surrounding the core (combinational or sequential). A systematic method for inserting a minimal set of observation points necessary for testing a core (with either parallel or serial access) is presented.

1. INTRODUCTION

A growing trend in VLSI design is to use pre-designed and pre-verified blocks of logic, called *cores*, purchased from external vendors. New test problems arise from the fact that in order to protect intellectual property, a core vendor may not reveal the internal logic of a core (i.e., the core is a black box) [Chandramouli 96], [Zorian 97]. Traditional test techniques, such as automatic test pattern generation (ATPG) and fault simulation, require structural knowledge of the circuit-under-test. Thus, test vectors for a core cannot be generated by the user, but rather must be specified by the core vendor. The logic surrounding the core limits the accessibility of the core for applying the specified test vectors and observing their response. As a result, design-for-testability (DFT) techniques are needed to provide sufficient controllability and observability of a core so that it can be tested with the specified core test vectors.

One commonly used DFT approach for testing cores is to multiplex the core I/O's to the chip pins [Immaneni 90]. This provides *parallel access* to the core allowing a test vector to be applied to the core each clock cycle. Parallel access allows sequential testing of the core where ordered sequences of vectors need to be applied. This is required for sequential cores that do not have scan. For combinational cores and sequential cores that have scan, a commonly used DFT approach is to place a boundary scan collar around the core. This provides *serial access* to the core allowing test vectors to be scanned in and the test response of the core to be scanned out.

This paper focuses on the problem of observing the test response of the core when testing it with the test vectors specified by the core vendor. Both of the conventional DFT approaches for testing cores directly observe the test response by placing an observation point at *every* output of the core. This paper studies ways for "indirectly observing" the test response of a core through the logic surrounding it in order to reduce the number of observation points that must be added for testing the core. Assuming no knowledge of the internal structure of the core, necessary and sufficient conditions for guaranteeing no loss of fault coverage due to aliasing are shown. A method for inserting a minimal set of observation points necessary for testing a core (with either parallel access or serial access) is presented. For a muxed I/O DFT approach, this method can be used to significantly reduce the number of observation points that need to be routed to chip pins thereby reducing routing complexity and area. For a boundary scan DFT approach, this method can be used to significantly reduce the number of scan elements that are needed thereby reducing overhead and test time.

Note that the problem being addressed here is very different from the classical observation point insertion problem [Fox 77]. If the structure of the core was known, then the fault propagation paths could be determined and observation points could be inserted using existing methods. However, since the structure of intellectual property cores is not known, no assumptions can be made about what errors the faults in the core will cause at the core outputs. Thus, all possible errors at the core outputs must be detectable to ensure no loss of fault coverage. A sufficient set of observation points must be inserted to guarantee this.

2. THE CORE OBSERVATION PROBLEM

A core-based design consists of a set of intellectual property cores (which must be considered as black boxes for testing) and user-defined logic (UDL). For each core, there is a specified set of test vectors, provided by the core vendor that must be used for testing the core. Consider the case where a boundary scan is placed at the inputs of each core. For testing a particular core, each of the specified core test vectors can be scanned into the boundary scan at the inputs of the core and the internal scan inside the core and be applied to the core. The problem of interest is how to check the test response of the core to make sure that it is correct. The contents of the internal scan inside the core can be shifted out, so checking the state of the core is no problem. The problem lies in checking the outputs of the core. Any possible error at the outputs of the core must be detected.

The outputs of the core-under-test drive some combinational logic which will be referred to as the "output combinational logic block" (OCLB) for the core. Fig. 1 shows the general form of the OCLB for a core-under-test. The outputs of the OCLB can feed any of the following: the inputs of another core, UDL flip-flops that are either scanned or not scanned, or chip pins.

0-7803-5471-0/99/$10.00©1999 IEEE

The outputs of the OCLB can be divided based on whether or not they can be observed when testing the core-under-test:

$OCLB_{out}$(Scan Observable) - These are OCLB outputs that feed scan elements whose contents can be observed by shifting them out. Included in this set are OCLB outputs that feed UDL scan flip-flops or feed cores that have boundary scan at their inputs.

$OCLB_{out}$(Pin Observable) - These are OCLB outputs that directly feed chip output pins or are multiplexed with chip outputs pins such that they can be observed without any scan shifting. These are the outputs of interest when considering parallel access.

$OCLB_{out}$(Non-Observable) - These are OCLB outputs that cannot be observed when testing the core-under-test. OCLB outputs that feed UDL flip-flops that are not scanned are included in this set. Also included in this set are any OCLB outputs that feed cores that do not have boundary scan at their inputs.

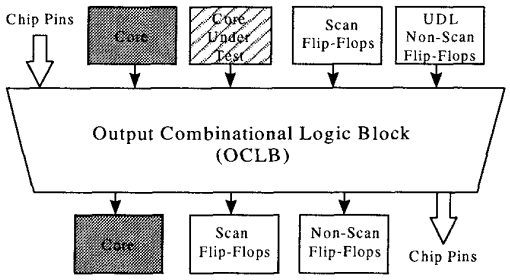

Figure 1. General Form of Output Combinational Logic Block (OCLB) for a Core-Under-Test

The values at the output of the OCLB depend not only on the core-under-test outputs, but also on other "side inputs." The side inputs to the OCLB can be any of the following: the outputs of another core, UDL flip-flops that are either scanned or not scanned, or chip pins. The procedure described in this paper is applicable to both combinational and sequential UDLs. A sequential UDL can always be represented as an OCLB with the flip-flops as side inputs. The inputs to the OCLB can be divided based on whether or not they can be controlled when testing the core-under-test:

$OCLB_{in}$(Scan Controllable) - These are OCLB side inputs whose value can be set by scanning in a vector. OCLB side inputs that are driven by UDL scan flip-flops are included in this set.

$OCLB_{in}$(Pin Controllable) - These are OCLB side inputs that are directly fed by chip input pins or are multiplexed with chip inputs pins such that they can be directly controlled.

$OCLB_{in}$(Non-Controlled) - These are OCLB side inputs that cannot be controlled when testing the core-under-test. OCLB side inputs which are driven by non-scanned UDL flip-flops or other cores who values cannot easily be controlled fall in this set.

$OCLB_{in}$(Core Response) - The OCLB inputs that are fed by the core-under-test form the final set. These are the OCLB inputs that need to be observed.

The test response of the core-under-test can be "indirectly observed" through the OCLB by looking at the values of the OCLB outputs that can be observed. The OCLB can be thought of as mapping a core response vector R into an observable output vector Z. For serial (i.e., scan) testing, outputs in both *$OCLB_{out}$(Scan Observable)* and *$OCLB_{out}$(Pin Observable)* can be observed. For parallel (i.e., non-scan) testing, only outputs in *$OCLB_{out}$(Pin Observable)* can observed. So the OCLB performs the mapping function, $R \rightarrow Z$, where

R = *$OCLB_{in}$ (Core Response)*

Z = *$OCLB_{out}$(Pin Observable)* for parallel testing, or

$OCLB_{out}$(Pin Observable) @ *$OCLB_{out}$(Scan Observable)* for serial testing ('@' denotes concatenation)

Note that the mapping function performed by the OCLB depends on the value of the side inputs.

The problem with indirect observation through surrounding logic is that there can be aliasing. For a particular core test vector, a fault in the core may cause a faulty core response vector R_{faulty} that differs from the fault-free core response vector $R_{fault\text{-}free}$, but if the mapping through the OCLB causes the corresponding Z_{faulty} and $Z_{fault\text{-}free}$ vectors to be the same, then the fault goes undetected. In order to avoid fault coverage reduction when indirectly observing the test response of the core-under-test, steps must be taken to guarantee that no aliasing will occur.

3. CONDITIONS FOR GUARANTEEING NO ALIASING

If the structure of the core was known, then the set of faulty core response vectors for each fault in the core could be determined. No aliasing with respect to (w.r.t.) a particular fault class could be guaranteed by ensuring that for each fault, at least one of the faulty core responses that it causes maps to a different observable OCLB output vector than the corresponding fault-free core response vector maps to. This ensures that each fault can be detected. Existing observation point insertion techniques are based on avoiding aliasing w.r.t. a particular fault class.

For intellectual property cores, because there is no way to determine how each fault will affect the core response, all possible faulty core responses must be detectable for each of the specified core test vectors in order to guarantee no loss of fault coverage. Consider the following definition:

Definition: An observable OCLB output vector Z is *unique* (w.r.t. a particular assignment of the side inputs) if the core response vector R that maps to it is the only vector in the input space of the OCLB mapping to it.

If there is a fault in the core that is detected by a test vector, it will produce a faulty core response vector R_{faulty} that differs from the fault-free core response vector $R_{fault\text{-}free}$. Let Z_{faulty} and $Z_{fault\text{-}free}$ be the observable OCLB output vectors that R_{faulty} and $R_{fault\text{-}free}$ respectively map to. For aliasing to occur Z_{faulty} must be same as $Z_{fault\text{-}free}$ which is not possible if $Z_{fault\text{-}free}$ is *unique*. Therefore, no aliasing can occur for any fault f in a core if fault f is detected by a test vector whose corresponding fault-free core response vector maps to a unique observable OCLB output vector.

Given the set of fault-free core response vectors for the core test set specified by the core vendor, an assignment of the controllable side inputs must be found for each fault-free core response vector, $R_{fault\text{-}free}$, so that it maps to an observable OCLB output vector, $Z_{fault\text{-}free}$, that is unique. If no such assignment of the controllable side inputs exists for one of the fault-free core response vectors, then one or more observation points must be inserted in the OCLB to prevent aliasing. Adding observation points adds extra bits to the observable OCLB output vector.

The location of the observation points must be chosen so that observable OCLB output vector, $Z_{fault\text{-}free}$, becomes unique.

4. OBSERVATION POINT INSERTION FOR EMBEDDED CORES

Based on the discussion in the previous section, the problem of inserting a sufficient set of observation points for testing an embedded core can be formulated as follows: enough observable outputs must be added to the OCLB to ensure that all the fault-free core response vectors for a specified core test set map to unique observable OCLB output vectors. Inserting observation points at all of the outputs of the core is a simple brute-force solution and will always satisfy this requirement and can be very costly and impractical in some situations. In this paper, the problem of interest is to insert a minimal set of observation points in the OCLB that guarantees no aliasing (thus providing the same fault coverage obtained when directly observing the core outputs).

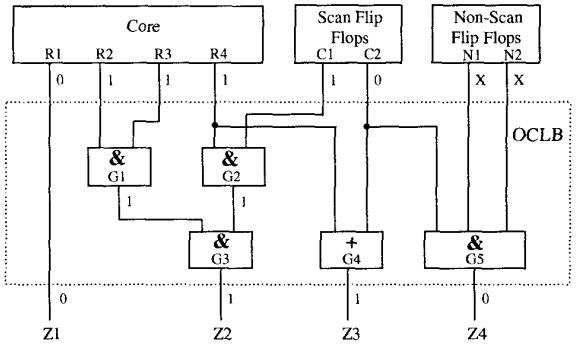

Figure 2. Example of $R_{fault\text{-}free}$ (0111) Mapping to a Unique $Z_{fault\text{-}free}$ (0110)

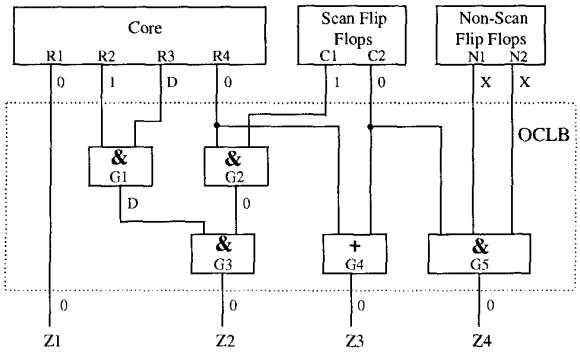

Figure 3. Simulation of 5-Valued Cube for Difference Between $R_{fault\text{-}free}$ (0110) and R_{faulty} (0100).

Consider the example in Fig. 2 in which an OCLB is shown with 4 inputs coming from the core-under-test and 2 side inputs each coming from UDL scan flip-flops and UDL non-scan flip-flops. One of the core response vectors specified by the core supplier is 0111. If the vector scanned into the scan flip-flops (which are completely controllable) is 10, then the output of the OCLB is 0110. Note that the values in the non-scan flip-flops are assumed to be unknown X's. Any faulty core response vector resulting from some fault in the core changes the OCLB output and thus the fault is detected. Therefore the core response vector 0111 ($R_{fault\text{-}free}$) maps to a unique OCLB output 0110 ($Z_{fault\text{-}free}$). Another core response vector specified is 0110 ($R_{fault\text{-}free}$), and the corresponding OCLB output is 0000 ($Z_{fault\text{-}free}$). If a fault in the core changes the core response vector to 0100, 0010 or 0000, the fault goes undetected as all three core responses map to 0000 which is same as the fault-free output vector (i.e., $Z_{faulty} = Z_{fault\text{-}free}$). Observation points must be inserted to distinguish $Z_{fault\text{-}free}$ from Z_{faulty}. 5-valued logic cubes can be formed to represent the difference between $R_{fault\text{-}free}$ and each R_{faulty}. In 5-valued logic, each logic value is in the following set {0, 1, X, D, D'}. D (D') represents the case where the fault-free value is 1 (0) and the faulty value is 0 (1). For the case where $R_{fault\text{-}free}$ is 0110, and R_{faulty} is 0100, the 5-valued cube representing the difference would be 01D0. An observation point inserted at any of the nodes to which either a D or D' value propagates to will allow the fault-free core response vector to be distinguished from the faulty core response vector. An example of 5-valued simulation is shown in Fig. 3. An observation point at either the core output R3 or at the output of gate G1 will allow the fault-free core response 0110 be distinguished from the faulty core response 0100.

A pair of fault-free and faulty core response vectors ($R_{fault\text{-}free}$, R_{faulty}) that cannot be distinguished at the observable OCLB outputs will be referred to as a "conflict." In order to eliminate a conflict, an observation point that allows that distinguishes the two must be inserted. Making the fault-free observable OCLB output vector ($Z_{fault\text{-}free}$) unique requires that all conflicts be eliminated. In the example in Fig. 3, there are 3 conflicts. Table 1 shows the propagation information of the 5-valued simulation of the each of the conflict cube. Notice that inserting an observation point at the output of gate G1 resolves all three conflicts.

Table 1. Propagation Information

5-Valued Conflict Cubes	Nodes R2	R3	G1
(0110, 0100) → 01D0		X	X
(0110, 0010) → 0D10	X		X
(0110, 0000) → 0DD0	X	X	X

The problem of inserting a minimal set of observation points can be formulated as a column-covering problem. A matrix is formed in which each row corresponds to a conflict. The columns in the matrix correspond to nodes in the OCLB. For each row, an 'X' is placed in each column that corresponds to a node where the faulty and fault-free core response vectors have different values. A set of columns is said to cover the matrix if every row has a 'X' in at least one of the columns in the set. The minimal set of columns needed to cover all the rows corresponds to the minimal set of nodes for which inserting observation points will make all of the core response vectors map to unique observable OCLB output vectors. Column covering is an NP-complete problem, but efficient techniques and heuristics exist for solving it [Coudert 96]. The next section describes a systematic procedure for forming this matrix for a specified set of fault-free core responses.

5. OBSERVATION POINT INSERTION PROCEDURE

Consider all of the fault-free core response vectors corresponding to the core test set specified by the core vendor. If all the conflicts due to pairs of fault-free and faulty core response vectors mapping to the same observable OCLB output vector could be included as rows in the covering matrix, then an optimal set of observation points could be selected with column covering. However, finding all the conflicts at once is generally not feasible because there can be an exponential number of conflicts in the worst case. This is due to the fact that (assuming no knowledge of the structure of the core) there are an exponential number of possible faulty core response vectors (2^n-1 for a core with n outputs). Since finding all the conflicts is computationally not feasible, the proposed strategy is to find a representative sample of the conflicts, form the covering matrix, and then insert observation points based on those conflicts. The observation points that are inserted will resolve not only the conflicts explicitly included in the covering matrix, but also could eliminate most other conflicts as well. After the observation points have been inserted, then the number of remaining conflicts will be much smaller. A new covering matrix can then be formed and the process repeated. Iterations can be done until the procedure converges to a solution where there are no conflicts remaining.

5.1 Forming Initial Covering Matrix

The heuristic used for forming the initial covering matrix is to consider conflicts between all pairs of fault-free and faulty core response vectors that differ in only one bit. The pairs that differ in only one bit tend to be the most difficult to differentiate. If all the pairs with single bit differences can be distinguished, then typically most of the pairs with multiple bit difference can also be distinguished (but not necessarily all because there is a possibility of multiple bit differences canceling out each other if there is fanout reconvergence).

If n is the number of core outputs, then for each fault-free core response vector, there are n faulty core response vectors that differ in only one bit. For each of the n pairs of fault-free and faulty core response vectors, 5-valued simulation is done. If no D or D' propagates to an observable OCLB output, then that means there is a conflict and hence a row is added to the covering matrix. If T is the number of core test vectors, then $(n)(T)$ 5-valued vectors are simulated to form the initial covering matrix.

When forming the initial covering matrix, one degree of freedom that can be used to reduce the number of conflicts is in selecting the values of the controllable side inputs to the OCLB. A different value of the side inputs can be chosen for each fault-free core response vector since the values of the side inputs can be set each time a new core test vector is applied. Given a conflict for a particular fault-free core response vector, ATPG techniques can be used to find an assignment of the side inputs that will eliminate the conflict if possible. ATPG techniques can be used to try to propagate a D or D' in the 5-valued cube (that corresponds to the conflict) to an observable OCLB output. The ATPG procedure attempts to specify the values of the side inputs to advance the D-frontier [Abramovici 90]. If the conflict can be removed by specifying only a subset of the side inputs, then the remaining side inputs are left at unassigned values (X's). If another conflict is found, then the unassigned side inputs can be specified to eliminate the conflict if possible.

Once the initial covering matrix is formed, then a column covering procedure is used to find a minimal set of observation points. The observation points are then inserted into the OCLB. Now all faulty core response vectors that differ in only one bit from the fault-free core response vector cannot alias. Consequently, most of the faulty core responses that differ in multiple bits will also be detected, but some may not. In applications where slightly less than 100% fault coverage is acceptable, then no more observation points need to be inserted. To guarantee that absolutely no aliasing will occur, however, the remaining conflicts must be identified and eliminated.

5.2 Forming Subsequent Covering Matrices

To identify any remaining conflicts for a fault-free core response vector ($R_{fault\text{-}free}$) that maps to an observable OCLB output vector ($Z_{fault\text{-}free}$), a standard APTG tool can be used as illustrated in Fig. 4. Three fictional gates are added to the OCLB for the purpose of identifying the remaining conflicts. An AND gate ($G1$) is appended to the output of the OCLB and inverters are added on the inputs of the AND gate that correspond to each bit in $Z_{fault\text{-}free}$ that is a '0'. The output of gate $G1$ will be true only for core response vectors that map to $Z_{fault\text{-}free}$. A NAND gate ($G2$) is added such that its output is true only if the core response vector is different from $R_{fault\text{-}free}$. The output of gate $G1$ and gate $G2$ are ANDed together through gate $G3$. A stuck-at-0 fault at the output of gate $G3$ is then targeted with ATPG. If the fault is redundant, then that means that $Z_{fault\text{-}free}$ is unique (no core response maps to it besides $R_{fault\text{-}free}$) and thus there are no more conflicts for $R_{fault\text{-}free}$. However, if a test cube (test pattern where the unspecified inputs are left as X's) is found for the fault, then that test cube represents a conflict. A 5-valued cube is formed to represent the differences between $R_{fault\text{-}free}$ and the test cube. 5-valued simulation is used as before to find all the nodes that the differences propagate to. This information is used to add a row for the conflict to the covering matrix.

So the procedure for forming a covering matrix after the initial covering matrix is as follows. ATPG is done for each

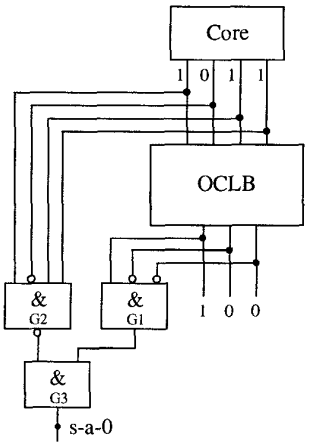

Figure 4. Using ATPG to Identify Conflicts with $R_{fault\text{-}free}$ (1011) → $Z_{fault\text{-}free}$ (101).

core response vector. If no test cube is found for a particular core response vector, then the core response vector can be dropped from further consideration since it maps to a unique output vector. If a test cube is found then a row is added to the covering matrix as described before. After all the rows have been added, then a column covering procedure is used to find a minimal set of additional test points needed to cover this matrix. The observation points are then inserted in the OCLB, and this procedure is repeated until a point is reached where all of the fault-free core response vectors map to unique observable OCLB output vectors. At that point, the procedure completes. The resulting set of observation points inserted in the OCLB guarantees that no aliasing can occur. The computation for the whole procedure is not adverse considering the fact that the OCLB is in general relatively small glue logic.

5.3 Combining Observation Points

When the observation points are inserted, it is sometimes possible to combine some of them through XOR gates. Consider two columns, c_i and c_j, in a minimal column covering set that was selected for a covering matrix. If c_i and c_j are disjoint, then for all the conflicts in the matrix, there are no cases where errors propagate to both observation points at the same time. Thus, they can safely be combined through an XOR gate with no aliasing. If c_i and c_j are not disjoint, then for the rows where c_i and c_j are both X's, errors will propagate to both of the observation points and cancel out if they are combined with an XOR gate. However, if for the rows where c_i and c_j are both X's, there are other columns in the covering set that are also X's, then c_i and c_j can still be combined with an XOR gate. The reason is that even though the errors will cancel out and not be observed through the c_i and c_j, it is okay because the errors can still be observed through another observation point.

In short, the way to check if two observation points can be combined through an XOR gate is to exclusive-OR together the corresponding columns in the covering matrix and see if all the rows are still covered. If all the rows are still covered, then there will be no aliasing due to combining them with the XOR gate.

6. EXPERIMENTAL RESULTS

Experimental results for the procedure described in this paper were generated for some core-based designs constructed from the ISCAS benchmark circuits. In each design, one of the benchmark circuits is considered to be an intellectual property core and treated as a black box while another benchmark circuit is considered to be the logic that is driven by the core. Table 2 shows the number of outputs and the size of the specified core test set for the benchmark circuits that were used as cores.

In Table 3, results are shown for the case where some of the sequential benchmark circuits were considered to be the logic driven by the core. All of the flip-flops in the sequential benchmark circuits are assumed to be part of a scan chain and are treated as controllable side inputs. The combinational logic in the sequential benchmark circuits is considered to be the OCLB for the core. For the benchmark circuits that were used, if there is a mismatch between the number of outputs of the core and the number of primary inputs of the benchmark circuit, the extra core outputs were discarded. For each benchmark circuit, three different cores are considered. A comparison is made between the number of direct observation points required (which is equal to the number of outputs in the core) and the number of indirect observation points that were inserted with the method described here. As can be seen, there is a significant reduction in the number of observation points.

Table 2. Information About Cores

Name	Num. Outputs	Num. Vectors
dsip	197	87
s13207	152	106
s15850	684	125

Table 3. Results for Sequential UDL with Scan

Circuit				Core					
				dsip		s13207		s15850	
Name	PI	PO	FF	Direct Obsv Points	Indir. Obsv Points	Direct Obsv Points	Indir. Obsv Points	Direct Obsv Points	Indir. Obsv Points
s641	35	24	19	35	12	35	14	35	16
s820	18	19	5	18	17	18	16	18	17
s1196	14	14	18	14	10	14	9	14	11
s1423	17	5	74	17	6	17	5	17	7
s5378	35	49	179	35	25	35	22	35	23
s9234	36	39	211	36	14	36	10	36	15

7. CONCLUSIONS

The new test point insertion procedure described here makes no assumptions about the types of faults or structure of the core whose output is being observed. No loss of fault coverage due to aliasing is guaranteed for both combinational and sequential user-defined logic. While described in the context of intellectual property core testing, the techniques in this paper have applications in other areas such as concurrent error checking and fault diagnosis where observation of errors at the outputs of a module is required.

ACKNOWLEDGEMENTS

This material is based on work supported in part by the National Science Foundation under Grant No. MIP-9702236, and in part by the Texas Advanced Research Program under Grant No. 1997-003658-369.

REFERENCES

[Abramovici 90] Abramovici M., M.A. Breuer, A.D. Friedman, *Digital Systems Testing and Testable Design*, Piscataway, NJ: IEEE Press, 1990, Chapter 6.

[Chandramouli 96] Chandramouli, R., and S. Pateras, "Testing Systems on a Chip," *IEEE Spectrum*, pp. 42-47, Nov. 1996.

[Coudert 96] Coudert, O., "On Solving Covering Problems," *Proc. of 33rd Design Automation Conf.*, pp. 197-202, 1996.

[Immaneni 90] Immaneni, V., and S. Raman, "Direct Access Test Scheme - Design of Block and Core Cells for Embedded ASICS," *Proc. of Int. Test Conf.*, pp. 488-492, 1990.

[Zorian 97] Zorian, Y., "Test Requirements for Embedded Core-based Systems and IEEE P-1500," *Proc. of International Test Conference*, pp. 191-199, 1997.