Testing Embedded Cores Using Partial Isolation Rings

Nur A. Touba and Bahram Pouya

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084

Abstract

Intellectual property cores pose a significant test challenge. The core supplier may not give any information about the internal logic of the core, but simply provide a set of test vectors for the core which guarantees a particular fault coverage. If the core is embedded within a larger design, then the problem is how to apply the specified test vectors to the core and how to test the user-defined logic around the core. A simple and fast solution is to place a full isolation ring (i.e., boundary scan) around the core, however, the area and performance overhead for this may not be acceptable in many applications. This paper presents a systematic method for designing a partial isolation ring that provides the same fault coverage as a full isolation ring, but avoids adding MUXes on critical timing paths and reduces area overhead. Efficient ATPG techniques are used to analyze the user-defined logic surrounding the core and identify a maximal set of core inputs and outputs (that includes the critical timing paths) that do not need to be included in the partial isolation ring. Several different partial isolation ring selection strategies that vary in computational complexity are described. Experimental results are shown comparing the different strategies.

1. Introduction

In order to shorten product development cycles for integrated circuits and systems, pre-designed cores are increasingly being used. An important issue is how to test intellectual property cores embedded within a larger design. If the core supplier is not willing to give any information about the internal logic of the core (i.e., it is a black box), then ATPG and fault simulation cannot be performed. The core supplier may only provide a set of test vectors for the core that guarantees a particular fault coverage. If the core is embedded within a larger design, then the problem is how to apply the specified test vectors to the core and how to test the user-defined logic (UDL) around the core. One simple solution is to use multiplexing to make the inputs and outputs of the core accessible to the chip pins [Immaneni 90]. However, this approach does not help with testing the UDL around the core and thus results in degraded fault coverage. Another approach is to place an isolation ring around the core, as illustrated in Fig. 1. The isolation ring is essentially a boundary scan that provides full controllability of the inputs of the core and full observability of the outputs of the core as well as providing full observability of the logic driving the core and full controllability of the logic that is driven by the core. The drawback of using a full isolation ring is the large area and performance overhead that it adds. A boundary scan element and associated routing is required for each input and output of the core, and a MUX delay is added to every path to and from the core. As a result, a full isolation ring may not be an acceptable solution for many high performance applications.

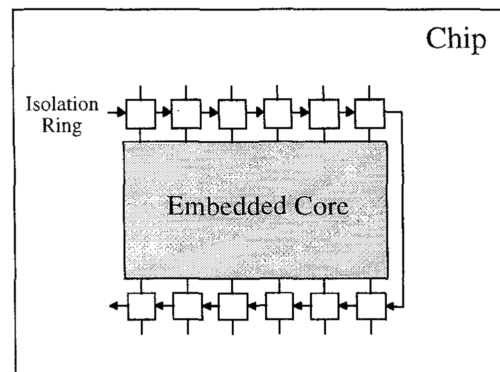

Figure 1. Isolation Ring for Testing Embedded Core

A grid-based direct access methodology for testing core-based designs was recently introduced by Bhatia, *et. al*, at CrossCheck [Bhatia 96]. This method differs from the embedded grid test method described in [Gheewala 89] and [Chandra 91], in that it uses a "soft" netlist level grid as opposed to a "hard" grid embedded at the base of gate arrays. The "soft" netlist level grid described in [Bhatia 96] provides direct access to storage elements, observation test points, and bi-directional test points via the chip pins. The basic approach is to place bi-directional test points at the inputs and outputs of the embedded core, and matrix accessible storage elements and observation test points in the UDL to provide sufficient fault coverage. This approach requires new

0-8186-7810-0/97 $10.00 © 1997 IEEE

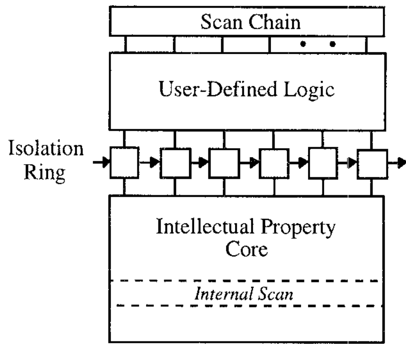

Figure 2. Architecture for Testing Core and User-Defined Logic Driving Core

library cells to implement the matrix accessible storage elements, bi-directional test points, and I/O pads, and it requires global routing of the test grid.

This paper presents a method for reducing the area and performance overhead of using an isolation ring. A systematic procedure is described for designing a partial isolation ring that provides the same fault coverage as a full isolation ring but avoids adding MUXes on critical timing paths and reduces area overhead. The core is treated as a black box; no information about the core is assumed other than the set of test vectors for the core specified by the core supplier. The procedure uses ATPG techniques to analyze the UDL surrounding the core and identify a maximal set of core inputs and outputs (that includes the critical timing paths) that do not need to be included in the partial isolation ring.

2. Selecting Core Inputs to Include in Partial Isolation Ring

In this section, the focus is on reducing the number of isolation ring elements at the inputs of the core. Reducing the isolation ring elements at the outputs of the core will be the focus of Sec. 3. The isolation ring elements at the inputs of the core serve two purposes. The first is that they provide controllability of the inputs to the core. The test vectors for the core can be shifted into the isolation ring and applied directly to the core. Note that the core may have an internal scan chain for controlling internal flip-flops, in which case in addition to shifting a test vector into the isolation ring, a test vector would also be shifted into the internal scan chain of the core. The second purpose of the isolation ring elements at the inputs of the core is to provide observability to the UDL driving the core. Assuming that a scan methodology is used, then the architecture is as illustrated in Fig. 2. Testing the UDL driving the core involves shifting a test pattern into the scan chain and loading the response into the isolation ring.

For a partial isolation ring, the core inputs are partitioned into two sets: *IR*, the set of core inputs that are included in the isolation ring; and *NIR*, the set of core inputs that are not included in the isolation ring. For the *NIR* core inputs, an alternate means is required for applying the specified core test vectors to them as well as for observing the outputs of the UDL that drives them. The approach used here for applying the specified core test vectors to the *NIR* core inputs is to justify the vectors through the UDL that drives *NIR* core inputs. The approach used here for observing the outputs of the UDL that drives *NIR* core inputs is to perform "space compaction" by exclusive-ORing each output that drives an *NIR* core input with an output that drives an *IR* core input and feeding the combined output into the partial isolation ring (no aliasing for single stuck-at faults will occur if the techniques described in [Chakrabarty 94] are used). Section 2.1 presents a procedure for selecting which core inputs to include in the isolation ring to ensure that the specified test vectors can be applied, and Sec. 2.2 describes how to design a space compactor to observe all of the outputs of the UDL driving the core.

2.1 Selection Procedure for Core Inputs

In general, the output space of the UDL driving the core may not contain all of the test vectors for the core that are specified by the core supplier. This does not mean that faults in the core are necessarily redundant. A large set of test vectors may exist for a fault in the core, but the particular test vector that is specified by the core supplier may happen to be one that is not contained in the output space of the UDL. Other test vectors for the fault may exist in the output space of the UDL driving the core such that the fault is not redundant, but those test vectors cannot be identified without knowledge of the internal logic of the core. So the problem is to select a set of core inputs to be included in the partial isolation ring that will enable all of the specified test vectors to be applied to the core. This section describes a selection procedure that minimizes the number of core inputs included in the partial isolation ring.

The first step is to see which of the specified test vectors for the core can be justified through the UDL driving the core and which cannot. Each vector can be checked by appending an AND gate to the output of the UDL and adding inverters on the inputs of the AND gate that correspond to each bit in the vector that is a '0'. An ATPG tool can then be used to target a stuck-at 0 fault at the output of the AND gate. If the fault is detectable, then that means that it is possible to justify the specified test vector through the UDL to the inputs of the core. The specified test vectors that cannot be justified through the UDL are the ones that need to be considered in designing the partial isolation ring. If all of the specified test vectors can be justified through the UDL, then no isolation ring elements are needed at the inputs of the core.

If there are n core inputs, then there are 2^n possible partial isolation rings because each core input can either be included in the partial isolation ring or not included. A particular partial isolation ring enables a test vector to be applied to the core if the subset of bits in the test vector corresponding to core inputs not included in the partial isolation ring can be justified through the UDL. A particular partial isolation ring is a "solution" if it enables all of the specified test vectors to be applied to the core. Checking if a particular partial isolation ring is a solution can be done by simply appending AND gates (where the inputs of the AND gates correspond to the core inputs not included in the partial isolation ring) to the output of the UDL and performing ATPG as described before.

An exhaustive approach for selecting a partial isolation ring would be to check all 2^n possibilities to see which are solutions and then choose the solution with the fewest isolation ring elements where none of the core inputs on critical timing paths are included in the ring. However, if the number of core inputs is large (i.e., n is large), then this would not be computationally feasible. Thus, some other strategies for searching the exponential space of possible partial isolation rings are needed. Several strategies varying in computational complexity are described below. The first step is always to remove the core inputs that are on critical timing paths from the isolation ring since those are the ones that impact performance. Once the core inputs on critical timing paths have been removed, then the remaining task is to remove as many additional core inputs as possible.

Hill-Climbing - $O(n)$:

Each core input is removed from the isolation ring one at a time. When a core input is removed, a check is made to see if the resulting partial isolation ring is still a solution. If it is not a solution, then the core input is added back to the ring. If it is a solution, then the core input is left off the ring. After the procedures loops through all n core inputs, it stops and the resulting partial isolation ring is the solution. This search strategy is not very clever, but it is very fast.

Clique Hill-Climbing - $O(n^2)$:

Let A, B, and C, be core inputs, and let *RING - {A,B}* be the partial isolation ring that results from removing the set of core inputs A and B from the isolation ring. If *RING - {A,B}*, *RING - {B,C}*, and *RING - {A,C}* are all solutions, then that does not imply that *RING - {A,B,C}* is a solution. Consider the case where *111* is a core test vector, and the vectors *110*, *011*, and *101*, can all be justified through the UDL. Removing any two core inputs from the partial isolation ring is a solution, but removing all three is not a solution. If *RING - {A,B}* is not a solution, however, then that does imply that *RING - {A,B,C}* is not a solution. In other words, *RING - {A,B}*, *RING - {B,C}*, and *RING - {A,C}*, all being solutions is necessary but not sufficient for *RING - {A,B,C}* to be a solution. This fact can be used to compute a bound on the total number of core inputs that can be removed from an isolation ring. The way this is done is by forming a compatibility graph in which each node corresponds to a core input and an edge is placed between two nodes if the partial isolation ring that results from removing both of the corresponding core inputs from the ring is a solution. The largest clique (complete subgraph) in this compatibility graph corresponds to the largest set of core inputs that can potentially be removed from the isolation ring.

A search strategy for the best partial isolation ring is to construct the compatibility graph and then use it to guide the order in which hill-climbing is performed. The best possible solution that can be obtained by removing a particular core input from the isolation ring is bound by the size of the largest clique that the core input is in. So the idea is to guide the hill-climbing procedure so that it first considers the core inputs contained in the largest cliques since those are the ones that are most likely to be in the best solution. The largest cliques are identified, and the core inputs are ordered for the hill-climbing procedure. Identifying the largest cliques in a graph is an NP-complete problem, however, good heuristics exist for it.

Clique Greedy - $O(n^3)$:

A compatibility graph is constructed as previously described. The largest clique is identified, and the core input corresponding to the node that is in the largest clique and has the largest number of edges is removed from the isolation ring. This procedure is then recursively repeated for the resulting partial isolation ring. Each time a core input is removed from the partial isolation ring, more constraints are added for further removing core inputs, so as a result, edges are removed from the compatibility graph. Updating the compatibility graph each time provides more accurate information to guide the search.

Branch and Bound - $O(2^n)$:

Since the largest clique in the compatibility graph for a partial isolation ring provides a bound on the best possible solution, this information can be used to avoid unproductive searching. When exploring a branch of the search tree, if the largest clique in the compatibility graph indicates that the best possible solution cannot be better than the best partial isolation ring solution that has been found so far, then the branch can be cut-off. In the worst case, this procedure is exponential, but in general, the search space can be greatly reduced. The branch and bound procedure can be run as long as desired. Whenever it is stopped, it will provide the best solution that it has found so far. If it is allowed to run to completion, then it is guaranteed to find the optimum solution.

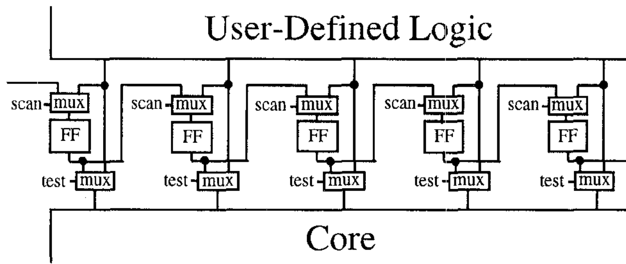

Figure 3. Example: Full Isolation Ring

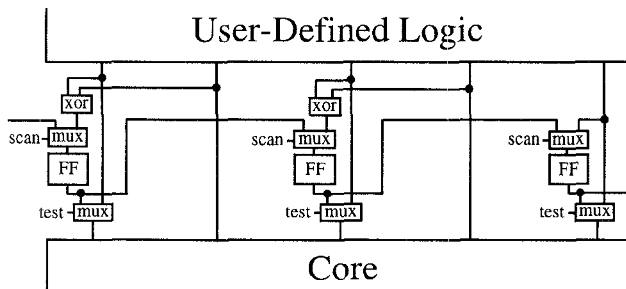

Figure 4. Example: Partial Isolation Ring

2.2 Observing User-Defined Logic Driving Core with Partial Isolation Ring

Once the set of core inputs to be included in the partial isolation ring has been selected, the remaining task is to design a space compactor that will combine the outputs of the UDL driving the core so that their test response can be observed in the partial isolation ring. If n is the number of outputs and k is the size of the partial isolation ring, then a space compactor with $(n-k)$ exclusive-OR gates can be used. The techniques described by Chakrabarty and Hayes in [Chakrabarty 94] can be used to ensure that the UDL driving the core can be tested for single stuck-at faults without any aliasing in the space compactor. These techniques involve either modifying the test set or making minor circuit modifications to ensure that each fault is sensitized to an odd number of outputs. This ensures that the effect of the fault will not be masked in the exclusive-OR gates, but rather be captured in the partial isolation ring. In Fig. 3, an example of a full isolation ring is shown. In Fig. 4, the 2nd and 4th core inputs are removed from the isolation ring and exclusive-OR gates are used to compact the outputs of the combinational logic driving those core inputs.

3. Selecting Core Outputs to Include In Partial Isolation Ring

In this section, the focus shifts to the core outputs. The isolation ring elements at the outputs of the core serve two purposes. They provide observability to the outputs of the core, and they provide controllability to the inputs of the UDL that is driven by the outputs of the core. If it were possible to justify a sufficient set of test vectors for the UDL driven by the core through the core itself, then only a shift register would be needed for observing the outputs of the core (or the outputs could be multiplexed to chip pins). A shift register is much better than an isolation ring because a shift register does not add any logic on the system paths whereas an isolation ring adds a MUX delay on every path and requires that a test mode line be routed to control the MUXes. The problem is that if the core contains sequential logic, it may be very difficult to place the core in the states needed to justify test vectors for the UDL at the core outputs. The core may have an internal scan path, but that won't help if nothing is known about the internal logic of the core. Using a full isolation ring solves this problem by enabling the test vectors to be shifted in and directly applied to the UDL. In order to reduce the area and performance penalty of a full isolation ring, a procedure is described here for replacing some of the isolation ring elements with shift register elements (or multiplexing them to the chip outputs). The idea is to justify a subset of the bits of each test vector through the core and use a partial isolation ring to shift in the rest of the bits. A small example is shown in Fig. 5 where each test vector is generated by shifting the 1st, 2nd, and 5th bits into a partial isolation ring and justifying the 3rd and 4th bits through the core. The goal of this approach is to avoid adding isolation ring elements on the critical timing paths.

The set of core outputs that do not need to be included in the isolation ring depends on what vectors can be justified at the core outputs. One set of vectors that can easily be justified at the core outputs is the output response of the test vectors specified by the core supplier. Since the core will be tested with the test vectors specified by the core supplier, the output response of those vectors will be generated at the core outputs and can be used in testing the UDL that is driven by the core outputs. Consider the example in Fig. 5. One of the test vectors for the core that the core supplier specified is *1011010* and the corresponding core output vector is *01100*. The test vector *1011010* is applied to the core inputs. If the core has an internal scan path, then the appropriate scan vector specified by the core supplier is also shifted into the core's internal scan path. The corresponding core output vector *01100* is now justified at the core outputs. This vector can be used for testing the UDL driven by the core outputs. Moreover, by using the partial isolation ring that is shown in Fig. 5, any test vector that has 10 in the third and fourth bit positions (having the form *XX10X*) can be applied to the UDL driven by the core outputs by shifting the bits into the partial isolation ring. Based on the set of vectors that are easy to justify at the outputs of a core, a partial isolation ring can be designed so that a sufficient set of test vectors can be applied to the UDL

driven by the core outputs. Note that depending on the functionality of the core, it may be possible to identify more vectors that are easy to justify at the core outputs than just the output response of the test vectors specified by the core supplier.

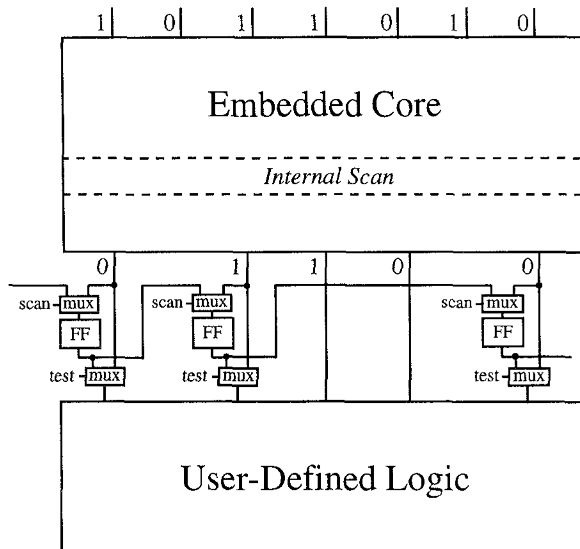

Figure 5. Example of Using a Partial Isolation Ring at Core Outputs

3.1 Checking Fault Coverage Provided by Partial Isolation Ring at Core Outputs

Given the set of vectors that can easily be justified at the outputs of a core, which will be referred to as the "core output vectors," the problem is to determine which core outputs can be removed from the isolation ring without reducing the fault coverage of the UDL. Let *FRR* be the set of faults that require an isolation ring in order to be detected. The set of faults in *FRR* is equal to all the faults in the UDL minus the faults that are redundant (cannot be detected by any set of test vectors) and minus the faults that can be detected by the core output vectors. Computing the set of faults in *FRR* requires doing a redundancy check on the UDL and doing fault simulation for the core output vectors.

If a set of core outputs is removed from the isolation ring, then the vectors that can be justified at those core outputs may be restricted. Consider the example in Fig. 5. If it is not possible to justify a *11* on the 3rd and 4th core outputs, then no test vectors of the form *XX11X* can be applied to the UDL. The question then is whether or not all of the faults in *FRR* can be detected without using any test vectors of the form *XX11X*. This can be determined by doing ATPG with constraints on the allowable values of the bits of a test vector. Techniques for doing such constrained ATPG have been described in [Konijnenburg 93, 95] and [Wohl 96]. These ATPG techniques consider the constraints as early as possible in the ATPG decision making process. These ATPG techniques can be used to check whether the faults in *FRR* can be detected under the constraints imposed by the use of a particular partial isolation ring. If some faults in *FRR* cannot be detected due to the restricted set of vectors that can be applied by a particular partial isolation ring, then the partial isolation ring is not a solution because it will degrade fault coverage.

Given the set of core outputs not included in the partial isolation ring, the ATPG constraints can be determined by analyzing the core output vectors. For example, if the 2nd, 3rd, and 4th core outputs are not included in the partial isolation ring and the core output vectors are *110101*, *100101*, *011010*, and *011101*, then the ATPG constraints are that all test vectors must be of the form *X101XX*, *X001XX*, *X110XX*, or *X111XX*. This set of *allowable test cubes* can be simplified to *XX01XX* and *X11XXX* by combining adjacent cubes. So the procedure for checking if a particular partial isolation ring is a solution is as follows. The bit positions in the core output vectors that correspond to core outputs that are included in the partial isolation ring are set to don't cares (X's) to form the set of allowable test cubes. A two-level minimizer is then used to minimize the set of allowable test cubes. ATPG is then performed for the faults in *FRR* under the constraint that test vectors must be contained in the allowable test cubes. If all of the faults in *FRR* can be detected under these constraints, then the partial isolation ring is a solution.

An obvious concern about the procedure for checking if a partial isolation ring is a solution is the amount of ATPG that is required. Several approaches can be taken for reducing the amount of ATPG. Two techniques that are very effective are: 1) Doing fault simulation for some random patterns to reduce the number of faults for which ATPG is needed. The random patterns can be generated by randomly specifying the unspecified bit positions in allowable test cubes; this ensures that the random patterns are contained in the allowable test cubes. 2) Recording each test vector that is found for each fault. Since the same faults are targeted each time a partial isolation ring is considered, a check can be made to see if one of the previously identified test vectors for the fault satisfies the current constraints. If so, then ATPG is not necessary. These two techniques can dramatically reduce the amount of ATPG that is required. Note that it is not all the faults that are being considered, but only those in *FRR*. Note also that as soon as one fault is found to be untestable under the constraints, the partial isolation ring is classified as not being a solution, thus almost all of the ATPG is targeting faults that are testable. Time consuming ATPG for untestable faults is limited to no more than once per partial isolation ring being considered.

3.2 Selection Procedure for Core Outputs

The procedure for selecting which core outputs to remove from the partial isolation ring is exactly the same as that for selecting which core inputs to remove. The only difference is the method for determining whether or not the resulting partial isolation ring is a solution. Thus, all of the search strategies that were described before in Sec. 2.1 can be used.

Table 1. Information about Designs

#	UDL		Core			
	Name	Inputs	Name	Inputs	Faults	Vectors
1	vda	17	s838	34	820	185
2	k2	45	s9234	36	6010	285
3	apex7	49	C499	41	928	68
4	x4	94	s13207	62	8450	831
5	apex6	135	s15850	77	4055	738
6	C2670	233	dsip	140	5854	63
7	C5315-a	178	C5315-b	117	493	71
8	C7552-a	207	C7552-b	261	877	92

4. Experimental Results

The procedures described in this paper were used to select partial isolation rings for some designs that were constructed from the MCNC benchmark circuits. In each design, one of the benchmark circuits was considered to be an intellectual property core and treated as a black box while another benchmark circuit was considered to be UDL either at the input or at the output of the core. Two of the benchmark circuits, *C5315* and *C7552*, were partitioned into two parts where one part was considered to be a core and the other part was considered to be UDL.

4.1 Selecting Partial Isolation Ring at Core Inputs

Table 1 gives information about the designs that were used for experiments with selecting partial isolation rings at the inputs of a core. For each design, the name of the UDL driving the core is shown followed by the name of the core. The number of inputs to the core, number of faults in the core, and number of specified test vectors for the core are shown (the test vectors were obtained by doing ATPG on the core).

Table 2. Results for Partial Isolation Rings at Inputs of Cores

#	UDL Name	Core Name	Full Ring Size	Hill Climbing O(n)		Clique Hill Climb $O(n^2)$		Clique Greedy $O(n^3)$		Branch & Bound $O(2^n)$	
				Ring Size	Time	Ring Size	Time	Ring Size	Time	Ring Size	Time
1	vda	s838	34	31	1	30	1	29	4	29	28
2	k2	s9234	36	33	1	31	3	31	9	31	70
3	apex7	C499	41	9	1	7	2	6	31	6	200
4	x4	s13207	62	9	6	9	15	NA	NA	NA	NA
5	apex6	s15850	77	8	11	8	16	NA	NA	NA	NA
6	C2670	dsip	140	29	15	28	36	NA	NA	NA	NA
7	C5315-a	C5315-b	117	74	12	73	30	NA	NA	NA	NA
8	C7552-a	C7552-b	261	69	10	63	43	NA	NA	NA	NA

Table 3. Results for Partial Isolation Rings at Outputs of Cores

Core			UDL Name	Num FRR	Full Ring Size	Hill Climbing O(n)		Clique Hill Climb $O(n^2)$	
Name	Outputs	Output Vectors				Ring Size	Time	Ring Size	Time
mm30a	30	210	x1	128	30	11	1	8	6
s9234	39	285	C880	56	39	7	2	7	12
s5378	49	402	apex6	54	49	7	19	7	140
sbc	56	229	i5	87	56	25	12	19	16
s15850	150	738	i4	79	150	28	22	25	180
dsip	197	63	i7	21	197	4	10	4	710
C5315-a	117	34	C5315-b	30	117	12	13	11	200
C7552-a	261	63	C7552-b	80	261	12	25	12	230

Table 2 shows the results for each of the designs in Table 1. The number of isolation ring elements in a full isolation ring is shown followed by results for each of the four search strategies that were described for selecting a partial isolation ring. For each search strategy, the number of isolation ring elements in the selected partial isolation ring is shown along with the CPU time in minutes (the CPU times would be much smaller if an industrial quality ATPG tool was used). The procedures were run on a Sun UltraSPARC. An *NA* is placed in the entries where the procedure ran for more than 5 hours. As can be seen, the *Clique Greedy* and *Branch & Bound* search strategies could only be used when the number of core inputs (n) was small (less than 50). The *Branch & Bound* search strategy is guaranteed to find the optimum solution, so for the circuits where all four search strategies were used, the results indicate that the *Clique Hill Climbing* strategy found something very close to the optimum solution. In some cases, the simple *Hill Climbing* strategy performed poorly because it selected a core input early on that was not compatible with many other core inputs.

The results for the two circuits, *C5315* and *C7552*, that were partitioned into two parts are interesting. In each case, when ATPG is done on the combined circuit, faults in part b can be detected by vectors in the output space of part a. However, the fact that so many isolation ring elements are needed indicates that many of the test vectors that are obtained by doing ATPG on part b independent of part a are not in the output space of part a. There are some redundant faults in the combined circuit, so some of those faults may become detectable when part b is considered independent of part a and of course the test vectors for those faults will definitely not be in the output space for part a. For the test vectors specified by the core supplier (obtained by doing ATPG independent from the UDL) that are not in the output space of the UDL, there is no way to tell which are for faults that are redundant in the combined circuit (UDL cascaded with core) and which are for faults that can be detected by some other vector in the output space of the UDL. As a result, a partial isolation ring is needed to apply all of the test vectors provided by the core supplier in order to be safe.

4.2 Selecting Partial Isolation Ring at Core Outputs

Table 3 shows results for experiments with selecting partial isolation rings at the outputs of a core. For each design, the name of the core, number of outputs in the core, and number of core output vectors is shown. This is followed by the name of the UDL that is driven by the core and the number of faults that require an isolation ring in order to be detected (faults in *FRR*). Results are shown for two search strategies. For each search strategy, the number of isolation ring elements in the selected partial isolation ring is shown along with the CPU time in minutes.

5. Conclusions

There are an exponential number of possible partial isolation rings. ATPG techniques were described for efficiently checking whether a particular partial isolation ring will degrade fault coverage. Exact and heuristic search strategies were presented for selecting a partial isolation ring for both the inputs and outputs of the core. These different search strategies provide a means for trading off between computation time and the optimality of the result. Results indicate significant area and performance overhead reduction is possible by using a partial isolation ring compared with a full isolation ring.

Acknowledgments

The authors would like to thank Prof. Edward McCluskey and Rob Norwood from the Center for Reliable Computing at Stanford University for their helpful comments and suggestions. This work is part of the TOPS project at the Center for Reliable Computing at Stanford University and was supported by the Advanced Research Projects Agency under prime contract No. DABT63-94-C-0045.

References

[Bhatia 96] Bhatia, S., T. Gheewala, and P. Varma, "A Unifying Methodology for Intellectual Property and Custom Logic Testing," *Proc. of Int. Test Conf.*, pp. 639-648, 1996.

[Chakrabarty 94] Chakrabarty, K., and J.P. Hayes, "Efficient Test Response Compression for Multiple-Output Circuits," *Proc. of Int. Test Conf.*, pp. 501-510, 1994.

[Chandra 91] Chandra, S., T. Ferry, T. Gheewala, and K. Pierce, "ATPG Based on a Novel Grid Addressable Latch Element," *Proc. of 28th Design Automation Conf.*, pp. 282-286, 1991.

[Gheewala 89] Gheewala, T., "CrossCheck: A Cell Based VLSI Testability Solution," *Proc. 26th Design Automation Conf.*, pp. 706-709, 1989.

[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.

[Konijnenburg 93] Konijnenburg, M.H., J.Th. van der Linden, and A.J. van de Goor, "Test Pattern Generation with Restrictors," *Proc. of Int. Test Conf.*, pp. 598-605, 1993.

[Konijnenburg 95] Konijnenburg, M.H., J.Th. van der Linden, and A.J. van de Goor, "Automatic Test Pattern Generation for Industrial Circuits with Restrictors," *Microelectronics Journal*, Vol. 26, No. 7, pp. 598-605, Oct. 1995.

[Wohl 96] Wohl, P., and J. Waicukauski, "Test Generation for Ultra-Large Circuit Using ATPG Constraints and Test-Pattern Templates," *Proc. of Int. Test Conf.*, pp. 13-20, 1996.