Efficient Compression of X-Masking Control Data via Dynamic Channel Allocation

Asad A. Bawa, M. Tauseef Rab, and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
Email: {bawa, tauseefrab, touba}@utexas.edu

Abstract

A scheme is presented which uses one sequential linear decompressor to decompress test cubes and another sequential linear decompressor to decompress mask control data for masking unknown X's in the output response. However, instead of the conventional approach of using a fixed number of tester channels for driving each, a method for dynamically adjusting the number of channels driving each is proposed. The key idea is that by carefully ordering the test cubes such that the channel requirements of the decompressor used for masking the previous output response (currently being scanned out) can be balanced with the channel requirements of the decompressor used for scanning in the current test cube. By matching test cubes with more specified bits with output response with fewer X's, and test cubes with fewer specified bits with output response with more X's through careful ordering of the test cubes, the total aggregate worst-case number of tester channels needed with dynamic channel allocation can be reduced considerably compared with conventional fixed channel allocation approaches thereby improving compression. For test cube and output response pairs where there are excess free variables, a method is described for boosting observability by selectively ANDing together outputs of multiple stages of the mask decompressor for driving the masking logic. Experimental results show that compression and observability can be significantly improved with the proposed scheme.

Keywords: output compression; output masking; X-masking

1. Introduction

There are various sources of unknown *X* values in the output response from things such as false paths, multi-cycle paths, analog blocks, tri-state buses, uninitialized memories, etc. Dealing with *X*'s is a major issue for test compression and BIST because *X*'s in the output response can corrupt the final signature making it unknown.

One approach for handling *X*'s which does not require modifying the circuit-under-test (CUT) is to use *X-masking* in which *X*'s are masked out at the input to the compactor, e.g., a multiple-input signature register (MISR). The key issue for *X*-masking is how to select which bits to mask. It would be ideal if only the *X* values were masked and all non-*X* values were not masked, but the amount of mask control information required for this level of precision is prohibitive. Consequently, some loss of observability is necessary in order to sufficiently compress the mask control information. A number of schemes for performing *X*-masking have been proposed.

One approach is to mask the output of scan chains that contains *X*'s during a whole scan out [Barnhart 01], [Rajski 02]. If *m* is the number of scan chains, this approach would require *m* control bits (one per scan chain to indicate if it should be masked or not masked) for each test vector. In this approach, the amount of mask control data is small, but observability is lost for all scan cells in each masked scan chain which can be problematic if some fault is only observable in a particular scan chain.

To avoid losing observability of entire scan chains, another approach is to select which scan chains to observe in each shift cycle. To reduce the number of control bits required each shift cycle in this case, a small number of modes can be defined in which certain combinations of scan chains can be either masked or observed [Chickermane 04], [Wohl 07]. In this case, the number of control bits required each shift cycle is $\log_2$ of the number of modes.

Some techniques have been developed to exploit correlations in the location of *X*'s across output responses to reduce the amount of mask control data. In [Wohl 08], the scan cells that capture the most *X*'s are stitched in "*X*-chains" which allows the masking modes to be better optimized [Wohl 08]. In [Wohl 10] and [Czysz 10], the mask control data for output responses with similar *X* locations are merged together to reduce the amount of mask control data loaded from the tester.

Mask control data can be compressed by using a sequential linear decompressor such as a linear feedback shift register (LFSR) or ring generator [Mrugalski 04]. In [Naruse 03], the mask control data on a per-shift basis is compressed using LFSR reseeding [Könemann 91]. A drawback of this approach is that since the output of an LFSR is 1 and 0 roughly 50% of the time, half of the non-*X* scan cells get masked as well, so overall observability is reduced by 50%. In [Volkerink 05], multiple stages of an LFSR are ANDed together to reduce the probability of masking non-*X* values and hence increases observability. In [Wang 08a], a minimal set of scan cells that need to be observed to detect all faults is determined by looking at where the fault effects for each fault propagate during the automatic test pattern generation (ATPG)

process. This information is used to simplify the linear equations when performing LFSR reseeding to obtain better compression of the mask control data. Further optimizations for LFSR reseeding are described in [Wang 08b] by using group masking. In [Czysz 10] and [Wohl 10], sequential linear decompressors are used to load a shadow register which holds the mask control data. In this case, when the same masking pattern can be used for consecutive shift cycles, it is not necessary to reload the shadow register. It only needs to be loaded when the masking pattern changes.

Note that when using sequential linear decompressors, the amount of compressed data that is required is proportional to the number of specified bits that need to be generated. Each bit stored on the tester is a free variable that can be assigned either a 0 or 1. To generate a certain set of specified bits at the output of a sequential linear decompressor, a linear equation needs to be solved for each specified (i.e., care) bit. In order to solve the system of linear equations, more free variables than care bits are required. So the number of free variables that need to be delivered to the decompressor is proportional to the number of care bits it needs to generate. In the case of using a sequential linear to generate mask control data, the number of free variables is proportional to the number of *X*'s in the output response. The number of *X*'s can vary considerably from one output response to the next. The same issue is present when using a sequential linear decompressor for compressing test cubes (i.e., test vectors in which the unassigned inputs are left as don't cares). The number of free variables required to encode a test cube is proportional to the number of care bits in the test cube, which again can vary considerably from one test cube to the next. The tester channels deliver free variables in a steady stream each clock cycle. The number of tester channels and clock cycles used to decompress either a test cube or mask control data is set based on the maximum number of care bits that need to be generated. So the compression achieved depends on the worst case test cube or mask control data.

It is not feasible to use the same sequential linear decompressor for encoding both test cubes and mask control data because when encoding a test cube, the values in the non-care bits cannot be determined until the linear equations are solved. However, without knowing the values of non-care bits in the test cube, it is not possible to determine the presence and location of the *X*'s in the output response. So for this reason, separate decompressors have to be used for the test cubes and the output response. Once the test cubes are encoded using the *test cube decompressor* and the fully specified decompressed test vectors are known, then the output response can be determined and the required mask control data can be encoded with the *mask decompressor*. Existing state-of-the-art techniques, e.g., [Czysz 10] and [Wohl 10], are based on having separate decompressors as described above.

Note that the scan out of an output response is done concurrently with the scan in of the next test vector. So the free variables that are required to encode a test cube need to be delivered at the same time as the free variable that are required to encode the mask control data for the output response of the previously applied test cube.

The idea proposed in this paper is that rather than the conventional approach of having a fixed number of tester channels feeding the test cube decompressor and a fixed number feeding the mask decompressor, the number of channels feeding each is dynamically adjusted. The key to making this very efficient is careful ordering of the test cubes. Test cubes with larger numbers of care bits requiring more tester channels to bring in free variables are placed after test cubes whose output response has smaller numbers of *X*'s requiring mask control data with smaller number of care bits and hence needing fewer tester channels to bring in free variables. On the flip-side, test cubes with smaller numbers of care bits are placed after test cubes whose output response has larger numbers of *X*'s. By balancing the number of free-variables needed by each decompressor, the worst-case total aggregate number of free variables needed for decompressing any test cube and output response pair can be minimized. This allows greater compression for a given number of total tester channels that are available from the tester.

To illustrate the improvement in encoding efficiency that is possible with the proposed approach, consider the following example. Suppose the worst-case number of free variables needed to encode any test cube was 1000, and the worst-case number of free variables needed to encode the mask data for any output response was 600. If there were 16 tester channels, then the conventional approach would be designed with 10 channels feeding the test cube decompressor and 6 channels feeding the output response decompressor. In this scenario, 100 clock cycles would be used to decompress each test cube so that enough free variables are supplied to the decompressors to encode the worst-case test cube and worst-case mask data. With the proposed approach, the worst-case test cube needing 1000 free variables would get matched with the output response requiring the fewest free variables, let's say it was 100 for example. When decompressing that test cube, 14 channels could be allocated for the test cube decompressor and 2 channels could be allocated to the mask decompressor. In that case, only $\lceil 1000/14 \rceil = 72$ clock cycles would be sufficient to provide enough free variables for both the test cube decompressor and mask decompressor (which would receive 2x72=144 free variables). For the worst-case mask data needing 600 free variables, perhaps it could be matched with a test cube needing let's say 400 free variables, then when decompressing that test cube, 9 channels could be allocated to the mask decompressor and 7 channels could be allocated to the test cube decompressor. In that case, only $\lceil 600/9 \rceil = 67$ clock cycles or more would be sufficient to provide enough free variables for both the mask decompressor and test cube decompressor (which would receive 7*67=469 free variables. Looking across all test cube and mask data pairs, if the worst-case number of clock cycles needed turned out to be 72, then that would determine the number of clock cycles used when decompressing all test cube and mask data pairs. The improvement in compression achieved by the proposed approach in this example would be (100-72)/72 =

39%. Both the test time and the tester memory usage would be reduced by 39%.

The basic idea in this paper can be applied on top of any of the existing schemes that use sequential linear decompressors for compressing masking data to achieve greater compression. For simplicity and without loss of generality, the approach is described here using a basic structure of having a sequential linear decompressor feeding masking logic on a per-shift basis similar to what is used in [Naruse 03] and [Volkerink 05]. Adding the optimizations described in other papers (e.g., [Wang 08a, 08b]) or adapting it for other architectures (e.g., Czysz 10]) is straightforward.

One technique for boosting observability that is proposed here uses the idea from [Volkerink 05] of ANDing together the outputs of multiple stages of a sequential linear decompressor to reduce the probability of non-*X* values getting masked. When multiple stages are ANDed together, it increases the number of free variables needed to encode the mask control data for eliminating the *X*'s. However, it may often be the case that more free variables can be supplied to the mask decompressor without impacting the compression. In other words, if it is determined, as in the example mentioned earlier, that say 72 clock cycles need to be used for the worst-case decompression, there are likely many other test cubes that need much less than the worst-case number of free-variables, so there is essentially an excess number of free-variables supplied versus the number needed for many test cubes. These excess free variables can be used to increase observability of non-*X* values by ANDing together multiple stages of the decompressor when producing the mask control data. This extra observability essentially comes at no extra cost in terms of test time or additional data on the tester. Additional observability helps to reduce test vector count and detect non-modeled faults.

The paper is organized as follows: Sec. 2 describes the proposed dynamic channel allocation scheme. Sec. 3 describes how observability can be increased by selectively ANDing together stages of the decompressor. Sec. 4 shows experimental results, and Sec. 6 is a conclusion.

2. Dynamic Channel Allocation

The proposed scheme is based on the idea of dynamically adjusting the number of tester channels that are used to feed the test cube decompressor and the masking decompressor on a per test cube basis. The hardware for implementing this is shown in Fig. 1. There are *b* channels coming from the tester, and each sequential linear decompressor has *b* injector inputs. At the start of decompressing each test cube, in the very first clock cycle, the data coming from the tester channels is loaded into the control register *q*. The binary number stored in register *q* is the number of tester channels to use for the test cube decompressor. This number is fed to a selector which selects $channel_1$ through $channel_q$ and masks off $channel_{q+1}$ through $channel_b$ for the test cube decompressor by ANDing them with 0. It does the opposite for the masking decompressor, i.e., $channel_1$ through $channel_q$ are masked and $channel_{q+1}$ through $channel_b$ are passed through. After the first clock cycle in which register *q* is loaded, in all subsequent clock cycles, the tester channels are used to inject free variables into the decompressors based on the channel allocation determined by *q*.

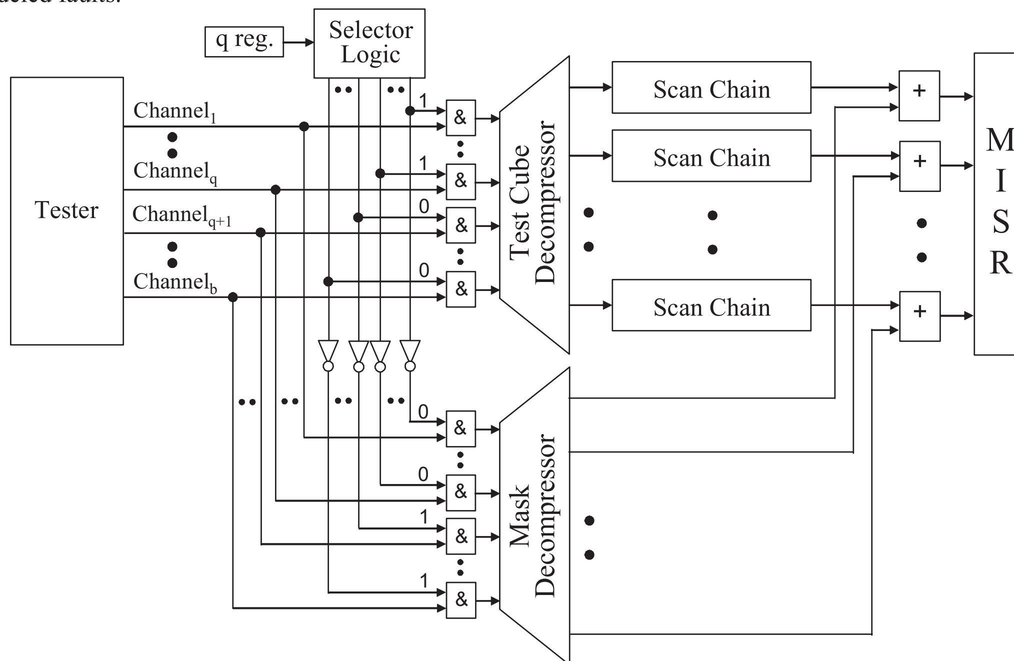

Figure 1. Proposed Dynamic Channel Allocation Scheme

Using this hardware, it is now possible to select the channel allocation when decompressing each test cube. The minimum number of channels needed to encode each test cube is determined by first estimating how many free variables will likely be needed to solve the linear equations based on the number of care bits in the test cube. The corresponding number of channels needed to get that number of free variables is then calculated. Then a linear equation solver is used to check if an encoding solution is possible for the estimated number of tester channels [Krishna 01], [Wang 06]. If so, then the number of channels is incrementally reduced as long as a solution can be found until the smallest number of channels that still produces a solution is identified. If the initial estimate of the number of channels was too low to begin with, then the number of channels is incrementally increased until a solution is found. Once the minimum number of channels needed to encoding the test cube is found, then that is the value of q that will be used when decompressing that test cube, and it will not be changed regardless of which output response is matched with this test cube. This allows the fully specified decompressed test vector to be computed and simulated to obtain the output response and the location of any unknown X's that are produced. These X values must be masked and hence create care bits in the mask control logic. Given the care bits in the mask control logic, the same process of using a linear equation solver as was used for the test cubes can be performed to find the minimum number of channels needed to encode the mask control data for each output response.

Given the number of channels used to encode each test cube, and the minimum number of channels needed to encode the mask control data for each output response, the order for applying the test cubes is selected. As mentioned in Sec. 1, the output response for the previous test cube is scanned out concurrently with the scan in of the current test cube. So the order in which the test cubes are applied determines which output responses are matched with which test cubes. The procedure for ordering the test cubes to minimize the total number of tester channels required (and hence maximize compression) is as follows.

For the very first test cube that is applied, the scan out is ignored, so there is no masking data required. The test cube requiring the maximum number of channels for decompression can be applied first. By the same token, no test cube is scanned in when the very last output response is scanned out, the test cube whose output response requires the most channels can be applied last. The rest of the test cubes are ordered such that test cubes requiring the most channels are matched with the output response requiring the fewest channels. A small example to illustrate the test cube ordering is shown in Fig. 2. There are 5 test cubes which are reordered to match the test cube and output response pairs in a way that minimizes the worst-case total number of channels. In this example, the worst case pair requires 16 tester channels to decompress.

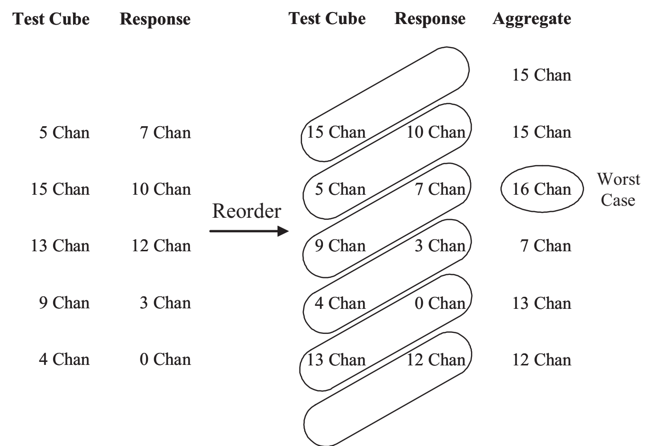

Figure 2. Example of Reordering Test Cubes to Minimize Worst Case Total Number of Tester Channels

Once the test cubes are ordered, the total number of channels needed to decompress each matched test cube and output response pair is computed by adding together the number of channels used for the test cube decompressor plus the minimum number of channels required for the mask decompressor. The maximum number of test channels required across all cases is the minimum number of tester channels that must be used when decompressing the test data for the CUT using this scheme.

3. Increasing Observability

As seen in the previous section, the total number of channels used depends on the worst-case across all matched test cubes and output response pairs. In many cases, the matched test cube and output response pair may require fewer channels. In those cases, more channels will be feeding the mask decompress than necessary creating excess free variables. In this section, a technique for exploiting these excess free variables to improve observability is described based on the idea in [Volkerink 05] of ANDing together stages of the decompressor.

If two stage of the decompressor are ANDed together, the number of care bits needed to mask each X increases to 2 because the decompressor needs to generate 1's on both inputs of the AND gate in order to mask each X. However, for the non-X values, the probability of each being masked and losing observability is reduced to 0.25. If a three-input AND is used, then 3 care bits are needed to mask an X, and the probability of losing observability for a non-X is reduced to 0.125.

The idea proposed here is to add a control register, m, which controls how many stages of the decompressor are ANDed together to drive the mask logic. This is illustrated in Fig. 3. The m register can be loaded at the same time as the q register in the first clock cycle for each test cube. In the example in Fig. 3, the m register is a two bit register which can take on 4 different values. This allows from 1 to 4 stages of the decompressor to be ANDed together to drive the mask logic. Depending on how many free variables are available in

the mask decompressor, the *m* register can be set to AND as many stages as possible while still being able to solve the linear equations to encode the mask data.

If *y* channels are needed to encode the minimum mask data with no ANDing of decompressor stages, then the number of channels needed to encode the mask dating with ANDing of *k* stages can be estimated as *(k)(y)* since the number of care bits in the mask control data increases by a factor of *k*.

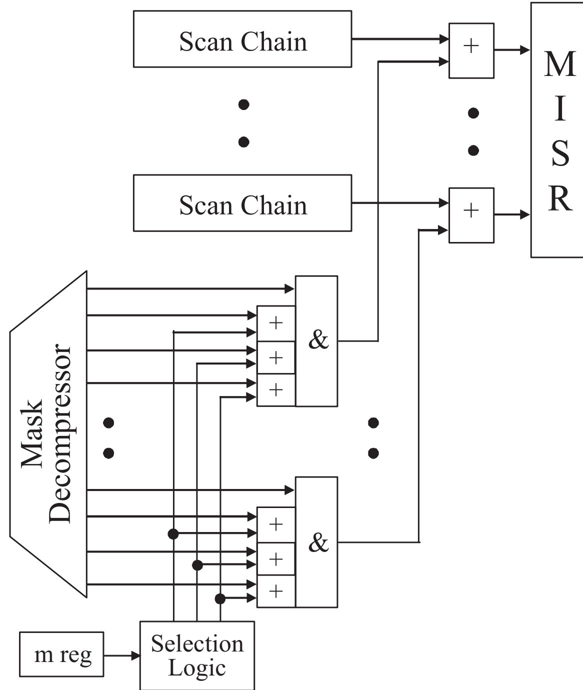

Figure 3. Scheme for Selective ANDing of Multiple Outputs of Mask Decompressor to Improve Observability

If the goal is to maximize the number of output responses that have *k>1*, then the procedure for ordering the test cubes is the following. Test cubes are selected in order from the one requiring the most channels to the one requiring the fewest channels. For each test cube, if all the remaining output responses that can be matched with it cannot have *k>1* without exceeding the total number of test channels available, then the output response with the most channels that can be matched without exceeding the total number of test channels is selected. This increases the chance that subsequent test cubes can be matched with *k>1*.

Note that if the number of channels from the tester is increased beyond the minimum needed to encode all masks for *k=1*, then more output responses can be encoded with *k>1* thereby increasing observability. The tradeoff between increasing the number of channels versus the improvements in observability are explored in the experimental results presented in the next section.

4. Experimental Results

Experiments were performed on two industrial circuits to evaluate the proposed method of using dynamic channel allocation with test cube ordering. In Table 1, results are shown for the two circuits. The number of scan cells are shown for each circuit, followed by results for using the conventional approach of having a fixed number of tester channels driving the test cube decompressor and a fixed number of tester channels driving the mask decompressor. The amount of test data after compression is shown followed by the observability which in the conventional case is 50% assuming the decompressor drives the mask logic with an equal distribution of 0's and 1's. Next, results are shown for the proposed method. By dynamically allocating the channels, the total number of channels needed is reduced resulting in more compression. Results are shown for three different levels of observability. The first line for each circuit, shows the case where compression is maximized as much as possible and what the corresponding observability is. The next two lines show where the number of channels is increased above the minimum to increase the number of excess free variables thereby allowing greater use of higher values of *k* to improve observability. As can be seen from the results, a significant improvement in both compression and observability is achieved with the proposed method.

The tradeoff between compression and observability is explored further in the graphs in Figs. 4 and 5 for *Ckt-A* and *Ckt-B*, respectively. These graphs show the number of test cubes with different values of *k* on the *y*-axis versus the increase in the number of tester channels over the minimum possible on the *x*-axis. As the number of tester channels are increased, the number of excess free variables increases allow more output responses to be shifted out with higher values of *k* thereby increasing observability of the non-*X* values.

Table 1. Experimental Results for Proposed Partial Masking in *X*-Chains for Different Designs

Circuit	Scan Chains	Fixed Channels		Proposed Dynamic Channel Allocation			
		Compressed Bits	Aggregate Observability	Compressed Bits	Percent Improve	Aggregate Observability	Percent Improve
Ckt-A	36,075	19.3M	50%	11.2M	42%	64%	28%
				13.4M	31%	70%	40%
				15.7M	19%	77%	54%
Ckt-B	505,051	97.7M	50%	65.8M	33%	79%	58%
				79.0M	19%	88%	76%
				92.1M	6%	93%	86%

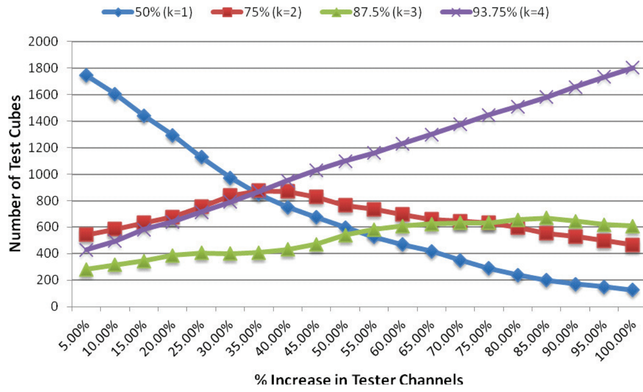

Figure 4. Plot of Observability Improvement Versus Number of Tester Channels for *Ckt-A*

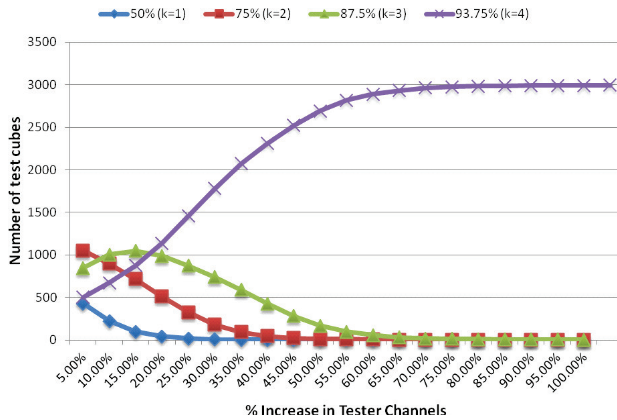

Figure 5. Plot of Observability Improvement Versus Number of Tester Channels for *Ckt-B*

As can be seen in Fig. 4 *Ckt-A* starts off with most test cubes masked with $k=1$ which then decreases almost linearly with more tester channels while the number of test cubes that can be masked with $k=4$ increases almost linearly. In Fig. 5, *Ckt-B* which has a lower X density, starts with most test cubes masked with $k>1$ and gives a much higher observability for a small increase in tester channels. As the tester channels keeps increasing, a point is reached where all the test cubes are masked with $k=4$.

5. Conclusions

Dynamic channel allocation between the test cube decompressor and output response decompressor can be implemented with a relatively small amount of logic (as illustrated in Fig. 1), but can have provide a significant boost in test compression. Moreover, higher observability of non-X values which is important for improving coverage of non-modeled faults can be achieved at no additional cost in terms of test data and relatively small additional hardware overhead (as illustrated in Fig. 3) with selective ANDing of the mask decompressor outputs.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant No. CCF-1217750

References

[Barnhart 01] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, and B. Koenemann, "OPMISR: the Foundation for Compressed ATPG Vectors", *Proc. of International Test Conference* , pp. 748-757, 2001.

[Chickermane 04] V. Chickermane, B. Foutz, and B. Keller, "Channel Masking Synthesis for Efficient On-Chip Test Compression", *Proc. of International Test Conference*, pp. 452-461, 2004.

[Czysz 10] D. Czysz, G. Mrugalski, N. Mukherjee, J. Rajski, and J. Tyszer, "On Compaction Utilizing Inter and Intra-Correlations of Unknown States," *IEEE Trans. on Computer-Aided Design*, Vol. 29, Issue 1, pp. 117-126, Jan. 2010.

[Könemann 91] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs," *Proc. European Test Conf.*, pp. 237-242, 1991.

[Krishna 01] C.V. Krishna, A. Jas, and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," *Proc. Int. Test Conf.*, pp. 885-893, 2001.

[Mrugalski 04] G. Mrugalski, J. Rajski, and J. Tyszer, "Ring Generators – New Devices for Embedded Test Applications", *IEEE Trans. on Computer-Aided Design*, Vol. 23, Issue 9, pp. 1306-1320, Sept. 2004.

[Naruse 03] M. Naruse, I. Pomeranz, S.M. Reddy, S. Kundu, "On-Chip Compression of Output Responses with Unknown Values Using LFSR Reseeding", *Proc. of International Test Conference*, pp. 1060-1068, 2003.

[Rajsi 02] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test," *Proc. of International Test Conference*, pp. 301-310, 2002.

[Volkerink 05] E.H. Volkerink, and S. Mitra, "Response Compaction with Any Number of Unknowns Using a New LFSR Architecture", *Proc. of Design Automation Conference*, pp. 117-122, 2005.

[Wang 06] L.T. Wang, C.-W. Wu, X. Wen, *VLSI Test Principles and Architectures*, Morgan Kaufmann, 2006.

[Wang 08a] S. Wang, K.J. Balakrishnan, and W. Wei, "X-Block: An Efficient LFSR Reseeding-Based Method to Block Unknowns for Temporal Compactors," *IEEE Trans. on Computers*, Vol. 57, No. 7, July 2008.

[Wang 08b] S. Wang and W. Wei, "An Efficient Unknown Blocking Scheme for Low Control Data Volume and High Observability," *IEEE Trans. on Computer-Aided Design*, Vol. 27, No. 11, Nov. 2008.

[Wohl 07] P. Wohl, J.A. Waicukauski, and S. Ramnath, "Fully X-Tolerant Combinational Scan Compression", *Proc. of International Test Conference*, Paper 6.1, 2007.

[Wohl 08] P. Wohl, J.A. Waicukauski, and F. Neuveux, "Increasing Scan Compression by Using X-Chains", *Proc. of Design Automation Conference*, Paper 35.1, 2008.

[Wohl 10] Wohl, P., J.A. Waicukauski, F. Neuveux, and E. Gizdarski, "Fully X-Tolerant, Very High Scan Compression," *Proc. of Design Automation Conference,* pp. 362-367, 2010.