Increasing Output Compaction in Presence of Unknowns using an X-Canceling MISR with Deterministic Observation

Ritesh Garg[1], Richard Putman[1,2], and Nur A. Touba[1]

[1]Computer Engineering Research Center
Dept. of Electrical and Computer Engineering
University of Texas, Austin, TX 78712
{rgarg2,touba}@ece.utexas.edu

[2]Cirrus Logic, Inc.
2901 Via Fortuna, Suite 100
Austin, TX 78746
Richard.Putman@cirrus.com

Abstract

Recently, an X-canceling MISR methodology was proposed in [Touba 07] which was based on providing very high probabilistic error coverage by canceling out X's in MISR signatures. This paper investigates a new methodology for using the X-canceling MISR architecture based on deterministically observing scan cells. The two main advantages of the proposed approach are (1) it can provide a higher amount of compaction, and (2) it is effective for larger percentages of X's in the output response. Also, this paper investigates a hybrid approach that combines X-masking with an X-canceling MISR. Experimental results indicate that significant amounts of output compression can be achieved with no loss of fault coverage.

1. Introduction

There are many sources of unknown *'X'* values that commonly arise during simulation, for example uninitialized memory elements, bus contention, floating tri-states, etc. Output response compaction in the presence of *X* values is a major issue for test compression and BIST. *X's* corrupt the final signature making it unknown. There are three basic approaches for handling *X's*. One is to do *X-bounding* which involves inserting design-for-testability (DFT) hardware into the CUT to prevent *X*'s from propagating to scan cells in the first place [Wang 06]. A second approach, which does not require modifying the CUT, is to use *X-masking*. This involves masking out the *X*'s at the input to the compactor. Mask data is required to specify which scan chain outputs should be masked in each clock cycle. A number of techniques have been developed for designing the masking hardware and compressing the amount of mask data that is required [Barnhart 01], [Wohl 01, 03, 04], [Pomeranz 02], [Naruse 03], [Chickermane 04], [Vokerink 05], [Chao 05], [Tang 06], [Rajski 06a]. A third approach is to use an *X-tolerant* compactor that can compact an output stream that contains *X's*. A number of *X-tolerant* compactors have been proposed including *X*-Compact [Mitra 04a], convolutional compactors [Rajski 05], low fanin compactors [Wohl 03b], *X*-MISR [Mitra 04b], *X*-Filter [Sharma 05], and modular compactors [Rajski 06b]. Recent work has also begun looking at ways for tolerating higher densities of *X's*. In [Wohl 07], selective *X*-masking is combined with a combinational compactor to tolerate higher *X* densities.

An *X*-canceling MISR methodology was proposed in [Touba 07] which is based on providing very high probabilistic error coverage by canceling out *X's* in MISR signatures. The error coverage can be made arbitrarily high and match that of using a conventional MISR to compact output responses without *X's*. This approach is highly efficient when the percentage of *X's* is low (e.g., 1% or less). It becomes less efficient for larger percentages of *X's*. This paper investigates two approaches for handling larger percentages of *X's* using an *X*-canceling MISR. The first is based on deterministically observing scan cells, and the second is based on using a hybrid approach that combines *X*-masking with an *X*-canceling MISR.

The first approach is a new methodology for using the *X*-canceling MISR architecture which is based on deterministically observing scan cells. It is effective for larger percentages of *X's* in the output response and can provide greater amounts of compression than probabilistic error detection. It can cancel out all *X's* and deterministically provide observation of any subset of non-*X* values. By having the automatic test pattern generator (ATPG) procedure record the subset of scan cells that must be observed to detect the necessary faults for a particular test pattern, the proposed method can then be used to deterministically observe those scan cells. By so doing, it can preserve the fault coverage of a test set in the presence of any distribution of *X's*.

The second approach described in this paper combines *X*-masking with an *X*-canceling MISR. The benefit of this hybrid approach is that the *X*-masking hardware can target only the easy to mask *X*'s and can let the rest of the *X*'s go through to the *X*-canceling MISR to be canceled there. This added flexibility allows for much greater compression of the masking control data without loss of fault coverage. One particular masking technique is proposed which can exploit the added flexibility.

1093-0167/08 $25.00 © 2008 IEEE
DOI 10.1109/VTS.2008.42

2. Overview of X-Canceling MISR

This section gives an overview of an *X*-canceling MISR and describes the new idea of how to use it for deterministic observation.

Consider the output response that has been captured in the scan chains after applying a test vector. Let the value in each scan cell be represented by a symbol (as illustrated in Fig. 1). Symbolic simulation can be performed to obtain the final state of the MISR in terms of the symbols after the output response has been shifted in to the MISR. Each bit of the MISR will be equal to a linear combination of the scan cells. This is shown in Fig. 1 where, for example, the final value of the top bit of the MISR will be equal to $X_1 \oplus O_3 \oplus D_8 \oplus O_{13}$.

In Fig. 1, assume each symbol X_i has an *X* value and each symbol D_i and O_i has a non-*X* value. Moreover, assume each symbol D_i corresponds to a scan cell that needs to be observed to ensure detection of the necessary faults for this particular test vector. In [Touba 07], only the *X* dependence was considered and all non-*X* values were observed probabilistically. In this work, the *D* dependence is also taken into consideration to ensure that the all *D*'s are deterministically observed.

Without loss of generality, assume all the O_i values in the output response are *0* so that each MISR bit is now simply equal to the linear combination of the *X* and *D* values. The *X* and *D* dependence of the MISR bits in this case are as shown in Fig. 2. The linear equations for each MISR bit can be represented as a matrix where each row corresponds to a MISR bit and each column corresponds to an *X* or *D*. Each entry in the matrix is a 1 if the MISR bit corresponding to the row depends on the *X* or *D* corresponding to the column. This is illustrated in Fig. 2. For example, in Fig. 2, the second row of the matrix corresponds to M_2, and the 1's in the first three columns indicate dependence on X_1, X_2, and X_3, respectively.

Gauss-Jordan elimination [Cullen 97] can be performed on the matrix in Fig. 2. Gauss-Jordan elimination involves performing rows operations that transform a set of columns into an identity matrix. Fig. 3 shows the matrix in Fig. 2 after Gauss-Jordan elimination has been performed. As was shown in [Touba 07], as long as the number of bits in the MISR is larger than the number of *X*'s compacted in the MISR, it is always possible to obtain rows after Gauss-Jordan elimination that have no dependence on the *X*'s. In Fig. 3, it can be seen that the last two rows have no dependence on the *X*'s. Looking at the last row, for example, if MISR bits M_3, M_4, and M_5 are XORed together, all the *X*'s cancel out and the resulting value will have no dependence on the *X*'s. This value can be compared with its fault-free value to detect errors in the non-*X* values that it depends on. Any combination of MISR bits can be XORed together using a programmable XOR as shown in Fig. 4. Since each XOR combination of MISR bits will depend on roughly half of the non-*X*

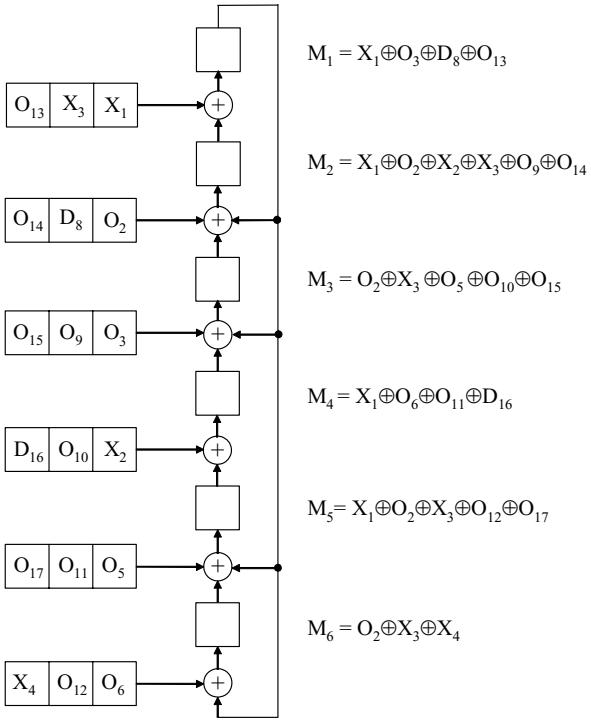

Figure 1. Example of Symbolic Simulation of MISR

$$
\begin{array}{l}
M_1 = X_1 \oplus D_8 \\
M_2 = X_1 \oplus X_2 \oplus X_3 \\
M_3 = X_3 \\
M_4 = X_1 \oplus D_{16} \\
M_5 = X_1 \oplus X_3 \\
M_6 = X_3 \oplus X_4
\end{array}
\rightarrow
\begin{array}{c}
\begin{array}{cccc:cc} X_1 & X_2 & X_3 & X_4 & D_8 & D_{16} \end{array} \\
\left[\begin{array}{cccc:cc}
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0
\end{array}\right]
\begin{array}{l} M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \end{array}
\end{array}
$$

Figure 2. Linear Equations for MISR in Fig. 1

$$
\begin{array}{c}
\begin{array}{cccc:cc} X_1 & X_2 & X_3 & X_4 & D_8 & D_{16} \end{array} \\
\left[\begin{array}{cccc:cc}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{array}\right]
\begin{array}{l} M_3 \oplus M_5 \\ M_2 \oplus M_5 \\ M_3 \\ M_3 \oplus M_6 \\ M_1 \oplus M_3 \oplus M_5 \\ M_3 \oplus M_4 \oplus M_5 \end{array}
\end{array}
$$

Figure 3. Gauss-Jordan Reduction of MISR Equations

values, it was shown in [Touba 07] that checking q such combinations of MISR bits would give an error coverage approximately equal to $1\text{-}2^{-q}$. By simply checking a sufficient number of X-canceled combinations of MISR bits, a high probabilistic error coverage could be obtained. For example, by checking 7 X-canceled combinations, over 99% error coverage can be obtained. This is very efficient when the percentage of X's is around 1% or less. However, it becomes less efficient for larger percentages of X's.

The idea in this paper is that rather than checking a larger number of combinations to ensure a high probabilistic error coverage of all non-X values, a deterministic procedure could be used to ensure that the necessary to observe values (i.e., the D's) are checked in a small number of combinations. Since checking each combination using the architecture in Fig. 4 requires m bits be supplied by the tester where m is the number of bits in the MISR, if the number of combinations that need to be checked can be reduced, then the data stored on the tester can be reduced resulting in greater test data compression.

To observe the D's deterministically, it is necessary to include them in the dependency matrix as is done in Fig. 2. Then when Gauss-Jordan elimination is performed, it not only processes the X columns, but also the D columns so that there is a single 1 in every row and column as shown in Fig. 3. The number of X's and D's compacted should be limited so that it is always possible to obtain such a matrix after Gauss-Jordan elimination. Generally that means that each MISR signature can compact up to a total number of X's plus D's equal to m, the size of the MISR, so that the number of columns does not exceed the number of rows in the dependency matrix. After Gauss-Jordan elimination, the rows that depend only on the D's (i.e., not on the X's) are all XORed together to form one MISR bit combination that depends on all the D's. In the example in Fig. 3, this means that the last two rows are XORed together resulting in the MISR bit combination equal to:

$$(M_1 \oplus M_3 \oplus M_5) \oplus (M_3 \oplus M_4 \oplus M_5) = M_1 \oplus M_4$$

The MISR bit combination $M_1 \oplus M_4$ will not depend on any X's, but will depend on all the D's. This can be seen by looking back at Fig. 1, and computing:

$$M_1 \oplus M_4 = (X_1 \oplus O_3 \oplus D_8 \oplus O_{13}) \oplus (X_1 \oplus O_6 \oplus O_{11} \oplus D_{16})$$
$$= O_3 \oplus O_6 \oplus D_8 \oplus O_{11} \oplus O_{13} \oplus D_{16}$$

As long as the number of X's plus the number of D's compacted by the MISR is less than or equal to m, and the Gauss-Jordan elimination produces a single 1 in every row and column, it is always possible to find a MISR bit combination that will depend on all the D's.

Checking a MISR bit combination that depends on all the D's still does not ensure that all errors in the D's will be detected. If an even number of D's have errors, the errors will cancel out in the MISR bit combination and not be detected. So it is still necessary to check more than one combination. So then the question becomes what is the advantage of deterministically considering the D's versus just using probabilistic error coverage as in [Touba 07]. When using just a single large MISR, there may not be much advantage to doing this. However, as will be shown in the next section, if the large MISR is replaced by multiple smaller MISRs, then deterministically considering the D's can provide a significant advantage over probabilistic error coverage for equivalent aggregate MISR sizes. One specific scheme is described in Sec. 4 which uses a total of 16 MISRs each of size 16 bits with deterministic consideration of the D's, and it is shown to provide almost a factor of 3 better compression in comparison to using a single MISR of size 256 with probabilistic error coverage.

3. Using Multiple MISRs

As was shown in the previous section, even with deterministic consideration of the D's, it is not sufficient to observe only one MISR bit combination because an even number of errors may cancel out. Assuming all error combinations are equally likely, then the error coverage for the D's would only be 50%. To reduce the probability of error canceling, it is necessary to observe a larger

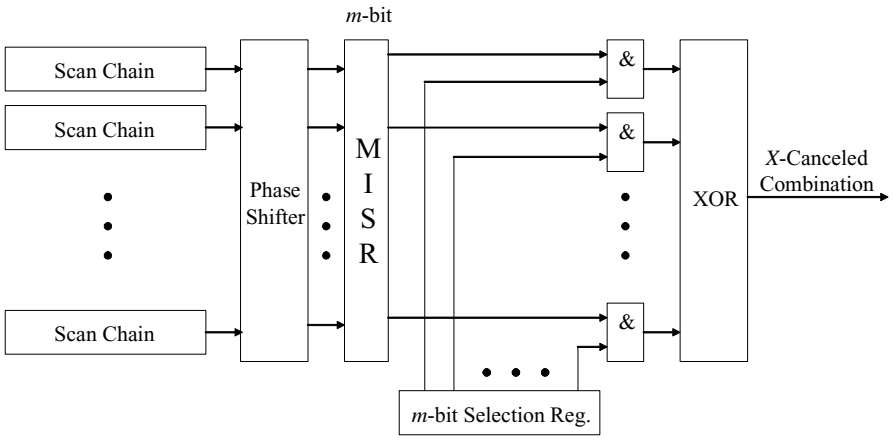

Figure 4. X-Canceling MISR

number of MISR bit combinations. However, if only a single large *m*-bit MISR is used, then checking each combination requires *m* bits to be stored on the tester. If two combinations are checked where each combination depends on 50% of the *D*'s, then the error coverage for the *D*'s would be 75% (the probability that there is an even number of errors in each combination is 0.5, so the probability that both have even errors is 0.25). So with a single *m*-bit MISR, obtaining 75% coverage of the *D*'s requires *2m* bits be stored on the tester. Now consider replacing the single *m*-bit MISR with two *m/2*-bit MISRs each of which compact half of the scan chains. For each of the two *m/2*-bit MISRs, a combination of MISR bits can be found using the procedure described in Sec. 2 such that it depends on all the *D*'s captured by that MISR. If half the *D*'s propagate to each MISR, then by checking one MISR bit combination for each of the *m/2*-bit MISRs, the same error coverage for the *D*'s (i.e., 75%) can be obtained as checking two MISR bit combinations of the *m*-bit MISR. However, checking each of the combinations for the *m/2*-bit MISRs requires only *m/2* bits, so in this case 75% coverage of the *D*'s is obtained with storing only *m* bits on the tester which is a factor of 2 improvement.

There are two drawbacks to using the two *m/2*-bit versus using one *m*-bit MISR:

1. The compaction must stop when either of the two MISRs captures a total number of *X*'s plus *D*'s equal to *m/2*. If the *X*'s and *D*'s are exactly evenly distributed between the two *m/2*-bit MISRs, then this would still happen at the same point as when a single *m*-bit MISR would capture a total number of *X*'s plus *D*'s equal to *m*. However, in reality, it is unlikely that the *X*'s and *D*'s would be exactly evenly distributed, so in general one of the *m/2*-bit MISRs would fill up sooner than the other and thus the total number of *X*'s plus *D*'s compacted for each signature with two *m/2*-bit MISRs would be less than it would for a single *m*-bit MISR.
2. If the *D*'s are not evenly distributed between the two *m/2*-bit MISRs, then the error coverage will be lower than for a single *m*-bit MISR.

One approach to mitigate both of these drawbacks would be to use two pairs of *m/2*-bit MISRs (4 MISRs altogether) in the following way. Divide the scan chains into four evenly sized groups G_1, G_2, G_3, and G_4. Have one MISR capture from G_1 and G_2 and another MISR capture from G_3 and G_4. Then for the other pair of MISRs, have one capture from G_1 and G_3 and the other from G_2 and G_4. For each signature, there is now a choice of using one or the other pair of MISRs. If one pair of MISR is highly skewed in the distribution of *X*'s or *D*'s, then the other pair can be used. This helps to smooth out the distributions so that the performance of the *m*/2-bit MISRs in terms of error coverage and number of scan slices compacted per signature will not significantly lag that of an *m*-bit MISR while still enjoying a factor of 2 reduction in storage requirements on the tester. The cost of using two pairs of *m/2*-bit MISRs versus only a single pair is of course the additional overhead of adding more MISRs plus one extra bit needs to be stored on the tester per signature to dynamically select which pair of MISRs to use for each signature.

This approach of replacing an *m*-bit MISR with two pairs of *m/2*-bit MISRs can be used recursively. Each of the four *m/2*-bit MISR could be replaced by two pairs of *m/4*-bit MISRs. The total number of MISRs would now be 16. A single signature in the *m*-bit MISR is now replaced with signatures from 4 of the *m/4*-bit MISRs. This would give an error coverage for the *D*'s close to that of checking 4 combinations in the *m*-bit MISR which is ($1-2^{-4} = 93.75\%$), and a reduction in tester storage of almost a factor of 4. Note that now 3 bits must be stored on the tester per signature to select which pairs of *m/4*-bit MISRs are used.

4. Multiple MISR Design Example

Using this approach of splitting a larger MISR into multiple smaller MISRs as described in Sec. 3, one particular design example is described here.

Consider the design example shown in Fig. 5. A single 64-bit MISR is replaced with 16 MISRs each of size 16. The 16-bit MISRs compact output response data scan slice by scan slice until a point is reached where compacting an additional slice would make it no longer possible to solve for all the *D*'s using signatures from 4 of the MISRs. At this point, 4 of the MISR signatures are processed. Gauss-Jordan reduction is performed on the linear equations for those MISRs as described in Sec. 2. In order to ensure high error coverage of the *D*'s, two linear combinations are checked for each MISR signature. Each of the two linear combinations per signature are formed by XORing together half of the rows in the Gauss-Jordan reduced matrix that depend only on *D*'s. This evenly divides the *D*'s between the two combinations. The end result is that 8 linear combinations are checked (two combinations for each of the 4 MISRs). Every *D* is included in exactly one of the linearly dependent combinations. The best case is if the *D*'s are evenly distributed among all 4 MISRs. In that case, the error coverage for the *D*'s where all error combinations are equally likely would be approximately $1-2^{-8}=99.6\%$. All odd errors in the *D*'s would be guaranteed to be detected since at least one of the 8 combinations must have an odd number of errors. The error coverage for all other non-*X* bits would be $1-2^{-2}=75\%$. In general, the error coverage would be slightly lower due to variance in the distribution of *D*'s. Experiments indicate the average error coverage for the *D*'s is greater than 98% and as high as 99.5%.

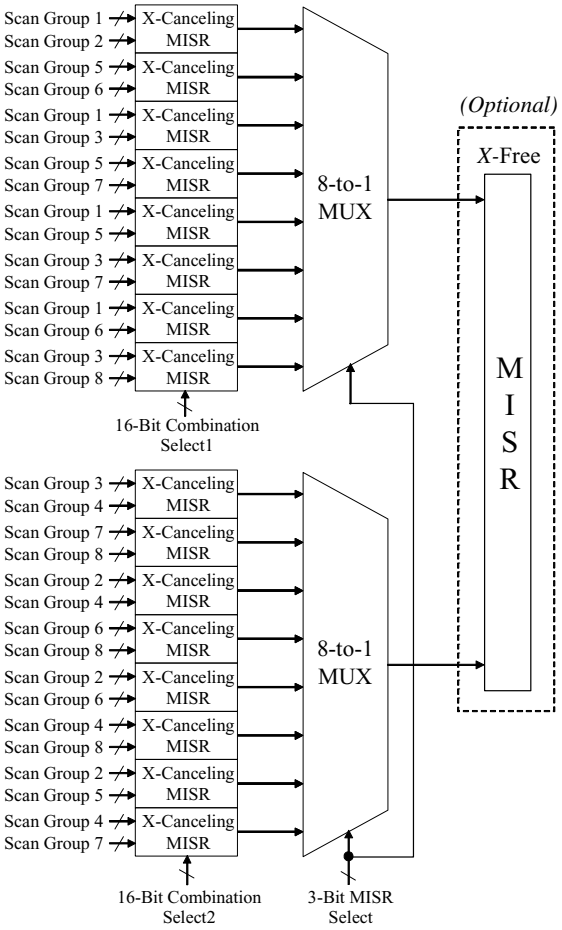

Figure 5. 16 *X*-Canceling MISRs Each of Size 16-Bits Compacting Scan Chains Divided Evenly into 8 Groups

The storage requirements for using a single 64-bit MISR checking 8 combinations would be (8*64)=512 bits per signature, and each signature could compact a total of 64 *X*'s plus *D*'s. So the cost is 512/64=8 bits of storage per *X* and per *D*. Going to a 256-bit MISR would also still require 8 bits per *X* and per *D*. The probabilistic approach in [Touba 07] would require 8 bits of storage per *X* (it doesn't depend on *D*'s) for 99.6% error coverage, so it would actually be better. Now compare this to using the multiple MISR design in Fig. 5. For the multiple MISR design, each signature can compact 64 *X*'s plus *D*'s, and requires generating 2 linear combinations of four 16-bit MISRs where each linear combination requires 3-bits to select which MISR and 16-bits to select the combination. So the storage requirement per signature as shown in the figure is 4*(16+16+3) = 140 bits. The cost is 140/64 =2.1875 bits of storage per *X* and per *D*. Consider a test set with 1% *X*'s and 2% *D*'s, in this case the method in [Touba 07] would require around 8*1%=8% whereas the proposed method would require around (2.1875)(1%+2%) =6.56%. However, now consider a design with 3% *X*'s and 2% *D*'s, in this case the method in [Touba 07] would require around 8*3%=24% whereas the proposed method would require around (2.1875)(3%+2%)=10.9375%. So the bottom line is that for low percentages of *X*'s, the method in [Touba 07] is very efficient. However, for designs with larger percentages of *X*'s, the proposed multiple MISR method is much more efficient.

5. Combining X-Masking and X-Canceling

To further improve the output compression achieved by an *X*-canceling MISR on designs with larger numbers of *X*'s, a hybrid approach that combines *X*-masking with an *X*-canceling MISR can be used. *X*-masking circuitry is added between the outputs of the scan chains and the inputs to the *X*-canceling MISR. The purpose of the *X*-masking circuitry, as well as the ensuing methodology, is to mask as many *X*'s as possible with the smallest amount of mask and control data, such that the overall amount of data stored on the ATE used to mask all of the *X*'s is less than it would have been if the *X*-canceling MISR was used by itself.

The key advantage of using *X*-masking plus *X*-canceling versus conventional *X*-masking only approaches that is exploited here is that the same mask can be reused for many scan slices since it is not necessary to mask all *X*'s. This is taken advantage of by using the *X*-masking architecture proposed in Fig. 6.

For *m* scan chains, the architecture in Fig. 6 consists of a control signal, an *m*-bit masking register, an interval counter, and two logic gates per internal scan chain for a total of *2m* logic gates. The control signal is used to determine whether or not to apply the mask on a per scan slice basis. If the control signal is a '1', then the mask is applied. If the control signal is a '0', then the mask data itself is blocked and cannot affect the scan output response data. The mask register holds the mask data, which can be applied to the current scan slice. If the control signal is a '1', and a given mask data bit is also a '1', then the corresponding output response data bit is masked and forced to be a '1'. A '0' in the mask data means no masking will occur on the corresponding output response data bit even if the mask control signal is '1'.

The interval counter counts down the number of shift cycles (i.e., scan slices) the current mask can be applied to before a new mask is loaded. A constant value representing that number of scan slices is loaded into the interval counter's control logic one time at the beginning of the scan test. Every time the interval counter hits zero, including when it is reset at the beginning of the scan test, a new mask is loaded into

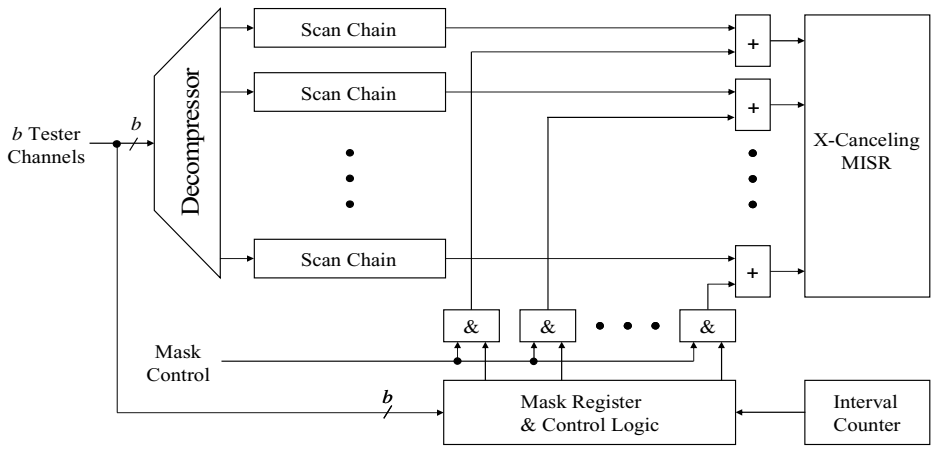

Figure 6. Proposed *X*-Masking Architecture for Use with *X*-Canceling MISR

the mask register, and the interval counter is loaded with its preprogrammed value. If there are *m* internal scan chains requiring *m* bits of mask data and *b* tester channels, then it will take *m/b* clock cycles to fully load the mask data at the beginning of each interval.

The goal of creating each mask is to make it applicable to as many adjacent scan slices as possible and mask as many *X*'s in those scan slices as possible, without masking out any of the *D*'s for which observation must be ensured. Also, it is desirable to minimize the number of non-*X* values that are masked as they may be useful for detecting unmodeled faults.

The mask is created by first determining the locations of the *D* and *X* bits for each scan slice. Next, an interval of *n* number of scan slices is chosen to be processed for each new mask. Then an optimal mask is created for each interval by determining which scan slices should not be masked, as well as which scan slice bit position should be masked.

The algorithm for creating all of the masks for the entire pattern set, along with an example illustrated in Fig. 7, is as follows:

1) Select the next interval of *n* scan slices to process. In the example in Fig. 7, the interval contains five scan slices (shown as rows numbered 1 to 5).
2) All mask bits corresponding to scan chains in the current interval that contain *X*'s and no *D*'s are set to '1' indicating that those scan chains will be masked. In Fig. 7, scan chain 6 is the only scan chain that this applies to, so it's mask bit is set to a '1'.
3) For each scan chain containing both *X*'s and *D*'s, the only way to mask the scan chain is if the mask control bit for the scan slices containing the *D*'s is set to 0. Setting the mask control bit for a scan slice to 0 prevents masking any *X*'s in that scan slice. A benefit function can be computed as the number of *X*'s that can be masked in the scan chain minus the number of *X*'s that cannot be masked for each of the scan slices containing *D*'s for which the mask control bit would need to be set to 0. For example, for scan chain 4, it has three *X*'s and one *D*. The one *D* resides in scan slice 2 which contains one *X*. So the benefit function for scan chain 4 would be 3-1=2. The scan chain with largest positive benefit function is selected first. It's mask bit is set to '1' and the mask control bit for each scan slice containing a *D* in that scan chain is set to '0'. This is repeated in a greedy fashion until no more scan chains exist whose benefit function is positive. In the example in Fig. 7, there are two scan chains that have both *X*'s and *D*'s, namely scan chains 4 and 5. Their initial benefit functions are 2 and 0 respectively. So only scan chain 4 has its mask bit set to '1' and the mask control bit for scan slice 2 is set to '0' to prevent masking the *D* in scan chain 4.
4) When the greedy mask selection procedure in step 3 completes, then all the beneficial scan chain mask bits have been set to '1' and all the necessary mask control bits have been set to '0'. The procedure can now move to the next interval of *n* scan slices.

Scan Slice	Mask Cntrl.	Scan Chains 0	1	2	3	4	5	6	7	8	9
1	1	s	d	s	s	**x**	d	**s**	s	s	s
2	0	s	s	s	s	d	s	x	s	s	s
3	1	s	s	s	s	**x**	d	**x**	s	s	s
4	1	s	d	s	s	**s**	x	**s**	s	s	s
5	1	s	d	s	s	**x**	s	**x**	s	s	s
	Mask	0	0	0	0	1	0	1	0	0	0

(a)

s	d	s	s	s	d	s	s	s	s
s	s	s	s	d	s	**x**	s	s	s
s	s	s	s	s	d	s	s	s	s
s	d	s	s	s	**x**	s	s	s	s
s	d	s	s	s	s	s	s	s	s

(b)

Figure 7. (a) Example of Mask Data for Output Response Interval; (b) Output Response after Masking

For the example output response in Fig. 7, the algorithm only masks scan chains 4 and 6 and disables masking of scan slice 2. All but two of the *X*'s are masked. The resulting masked output response data (which is shown in Fig. 7b) is fed into the *X*-canceling MISR.

By being able to reuse a carefully created mask for multiple scan slices, without masking any *D*'s and masking out a large percentage of *X*'s from the output response data, the proposed *X*-masking approach, when combined with a downstream *X*-canceling MISR, can provide a significant improvement in output response data compaction, especially in the presence of a large number of unknowns.

6. Experimental Results

Experiments were performed using the compactor design in Fig. 5 with 16 *X*-canceling MISRs each of size 16 for output streams with different percentages of *X*'s and *D*'s. The results are shown in Table 1. The first column shows the percentage of *X*'s in the output stream. The remaining columns show the various compressions achieved for different percentages of *D*'s corresponding to the different *X* percentages. Experimental results substantiate that the number of bits stored on the tester is approximately 2.1875 bits for each *X* or *D*, as described in Sec. 4. Thus, the amount of compression reduces as the number of *X*'s plus *D*'s increases.

To obtain higher compression, *X*-masking can be combined with *X*-canceling as described in Sec. 5. Experiments were performed on 3 industrial designs from Cirrus Logic. SynTest's ATPG tool was used to generate the tests and report a scan cell that each necessary fault propagated an error to. These scan cells were marked as *D*'s in the output response. Table 2 reports the results. The second and third column shows the percentage of *X*'s and *D*'s present in the corresponding designs. Experiments were performed for 3 different numbers of scan chains which are shown in the fourth column. The output response of all the three designs was compacted with and without using the *X*-masking before *X*-canceling, and the results are tabulated. As shown in the fifth column, a large percentage of *X*'s are masked from all the designs using the *X*-masking technique discussed in Sec. 5. The last three columns show the compression ratio achieved and the percentage improvement when using X-masking prior to X-canceling as compared to X-canceling alone. All the control and masking data needed to support the *X*-masking (as described in Sec. 5) is factored into the compression numbers.

Design *B* has the smallest number of *X*'s. As the number of *X*'s get smaller, the control data for *X*-masking starts to dominate the cost and this is seen from the minimum percentage improvement for Design *B* as compared to the other two designs. Furthermore, this dominance is exacerbated by the fact that although more *X*'s are able to be masked out for smaller numbers of scan chains, much more control data is required and hence there is a smaller percentage improvement.

Table 1. Compression for Different Percentages of *X*'s and *D*'s using Design in Fig. 5

% of *X*'s	% of *D*'s			
	1%	2%	4%	6%
1%	21.3x	14.5x	9.0x	6.6x
3%	11.0x	9.0x	6.6x	5.3x
5%	7.6x	6.6x	5.3x	4.5x
8%	5.3x	4.8x	4.2x	3.7x
10%	4.5x	4.2x	3.7x	3.3x

Table 2. Results for Combining *X*-Masking and *X*-Canceling

Design	Percentage *X*'s	Percentage *D*'s	Scan Chains	Percent X's Masked using *X*-Masking	Compression *X*-Canceling Alone	Compression *X*-Masking and *X*-Canceling	Percentage Improvement
A	5.37%	0.89%	64	83%	5.9x	14.3x	58.6%
			128	81%	6.5x	15.7x	58.9%
			256	78%	6.6x	15.9x	58.6%
B	2.58%	0.63%	64	83%	9.1x	21.0x	56.4%
			128	75%	10.7x	22.3x	51.8%
			256	71%	11.2x	23.6x	52.4%
C	8.33%	0.75%	64	83%	4.4x	11.6x	62.1%
			128	78%	4.8x	12.2x	60.2%
			256	77%	5.0x	12.7x	60.3%

7. Conclusions

Using an *X*-canceling MISR with probabilistic observation provides high error coverage for all errors, but it becomes less efficient for larger percentages of *X*'s. This paper showed how deterministic observation can be used to achieve greater compression for higher percentages of *X*'s. It was shown that by dividing a large *X*-canceling MISR into multiple smaller *X*-canceling MISRs, the number of bits required to process each signature could be reduced while still observing the necessary to observe bits (i.e., the *D*'s).

It was also shown that a hybrid approach combining *X*-masking with an *X*-canceling MISR can significantly increase the amount of compression. Large numbers of *X*'s can be masked at low cost by focusing only on the easy to mask *X*'s (where the mask data can be reused across many scan slices) and leaving the rest of the *X*'s to be handled by the *X*-canceling MISR.

References

[Barnhart 01] Barnhart, C., 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.

[Brglez 89] Brglez, F., D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. of International Symposium on Circuits and Systems*, pp. 1929-1934, 1989.

[Chao 05] M. C.-T. Chao, S. Wang, S.T. Chakradhar, and K.-T. Cheng, "Response Shaper: A Novel Technique to Enhance Unknown Tolerance for Output Response Compaction," *Proc. of International Conference on Computer-Aided Design*, pp. 80-87, 2005.

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

[Cullen 97] Cullen, C.G., *Linear Algebra with Applications*, Addison-Wesley, ISBN 0-673-99386-8, 1997.

[Mitra 04a] Mitra, S., and K.S. Kim, "X-Compact: An Efficient Response Compaction Scheme," *IEEE Trans. on Computer-Aided Design*, Vol. 23, No. 3, pp. 421-432, Mar. 2004.

[Mitra 04b] Mitra, S., S.S. Lumetta, and M. Mitzenmacher, "X-Tolerant Signature Analysis," *Proc. of International Test Conference*, pp. 432-441, 2004.

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

[Patel 03] Patel, J.H., S.S. Lumetta, and S.M. Reddy, "Application of Saluja-Karpovsky Compactors to Test Responses with Many Unknowns," *Proc. of VLSI Test Symposium*, pp. 107-112, 2003.

[Pomeranz 02] Pomeranz, I., S. Kundu, and S.M. Reddy, "On Output Response Compression in the Presense of Unknown Output Values," *Proc. of Design Automation Conference*, pp. 255-258, 2002.

[Rajski 00] Rajski, J., N. Tamarapalli, and J. Tyszer, "Automated Synthesis of Phase Shifters for Built-In Self-Test Applications," IEEE Trans. on Computer-Aided Design, Vol. 19, No. 10, pp. 1175-1188, Oct. 2000.

[Rajski 05] Rajski, J., J. Tyszer, C. Wang, and S.M. Reddy, "Finite Memory Test Response Compactors for Embedded Test Applications," *IEEE Trans. on Computer-Aided Design*, Vol. 24, No. 4, pp. 622-634, Apr. 2005.

[Rajski 06a] Rajski, J., J. Tyszer, G. Mrugalski, W.-T. Cheng, N. Mukherjee, and M. Kassab, "X-Press Compactor for 1000x Reduction of Test Data," *Proc. of International Test Conference*, Paper 18.1, 2006.

[Rajski 06b] Rajski, W., and J. Rajski, "Modular Compactor of Test Responses," *Proc. of VLSI Test Symposium*, pp. 242-251, 2006.

[Saluja 83] Saluja, K.K., and M. Karpovsky, "Testing Computer Hardware Through Test Data Compression in Space and Time," *Proc. of International Test Conference*, pp. 83-89, 1983.

[Sharma 05] Sharma M. and W.-T. Cheng, "X-Filter: Filtering Unknowns from Compacted Test Responses," *Proc. of International Test Conference*, Paper 42.1, 2005.

[Tang 06] Tang, Y., H.-J. Wunderlich, P. Engelke, I. Polian, B. Becker, J. Scholöffel, F. Hapke, and M. Wittke, "X-Masking During Logic BIST and Its Impact on Defect Coverage," *IEEE Trans. on VLSI*, Vol. 14, No. 2, Feb. 2006.

[Touba 07] Touba, N.A., "X-Canceling MISR – An X-Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR," *Proc. of International Test Conference*, Paper 6.2, 2007.

[Volkerink 05] Volkerink, E.H., 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.

[Wohl 01] Wohl, P., J.A. Waicukauski, and T.W. Williams, "Design of Compactors for Signature-Analyzers in Built-In Self-Test," *Proc. of International Test Conference*, pp. 54-63, 2001.

[Wohl 03a] Wohl, P., J.A. Waicukauski, S. Patel, and M.B. Amin, "X-Tolerant Compression and Application of Scan-ATPG Patterns in a BIST Architecture," *Proc. of International Test Conference*, pp. 727-736, 2003.

[Wohl 03b] Wohl, P., L. Huisman, "Analysis and Design of Optimal Combinational Compactors," *Proc. of VLSI Test Symposium*, pp. 101-106, 2003.

[Wohl 04] Wohl, P., J.A. Waicukauski, and S. Patel, "Scalable Selector Architecture for X-Tolerant Deterministic BIST," *Proc. of Design Automation Conference*, pp. 934-939, 2004.

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