On Compacting Test Response Data Containing Unknown Values

Chen Wang* Sudhakar M. Reddy** Irith Pomeranz*** Janusz Rajski* Jerzy Tyszer****

* Mentor Graphic Corporation, Wilsonville, OR 97070
** Elec. & Comp. Eng. Department, University of Iowa, Iowa City, IA 52242
*** School of Elec. & Comp. Eng. Purdue University, West Lafayette, IN 47907
**** Poznan University of Technology, ul. Piotrowo 3a, 60-965 Poznan, Poland

Abstract

The design of a test response compactor called a *Block Compactor* is given. Block Compactors belong to a new class of compactors called *Finite Memory Compactors*. Different from space compactors, finite memory compactors contain memory elements. Also unlike time compactors, finite memory compactors have finite impulse response. These properties give finite memory compactors the ability to achieve higher compaction ratios than space compactors and still be able to tolerate unknown values in test responses. The proposed Block Compactors, as an instance of finite memory compactors generate a signature of response data in several scan cycles. Results presented on several industrial designs show that Block Compactors provide better test quality and higher data compaction than earlier works on test response compactors.

1. Introduction

The major components of the cost of manufacturing test for VLSI circuits are test application time and tester storage for test patterns and test responses. Methods to reduce test application time include generation of compact tests [1, 2] and the use of multiple scan chains [3]. Methods to reduce tester storage requirements include methods to reduce test input data volume [4-11] and methods to reduce test response data volume [3, 6, 12-17]. Some of these methods [6, 13, 14,16, 17] address unknown values in the output response of the circuit.

Test response data compaction can be achieved using combinational compactors (also called space compactors) [12] and/or sequential compactors [13], which are typically multiple input signature registers and are also called time compactors. Time compactors have infinite impulse response property. Space compactors typically use linear circuits comprising of exclusive OR (XOR) gates. Figure 1 illustrates the use of a space compactor to compress test responses from a design with multiple scan chains. The number of inputs to the compactor is N, which is also the number of scan chains, and the number of outputs is Z, with Z << N. The test response data is compacted by a factor of N/Z, which is referred to as the *compaction ratio*. Because of data compaction some faults detected by a given test set may not be detected by observing the compactor outputs. This occurs when all the errors in the test response of a fault are masked by the compactor. This is referred to as aliasing. An ideal compactor has zero aliasing.

Work of S. M. Reddy supported in part by NSF Grant No. CCR-0097005 and by SRC Grant No. 2001-TJ-949. The work of I. Pomeranz supported in part by NSF Grant No. CCR-0098091 and by SRC Grant No. 2001-TJ-950.

A method to design space compactors with low aliasing probability using linear block code parity check matrices were proposed in [12]. This method assumed that the test responses do not contain any unknown values (represented by X in this work). Mitra and Kim proposed a method called X-Compact [16] that uses compactors also based on parity check matrices and achieves compaction of test response data containing Xs. This method uses parity check matrices where all the columns have the same weight (the weight of a column is the number of ones in it). A method called I-compact that uses parity check matrices of a linear block code (e.g., Hamming codes) to achieve compaction of test response data with Xs was recently proposed in [17]. This method uses results from coding theory on error detection in the presence of erasures. Using parity check matrices with equal weight columns, as in X-compact, is better suited for current architectures of high performance testers. Such testers employ independent processors to make detection decisions based on data from each compactor output independent of the data on other outputs. On the other hand, the errors and erasure based decisions in I-compact require consideration of the data on all the compactor outputs together.

In this work we propose a compactor called *Block Compactor* to compact test response data with Xs. A Block Compactor is neither a space compactor nor a time compactor. It belongs to a new class of compactors called *finite memory compactors*. These new type of compactors contain memory elements but have finite impulse response. The design of the proposed Block Compactors is also based on the parity check matrices of block codes. Another type of finite memory compactors called *Convolutional Compactors* based on the parity check matrices of convolutional codes is described in [18]. It will be demonstrated that the proposed compactor achieves much higher levels of compaction than X-compact and I-compact, which also use parity check

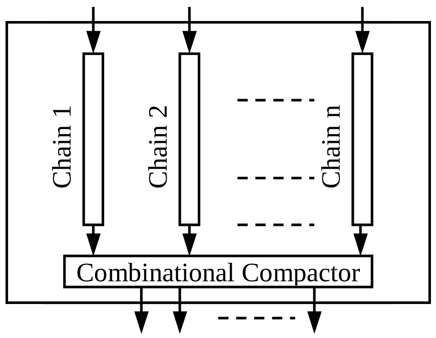

Figure 1: Use of a combinational compactor

matrices to design space compactors. Additionally, we consider several issues related to the design of compactors that achieve high levels of compaction with negligible aliasing probabilities.

Even though the proposed method can utilize the parity check matrices of any block code, in this paper we consider parity check matrices that contain columns of equal weight. The designs we consider in this paper are scan designs. However the method can be applied to non-scan designs also.

The remainder of the paper is organized as follows. In Section 2 a brief review of the design of compactors based on parity check matrices of block codes is given. In Section 3 the design of the basic version of the proposed compactor is given together with analysis that will lead to the design of compactors that simultaneously achieve high levels of compaction and low aliasing probability. In Section 4 we describe additional techniques to reduce the logic required and to reduce the aliasing probability of the compactor. In Section 5 experimental results on several industrial circuits are given. Section 6 concludes the paper.

2. Preliminaries

The functionality of a compactor is to compact test responses into small signatures. In a space compactor such as X-Compact [16] or I-Compact [17], compaction is achieved by mapping the test response data in each scan shift cycle into a signature. All the bits of the signature are observed at the compactor outputs at each cycle. A well-designed space compactor can tolerate a certain number of unknown values (Xs) and support fault detection and diagnosis. The achievable compaction ratio is determined by the method used and the desired tolerance to Xs.

In the following, we briefly review the basic idea behind the use of parity check matrices of block codes to design space compactors. This idea, originally suggested by Saluja and Karpovsky [12], is common to the compactor designs proposed in [12,16,17]. Consider the parity check matrix H given in Figure 2. H has four rows and six columns. It defines a space compactor with six inputs and four outputs. The compactor achieves a compaction ratio of 6/4 = 1.5. The compactor obtained by using H is shown in Figure 2(b). It can be seen that there is a compactor output corresponding to each row of the parity check matrix. The ith output Z_i is obtained by the binary sum (XOR) of the inputs corresponding to the columns that have a 1 in the ith row. For example $Z_1= I_1 \oplus I_2 \oplus I_3$, $Z_2= I_1 \oplus I_4 \oplus I_5$ and so on. Consider the case of a scan cycle where the test response data of the fault-free circuit is (001X00) and assume that for some faulty circuit the value of I_2 in the test response is in error (i.e., instead of it being zero, it is a one). The signature of the fault-free response (i.e., the compactor outputs) in this cycle will be (Z_1,Z_2,Z_3,Z_4) = (1XX1). In the presence of the error in I_2 due to the fault under consideration the compactor outputs will be (0XX1). It can be seen that the error in I2 causes an error in Z1. For this example if the fault-free response in a scan cycle has two X values as in (X0000X), then the fault-free signature for the cycle would be (XXXX) and hence any error in this scan cycle will be masked or blocked. In general, it can be shown that if a space compactor is based on a parity check matrix with columns of identical weight, then any single error value in a scan cycle produces an error at the compactor outputs even in the presence of an X value in another position in the same cycle. If Xs are not present, then any one or two errors in the same cycle produce an erroneous output. In X-compact [16] the parity check matrix columns are chosen to have odd weight (odd number of ones), which ensures error propagation to compactor outputs if one, two or any odd number of errors are present in a scan block with no Xs. In [16,17] construction of parity check matrices to detect errors in the presence of more than one X value are also given. Compaction ratios achieved by compactors using these matrices will be smaller by factors proportional to the number of X values tolerated.

$$H = \begin{array}{c|cccccc|c} & I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & \\ \hline & 1 & 1 & 1 & 0 & 0 & 0 & Z_1 \\ & 1 & 0 & 0 & 1 & 1 & 0 & Z_2 \\ & 0 & 1 & 0 & 1 & 0 & 1 & Z_3 \\ & 0 & 0 & 1 & 0 & 1 & 1 & Z_4 \end{array}$$

Figure 2 (a). A parity check matrix

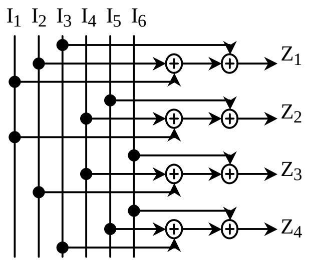

Figure 2 (b). A combinational compactor

The number of binary vectors of length Z with W ones in each vector is given by the binomial coefficient $\binom{Z}{W}$. The minimum number of outputs Z needed to compact test response data from N scan chains using the X-Compact procedure is obtained by determining Z such that $\binom{Z}{W} \geq N$, where W is an odd integer larger that 1. Since $\binom{Z}{W}$ has a maximum value when W is an odd integer equal to (Z-1)/2 or (Z+1)/2 if Z is an odd integer, and when W is equal to Z/2-1 or Z/2 if Z is an even integer, the minimum number of compactor outputs needed for X-Compact is obtained by setting W to these values.

For example, using X-Compact, if one wants a compactor to support 200 internal scan chains and guarantee detection of a single error in the presence of an unknown value in the same scan cycle as the error, a minimum of ten compactor outputs are necessary. Thus in this case the maximum achievable compaction ratio is 20. This limitation may be unacceptable in some cases. For example, the available tester may not support 10 scan outputs or it may be necessary to achieve a higher compaction ratio. Clearly, it is desirable to have a compactor design methodology that works for any given numbers of scan chains and outputs without degrading fault detection.

Another limitation of X-Compact is related to the hardware requirement. Assuming no fan out of more than one from an XOR gate the number of XOR gates required to compute the signature (parity checks) using a parity check matrix H of N columns and Z rows with column weight W is $(N*W - Z)$. This can be seen by noting that matrix H has a total of $N*W$ ones. Each parity check corresponding to a row of H with R ones needs $(R - 1)$ XOR gates. Thus the complexity of the compactor for a given design with N scan chains is primarily determined by the column weight W. In X-Compact based compactor designs W can be reduced only if the number of outputs Z is allowed to be larger than the minimum required for the given *N*. If a higher value of *Z* is used the compaction ratio decreases.

The Block Compactor described in the next section provides a method to design compactors for any given numbers of scan chains and compactor outputs. It also provides the flexibility to use any column weight. Additionally experimental results on industrial designs show that Block Compactors achieve higher compression ratios.

3. Block Compactor

The main idea behind the Block Compactor is to encode the response data of several scan cycles into a sequential signature and observe each signature in more than one scan shift cycle. By using the term sequential signature we emphasize the fact that it may take several clock cycles to observe the signature. This is in contrast to the earlier proposed space compactors, in which a signature for each scan cycle is observed in the same scan cycle. Initially, for the sake of simplicity, we describe a basic version of the Block Compactor in this section. A hardware efficient version of the compactor is described in Section 4. In this section we also propose guidelines to choose parameters of Block Compactors.

3.1 The Basic Block Compactor

Figure 3 illustrates an example of the Block Compactor. In this example, we are given six scan chains and two outputs. For this design, the compactor suggested by X-Compact cannot be used since at least five outputs are required by X-Compact. This can be seen by the fact that to compact data from six scan chains we need a parity check matrix with six columns with each column having the same odd weight larger than one. The minimum number of rows of the matrix to satisfy this constraint is five.

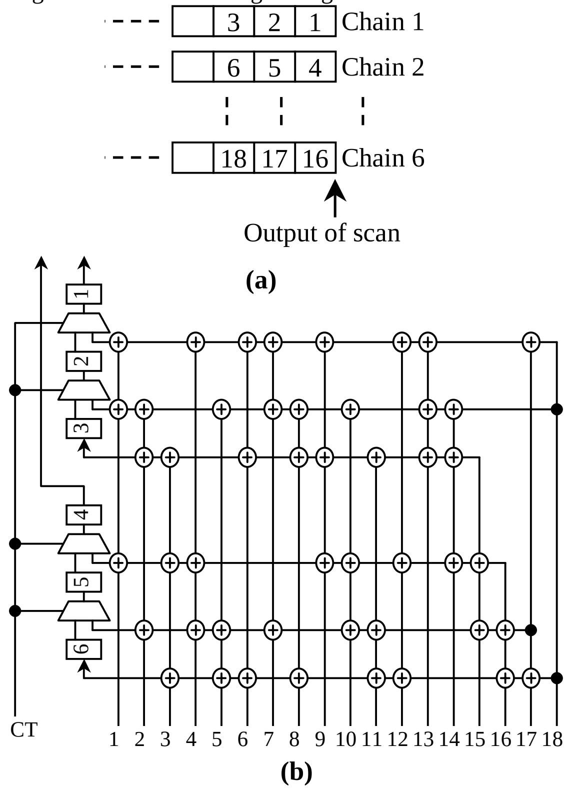

Figure 3. Example block compactor

In Figure 3(a), we show the output side of the scan chains and we number the last three scan cells of every chain as shown in the figure. In Figure 3(b), we illustrate the proposed Block Compactor. In Figure 3(b) the numbered rectangles are memory elements and the parallelograms are multiplexers. The compactor outputs are the output of memory element 1 and the output of memory element 4. Behind each of the two outputs, there are three memory elements which are used to store the sequential signature. The output of each numbered scan cell (cf. Figure 3(a)) is connected to the inputs of three memory elements through XOR gates that compute the signature, or parity checks, on the data in three scan cycles. The parity checks computed correspond to the parity check matrix shown in Figure 4. Note that this matrix has 18 columns corresponding to the eighteen numbered scan elements in Figure 3(a) and six rows corresponding to the number of memory elements in the compactor which store the signature computed over three scan clocks prior to shifting the signature out. The clock line of these memory elements is the same as the shift clock of the scan chains. Depending on the value of the control signal CT that drives the select inputs of the multiplexers, each memory element can capture either the output of the previous memory element or the signature computed by the network of XOR gates on the contents of the last three cells of the scan chains.

The Block Compactor works in the following manner. Before we shift out the test response, the control signal CT is set to 1 to capture the outputs of the XOR circuit computing the signature (or equivalently the parity checks). After the first clock, the sequential signature of the numbered scan cells of Figure 3(a) is captured into the memory of the compactor and CT is reset to 0. At this point, we can observe the first two bits of the sequential signature from the circuit output. At the second and third clock pulses, the other four bits of the signature are shifted out and the numbered scan cell positions are updated with 3 cycles of response data from scan cells to their left. After the third clock pulse, the control signal CT is set again and the above procedure is repeated. Figure 5 shows the waveform of the control signal and its relation to the scan shift clock. We call the response data that are compacted together into one sequential signature a *data block*. For instance, in the above example, the data block consists of three consecutive scan cycles.

In general, a Block Compactor with Z outputs that computes the signature of a data block of M scan cycles from a design with

$$
\begin{array}{cc}
 & \begin{array}{cccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \end{array} \\
H = & \left[\begin{array}{cccccccccccccccccc}
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}\right]
\begin{array}{c} 1 \\ 2 \\ 3. \\ 4 \\ 5 \\ 6 \end{array}
\end{array}
$$

Figure 4 The parity check matrix for Fig 3(b) compactor

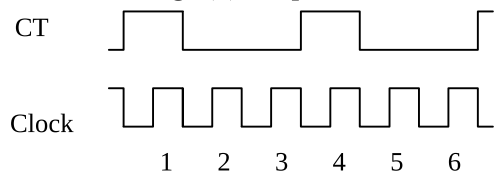

Figure 5. Waveform of control signal

N scan chains uses a parity check matrix with M*N columns and Z*M rows. In addition to the XOR gates for computing the signature, the compactor also uses Z*M memory elements to store the sequential signature prior to shifting out. It can be seen that the compactors proposed earlier in [12, 16, 17] are a special case of Block Compactors, where the response data in each shift cycle is a data block and each signature is observed in one cycle.

Next we consider a number of issues related to the design of "good" Block Compactors. From experiments conducted on several industrial designs we develop guidelines to design good Block Compactors.

3.2 General considerations

Given the number of scan chains and the number of desired outputs, we need to answer two questions in order to design a Block Compactor. One is how to choose the columns of the parity check matrix and the other is how many scan cycles are to be included in a data block. The latter determines the number of columns of the parity check matrix as well as the number of memory elements used in the compactor.

As mentioned earlier in this paper, we use equal weight columns as in X-Compact. That is, every column of the parity check matrix should have the same odd number of ones and no two columns are the same. When this rule is followed, it is guaranteed that one, two and any odd number of errors in a data block will be mapped to an incorrect signature and hence detected. In addition, a single error in a data block is guaranteed to be detected in the presence of an unknown value in the same data block.

With the parity check matrix column described above, we next determine the minimum amount of memory in the compactor or equivalently the minimum number of scan cycles in a data block. Let N be the number of scan chains, Z the number of outputs of the compactor and W the weight of the columns of the parity check matrix. Then the minimum number of scan cycles in a data block should satisfy the inequality given below.

$$\binom{Z * M}{W} \geq N * M \qquad (1)$$

Table 1 lists some compactor configurations and the maximum number of scan chains that can be accommodated for the given number of compactor outputs. The first column of Table 1 shows the number of outputs considered. The second column shows the maximum number of scan chains accommodated if X-compact is used. The last two columns give the maximum number of scan chains accommodated by two Block Compactor configurations with parity check matrix column weights 3 and 5, respectively. For each weight, we give two entries of maximum number of scan chains supported corresponding to the value of M, the number of scan cycles in a data block, equal to 4 and 6, respectively. It can be seen that for a given number of scan outputs, a Block Compactor can support a much larger number of scan chains than X-compact by using a sequential signature that requires a small number of memory elements. In other words, Block Compactors have the potential to achieve much higher compaction ratios than X-Compact. Actually, even with a single output, Block Compactors can support any number of scan chains if the data block size is large enough. Another observation is that larger column weights enable the compactor to support more scan chains than smaller column weights with the same number of memory elements. It can also be observed that for the same column weight W, the number of scan chains supported by a Block Compactor can be increased by increasing M, the data block size. Thus, in designing a Block Compactor for a design with a given number of scan chains and a desired compaction ratio, one can trade off the parity check column matrix weight W and the data block size M. Such flexibility does not exist in the designs based on X-Compact and I-Compact.

Table 1. Maximum number of observable scan chains in X-Compact and block compactor

Outputs	X-Compact	W=3	W=5
4	4	140, 337	1092, 7084
6	20	506, 1190	10626, 62832
8	56	1240, 2882	50344, 285384

It is worth mentioning that if we use larger data blocks than the minimum necessary for given numbers of scan chains and compactor outputs, parity check columns can be randomly chosen from the available pool of columns, which is larger than needed. This random selection helps to balance the XOR connections and reduces the probability of aliasing.

3.3 Unknown Value Tolerance Considerations

It is important for a compactor to enable detection of errors in the presence of unknown values in the test responses. Using an appropriate procedure for selecting columns of the parity check matrix, an error in a data block is guaranteed to be detected in the presence of one unknown value in the same data block. However, if a data block contains two or more unknown values, some of the known values in the data block may become unobservable or masked. As an example, consider the compactor illustrated in Figure 3 based on the parity check matrix of Figure 4. If the data in scan cells 1 and 2 are unknown, the data in cell 4 is blocked or masked. If all the errors in the test response of a faulty circuit are blocked, a defective chip escapes test and hence test quality is reduced due to the use of the compactor. To maintain high test quality using a Block Compactor, we next investigate the impact of data block size and parity check column weight on the unknown value tolerance of Block Compactors.

3.3.1 Data Block Size

From Inequality (1) of Section 3, we know that if the number of scan chains, the number of outputs and the parity check column weight are given, there is a minimum data block size that should be used for the Block Compactor. Smaller block sizes lead to smaller compactor hardware cost. However, the compactor with minimum block size may not achieve satisfactory test quality.

To compare the X tolerance ability of different compactors for a given design, we conduct the following experiment. First, we prepare a compact test set for the design without considering the compactor. Next, we simulate the test set on the design assuming a compactor is employed. After the simulation, we record the percentage of known response data blocked or masked due to

Table 2. X tolerance ability vs. block size for D1

Block Size	% block	Normalized Blockage
6 (1M)	1.65	1.00
12 (2M)	1.08	0.65
18 (3M)	0.58	0.35
24 (4M)	0.40	0.24

Table 3. X tolerance ability vs. block size for D3

Block Size	% block	Normalized Blockage
1 (1M)	11.62	1
2 (2M)	8.05	0.69
3 (3M)	5.80	0.50
4 (4M)	5.67	0.49

unknown values in the responses of the fault-free circuit to the tests. We repeat the above simulation procedure assuming different data block sizes.

The first experiment is conducted on an industry design that does not have many unknown values in its test response. We name the design D1. It has over 0.5M gates, 45K flip-flops and 308 scan chains. The percentage of Xs in the response for this circuit is 0.03%. Let the number of allowed compactor outputs be four and the weight of the parity check columns be three. Since the number of scan chains in the design is 308, from Inequality (1) we can calculate the minimum data block size for this design to be six scan cycles. In Table 2, we show the X tolerance ability of Block Compactors with different data block sizes for design D1.

The first column of Table 2 gives the data block size of the Block Compactor. In brackets after the block size, we show the block size relative to the minimum block size. For example 3M means 3 times as large as the minimum block size of six, which is 18. The second column shows the percentage of known response data bits that are blocked due to the corresponding compactor. In the last column, the percentage of blocked data in the second column is normalized such that the percentage of blocked data corresponding to the minimum block size is taken as one unit. It can be seen that with increasing block size, masking due to the use of the compactor decreases. Another observation is that the percentage of blocked data drops quickly as the block size changes from 1M to 3M and this improvement slows down when the block size is increased above 3M.

We verify the observations made above by performing the same experiment on another design. The design used in this experiment is D3 with 474 scan chains, 2.5M gates and 57K flip-flops. The percentage of Xs in the test response of D3 is 0.39%. In this case, we used 12 outputs and parity check column weight of five. Given that the number of scan chains in D3 is 474, we find that the minimum block size is one. Table 3 shows the experimental results for D3. The data in Table 3 is arranged similar to Table 2. Observations similar to the earlier ones can be made from Table 3. Actually, the compactor in the first row of Table 3 corresponds to the design using X-Compact. This example shows that even in the case where an X-Compact design can be used, a Block Compactor with larger data block size reduces masking or blocking of known data.

The relationship between the normalized percentage of blocked data and the relative block size is shown graphically in Figure 6. Since we obtain similar results from the experiments performed on different designs with different parity check column weights and different compaction ratios, we conclude that our observations are likely valid for other designs. That is, a larger block size is better than a smaller block size in tolerating unknown values in test responses. In addition, the improvement obtained by increasing the block size is large if the block size is no more than 3M where M is the minimum block size. If the block size is larger than 3M, the incremental benefit is minimal. So, our suggestion is to use block size of 3M in the Block Compactor design. In the rest of this work, we will follow this design rule for the compactors used in other investigations.

3.3.2 Weight of Parity Check Matrix Columns

To evaluate the impact of the weight of the parity check columns on the tolerance of X values for the compactors, we conducted experiments similar to the ones in the last subsection. In these experiments, we use design D1 and we consider four compaction ratio situations. For each compaction ratio, we built three Block Compactors, with column weights three, five and seven. For all the compactors considered, the block size is always 3M where M is the minimum block size for the configuration. Table 4 summarizes the results of this experiment.

Table 4. X tolerance ability vs. column weight for D1

# output	Compaction ratio	% block		
		Weight 3	Weight 5	Weight 7
6	51.3	0.38	0.60	1.12
7	44.0	0.34	0.29	0.66
8	38.5	0.21	0.20	0.28
9	34.2	0.14	0.10	0.13
10	30.8	0.12	0.05	0.05

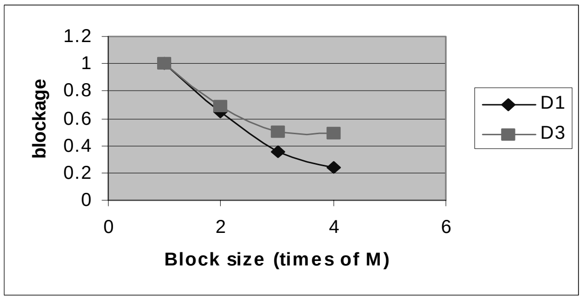

Figure 6. Block size vs. % blocked data

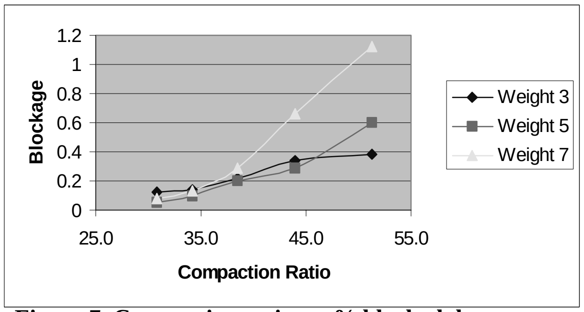

Figure 7. Compaction ratio vs. % blocked data

The first two columns of Table 4 give the number of compactor outputs and the corresponding compaction ratio. The last three columns give the percentage of blocked known values for the three Block Compactors. The data of Table 4 is given in graphical form in Figure 7. One observation that can be made from the results of this experiment is that if we keep the column weight constant, the percentage of unobserved known values increases as the compaction ratio increases. It can also be seen from the results that if the compaction ratio is large, a smaller column weight is better than a larger column weight in tolerating X values. However, if the compaction ratio is small, using a larger column weight gives better performance.

In reality, the impact of column weight is not only related to the compaction ratio but also to the density of unknown values in the test response. For example, when the compaction ratio is about 30 in Table 4, we find that the compactor with column weights five or seven is better than the compactor with weight three. However, when we performed the above experiment on D3, for which the percentage of unknown values is much higher than for D1, we found that the compactor with column weight three is better than the compactor with weight seven when the compaction ratio is near 30.

In conclusion, we observe that if the product of the compaction ratio and the percentage of unknown values is low, a Block Compactor with larger parity check column weight leads to higher test quality. On the other hand, if this product is high, a Block Compactor with a smaller parity column weight is better. We also noticed that in the situation where a larger column weight works better, the Block Compactor with a smaller weight also gives good enough results. For example, in the last row of Table 4, even though weights five and seven are better than weight three, the difference is not significant and the results using weight three may be adequate. Since the percentage of unknown values for a design is not known when the compactor is designed at the RTL level, we suggest using smaller column weight such as three. This choice leads to compactors with smaller logic and good performance.

3.4 Aliasing

In the presence of X values, aliasing due to data compaction may occur if the signatures of the fault-free circuit and a faulty circuit are the same as well as if all the errors in the response are masked by the X values. The experiments in the previous section showed that the probability that a known value in the test response is blocked or masked by the X values can be decreased by increasing the size of the data block or equivalently the size of the memory in the compactor. In this section, we show that the probability of aliasing in the absence of X values also decreases with the memory size of the compactor.

Given that the Block Compactor calculates the parity checks of a block code, it is known [12] that error patterns that alias must be code vectors or equivalently vectors that are in the null space of the parity check matrix. Let H be the parity check matrix. It has N*M columns and M*Z rows, where N is the number of scan chains, M is the number of scan cycles in a data block and Z is the number of outputs of the compactor. If all error patterns are equally probable then the aliasing probability is given by Equation (2) given below.

$$\text{P(aliasing)} = \frac{\text{(The number of vectors in the null space of H)} - 1}{\text{The total number of vectors of dimension N*M}}$$

$$= \frac{2^{N*M-M*Z}-1}{2^{N*M}} \approx 2^{-M*Z} \qquad (2)$$

Equation (2) shows that the probability of aliasing in the absence of X values decreases with M and Z. For a given number of compactor outputs Z, the probability of aliasing decreases with M.

4. Advanced Block Compactor Design Techniques

In the last section, we introduced the basic design of a Block Compactor. There are two hardware related disadvantages in this basic design. First, the inputs to the compactor come from the outputs of the scan cells in the last M stages of the scan chains. Thus, the core of the circuit under test is affected. It is preferable to drive the compactor inputs only from the outputs of the scan chains as done in most designs. Second, the number of XOR gates used to compute the parity checks are large due to the fact that parity checks on all bits in the data block with M scan cycles are computed simultaneously and stored in the compactor memory. While the contents of the compactor memory are shifted out, the XOR gates are not computing useful information.

A simple solution to remove the two disadvantages of the basic Block Compactor noted above is to choose a compactor design such that its inputs are driven only by the outputs of the scan chains. This can be achieved by properly selecting the parity check matrix for the compactor. This is described next.

4.1 Hardware Efficient Design of Block Compactors

Consider the parity check matrix H of Figure 4. The second column of H is obtained from the first column by rotating the first column down once (by one position). Similarly, the third column is obtained by rotating the second column down once, or rotating the first column down twice. It can be verified that the (3i+1)th, (3i+2)th and (3i+3)th columns have similar relations, i=0,1, …, 5. This relationship between columns can be exploited to design a much simpler compactor that is driven only by the outputs of the scan chains. The resulting design of the compactor is given in Figure 8. The operation of the compactor is discussed below.

It can be noted that only the last cells of the scan chains are connected to the XOR gates of the compactor and the output circuit consisting of the flip-flops and multiplexers is the same as in the basic compactor design. Additional memory elements and gates are included between the XOR gates computing the parity checks and the output memory elements. These additional memory elements develop the signature by rotating and adding the parity checks of the previous scan cycle to the parity checks of the current scan cycle. After the complete signature is computed for a data block, it is loaded into the output memory elements. While the sequential signature of the previous data block is shifted out, the signature of the next block is developed. Because the signature of the initial data block is computed over M scan cycles, the compacted response data is delayed by M cycles.

In general by selecting a parity check matrix such that a column corresponding to a scan cell is obtained by rotating down the column corresponding to the scan cell on its right in the same data block of the same scan chain, a compactor that is driven only by the outputs of the scan chains can be obtained. The investigations of the last section established the desirability of using larger than the minimum data block size needed for the Block Compactors. This also leads to a very large pool of parity check columns of chosen weight from which one can select column vectors to derive parity check matrices with the property described above. It should also be noted that the number of XOR gates to compute the parity checks is now much smaller since these are used only to compute the parity checks on the outputs of the scan chains and not on M scan cycles in a data block as in the case of the basic compactor design given in Section 3.

We conclude this section by an example comparing the logic for X-Compact and Block Compactors discussed in this section for design D1 with 308 scan chains. For this case X-Compact needs a minimum of eleven compactor outputs and parity check column weight of 5. The number of XOR gates needed for X-Compact is (5*308 – 11) = 1,529. Using parity check matrix column weight of 3 and 11 compactor outputs, the minimum number of scan cycles in a data block for the Block Compactor is 2. As we suggested in the last section we use three times larger data block of size 6. For this Block Compactor we need 859 XOR gates, 66 two input multiplexers, 66 two input AND gates and 132 flip-flops.

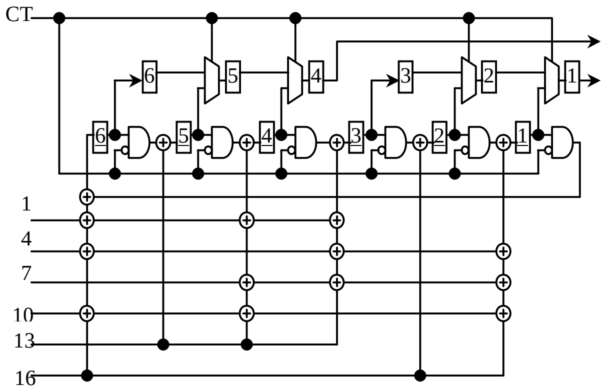

Figure 8. Block compactor accessing only scan chain outputs

4.2 Programmable connections to the compactor

The unknown values in test responses are usually not uniformly distributed. Some scan cells capture unknown values more frequently than others. If the connection between the scan chain outputs and the Block Compactor is fixed, some scan cells tend to be blocked more often than others due to the non-uniform distribution of unknown values. Thereafter, some faults may have much lower observability than other faults because these faults may only propagate to scan cells that are prone to being blocked by X values. If we have more than one way to connect the scan chains to the compactor, we can dynamically change the connections to reduce aliasing due to masking by X values.

To have multiple connections, we can use multiplexers. Figure 9 shows an example of how to implement four programmable connections between the core design and the compactor. In Figure 9, S1 and S2 are control signals provided by test stimuli. C1-C4 stand for four scan chains and P1-P4 for four inputs of the compactor. It can be seen from Figure 9 that with different S1, S2 combinations, C1-C4 connect to different compactor inputs.

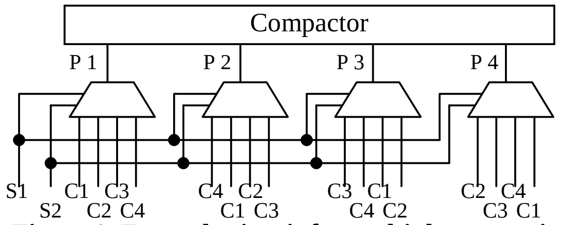

Figure 9. Example circuit for multiple connections

5. Experimental results

In this section, we present the fault aliasing performance of Block Compactors on industrial designs. Compactors are evaluated as follows. For a given design and a test set for the design, we know the number of faults N_f that can be detected by the test set. By adding a compactor on top of the design and fault simulating the test set, we find that the number of detected faults N_c is smaller than or equal to N_f. The fault coverage loss or the difference between N_f and N_c gives a measure of goodness of the compactor. The experimental results on four industrial designs are summarized in Table 5. In our experiment, all the test sets are generated by a well-known commercial ATPG tool.

In Table 5, after each design name, we list the number of faults detected by the tests, the number of scanned D flip-flops and the number of scan chains in the design. The column named X shows the percentage of unknown values in the test response.

For each design, we consider three compactor schemes. The first compactor investigated is the compactor by X-Compact [16] and the other two compactors are Block Compactors labeled C1 and C2, respectively (the descriptions of C1 and C2 are given below). For each considered compactor, we assume that there are 16 different connections between the design and the compactor. The method we used to determine which connection is to be used for a particular pattern is greedy, i.e., for each test pattern we select, out of the sixteen available, the connection that enables a maximum number of faults to be detected. Using an appropriate fault simulation procedure, one pass of fault simulation can be used to determine the optimal connection for a given pattern. It should be mentioned that the programmable connections to the compactor inputs were used in the case of X-Compact also to obtain comparable data. The number of outputs for the compactor of X-Compact is determined as given in [16]. For example, in order to observe 474 scan chains of design D3, at least 12 outputs are needed for X-Compact. Block Compactors C1 are built in such a way that their numbers of outputs are the same as that for the corresponding X-Compact compactors. The Block Compactors C2 are obtained by reducing the number of compactor outputs from that required for X-Compact and C1 until the number of faults left undetected is close but not higher than that for X-Compact. The columns denoted by W and M are the parity check column weights and the number of scan cycles in the data block used in the compactors, respectively. For all the constructed Block

Compactors, the column weight used is three and the data block size is 3M, where M is the minimum block size. The column weights for X-Compact compactors is 5 for all circuits. The last column of the table shows the number of faults (N_f - N_c) left undetected due to compaction. Inside parentheses, for designs D3 and D4, we give the number of faults left undetected by X-Compact if the programmable connections to the compactor inputs are not used.

Comparing the number of faults left undetected by X-Compact compactors and C1 compactors, we can see that with the same compaction ratio, well designed Block Compactors lose fewer faults for two of the designs and both compactors do not lose any faults for the other two designs. By observing the performance of compactors C2, we notice that on the average, over the four designs, Block Compactors provide a compaction ratio of 75 compared to a compaction ratio of 36 provided by X-Compact.

Finally from the last column of Table 5 we observe that the programmable connections to the compactor, suggested in this work, helps reduce the number of faults left undetected by X-Compact by over an order of magnitude. The hardware for X-Compact and Block Compactors for the four designs studied is comparable.

6. Conclusions

Finite memory compactor called a Block Compactor was proposed for compacting test response data with unknown values. Based on experiments, guidelines to design Block Compactors that achieve high compaction ratios and low aliasing were presented. Experimental data on several industrial designs showed that Block Compactors achieve lower aliasing and higher compaction than the recently proposed space compactors based on X-Compact [16].

7. References

[1] Pomeranz, L. N. Reddy and S. M. Reddy, “COMPACTEST: A Method to Generate Compact test Sets for Combinational Circuits,” IEEE Trans. CAD, pp. 1040-1049, 1993.

[2] Hamzaoglu and J. H. Patel, “Test Set Compaction Algorithms for Combinational Circuits,” Proc. ITC, pp. 283-289, 1998.

[3] M. Abramovici, M.A. Breuer and A.D. Friedman, “Digital Systems Testing and Testable Design”, IEEE Press.

[4] B. Koneman, “LFSR-Coded Test Patterns for Scan Designs,” Proc. European Test Conf, pp.237-242, 1993.

[5] R. Dorsch and H.-J. Wunderlich, “Tailoring ATPG for embedded testing,” Proc. ITC, pp. 530-537, 2001.

[6] J. Rajski et. al., “Embedded Deterministic Test for Low Cost Manufacturing Test,” Proc. ITC, pp. 301-310, 2002.

[7] I. Hamzaoglu and J. H. Patel, “Reducing Test Application Time for Full Scan Embedded Cores,” Proc. FTCS, pp. 260-267, 1999.

[8] I. Bayraktaroglu and A. Orailoglu, “Test Volume and Application Time Reduction Through Scan Chain Concealment,” Proc. DAC, pp.151-155, 2001.

[9] C. Krishna, A. Jas, and N.A. Touba, “Test vector encoding using partial LFSR reseeding,” Proc. ITC, pp. 885-893, 2001.

[10] S. M. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs", in Proc. VTS, pp. 103-108, 2002.

[11] E.H. Volkerink, A. Khoche and S. Mitra “Pcket-based Input Test Data Compression Techniques,” Proc. ITC, pp. 154-163, 2002.

[12] K. K. Saluja and M. Karpovsky, “Testing computer hardware through data compression in space and time,” Proc. ITC, pp. 83-88, 1983.

[13] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller and B. Koenemann, "OPMISR: The Foundation for Compressed ATPG Vectors", Proc. ITC, pp. 748-757, 2001

[14] I. Pomerantz, S. Kundu, S. M. Reddy, “On output response compression in the presence of unknown output values,” Proc. DAC, pp. 255-258, 2002.

[15] B. Pouya and N.A. Touba, “Synthesis of zero-aliasing elementary-tree space compactors” Proc. VTS, pp. 70-77, 1998.

[16] Mitra and K. S. Kim, “X-Compact: an efficient response compaction technique for test cost reduction,” Proc. ITC, pp. 311-320, 2002.

[17] J. H. Patel, S.S. Lumetta and S.M. Reddy, “Application of Saluja-Karpovsky Compactors to Test Responses with Many Unknowns,” Proc. VTS, pp. 107-112, 2003.

[18] J. Rajski, J. Tyszer, C. Wang and S. M. Reddy, “Convolutional Compaction of Test Responses,” Proc. of ITC, 2003.

Table 5. Experimental results

Design	#det. flts	DFFs	Scans	X	Comp.	# out	Ratio	W	M	Fault loss N_f- N_c
D1	0.85M	45K	308	0.03%	X-Compact	11	28	5	1	0
					C1	11	28	3	6	0
					C2	3	103	3	27	0
D2	2.23M	45K	380	0.01%	X-Compact	11	35	5	1	0
					C1	11	35	3	6	0
					C2	5	76	3	15	0
D3	1.65 M	57K	474	0.39%	X-Compact	12	40	5	1	1,635 (77,212)
					C1	12	40	3	6	464
					C2	8	59	3	24	1,416
D4	3.50 M	138K	457	0.09%	X-Compact	11	42	5	1	12 (638)
					C1	11	42	3	6	0
					C2	7	65	3	12	9