Synthesis of Single-Output Space Compactors with Application to Scan-based IP Cores*

Bhargab B. Bhattacharya
ACM Unit
Indian Statistical Institute
Calcutta - 700 035, India
bhargab@isical.ac.in

Alexej Dmitriev and *Michael Gössel*
Institute of Informatics
University of Potsdam
14415 Potsdam, Germany
alexej@mpag-inf.uni-potsdam.de
mgoessel@mpag-inf.uni-potsdam.de

Krishnendu Chakrabarty
Dept. of ECE
Duke University
Durham, NC 27708, USA
krish@ee.duke.edu

Abstract - This paper addresses the problem of space compaction of test responses of combinational and scan-based sequential circuits. It is shown that given a precomputed test set T, the test responses at the functional outputs of the given circuit-under-test (CUT) can be compacted to a *single periodic output*, with guaranteed *zero-aliasing*. The method is independent of the fault model and the structure of the CUT, and uses only the knowledge of the test set T and the corresponding fault-free responses—it is particularly suitable for intellectual property (IP) cores. A new concept of distinguishing outputs and characteristic response function is utilized for synthesizing the compactor. Relevant experimental results on hardware overhead for several ISCAS circuits are presented.

1 Introduction

Space compaction, which refers to the problem of reducing a wide data stream to a narrow signature stream, is commonly used for test response compression. A typical space compaction scheme for a general circuit-under-test (CUT) is illustrated in Fig. 1. A CUT with n primary inputs $X = \{x_1, x_2, \ldots, x_n\}$, m primary outputs $\{y_1, y_2, \ldots, y_m\}$ is given, and the m-bit test responses for a test set T are to be compacted to a q-bit data stream, where $q \ll m$. The *compaction ratio* (m/q) becomes maximum for a single-output space compactor, i.e., when $q = 1$. The main problem of space compaction is the loss of fault coverage due to *aliasing* which maps a faulty response to a fault-free signature of the CUT.

A complex system-on-a-chip (SOC) is built with several IP cores whose structural descriptions are not known to the system designer [1]. Specific design and test methodologies suitable for cores have been reported recently in [1]. Earlier methods [3]-[6] of space compaction need a fault model and an internal description of the CUT, and are therefore unsuitable for core-based systems. Space compactors [2, 7, 5] that do not rely on the structural information of the CUT, and are independent of fault model, have been proposed for reducing the volume of test response data in a core-based system.

*This work was funded in part by Deutsche Forschungsgemeinschaft, Germany, and in part by the US National Science Foundation under grant no. CCR-9875324.

It has been shown recently that zero-aliasing space compaction can always be achieved with $q = 1$ for a combinational CUT under a given test set, if the compactor, in addition to being fed by the outputs of the CUT, is also fed by the inputs to the CUT; see Fig. 2 [7]. Unfortunately, this approach is not applicable to a sequential circuit as its inputs include the internal state lines which may not be available outside the core for synthesizing the compactor. Recently, parity trees [8] and MISR [9] have been proposed for on-chip compression of scan outputs for large industrial circuits and SOC cores. In general, the number of functional outputs exceeds the number of the scan outputs in sequential circuits [1], and zero-aliasing space compaction for functional outputs remains an important open problem.

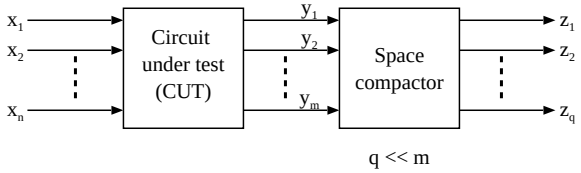

Figure 1: Conventional space compaction scheme.

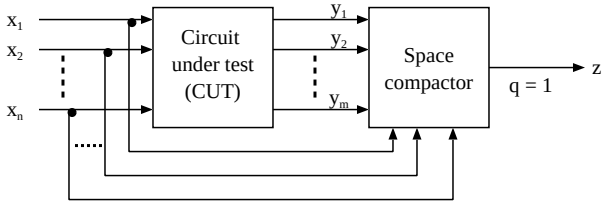

Figure 2: Modified space compaction scheme.

In this paper, we show that the limitation imposed by an earlier logarithmic lower bound on q [5] for zero-aliasing can be overcome by designing a compactor which is fed by the outputs of the CUT and a counter that designates the sequence of test application. For a combinational or a scan-based sequential circuit, we describe a procedure for synthesizing a *zero-aliasing* compactor that shrinks the functional outputs of the CUT to a *single periodic output* $(q = 1)$. Space compaction is not applied to the scan outputs. We assume

that the internal structure of the CUT is not known. The method is independent of the fault model, and uses only the information of the fault-free responses of the CUT to a precomputed test set T. All the errors produced by T at the outputs of the CUT, appear at the compactor output.

2 Zero-aliasing compaction with $q = 1$

Conventional space compactors [3, 4, 5] use a scheme as in Fig. 1. The lower bound on q for such a zero-aliasing space compactor is given by $q \geq \lceil log_2(\alpha+1) \rceil$, where α is the number of distinct fault-free responses of the CUT to the test set T, and therefore, in general, q cannot be made equal to 1, to preserve zero-aliasing [5]. However, if the input (test) information is also used in addition to the outputs (responses) to the CUT to synthesize the compactor, as in Fig. 2, then zero-aliasing can be achieved with $q = 1$ [7]. For a sequential circuit, the input information corresponding to the internal state lines may not be accessible. To circumvent this, we propose a new scheme as shown in Fig. 3.

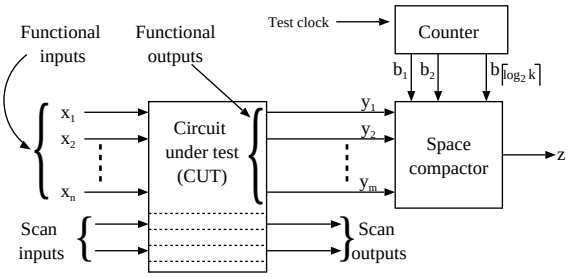

Figure 3: Space compaction using a counter.

Let the vectors in the test set $T = \{t_1, t_2, \ldots, t_k\}$, be applied to the CUT in the sequence: t_1 followed by t_2, t_2 followed by t_3, and so on. Each vector t_i consists of bits corresponding to the primary inputs, and the scan inputs. We use a binary counter with $\lceil log_2 k \rceil$ bits, whose current state designates the particular test being applied to the CUT at that instant. The counter starts from the all-0 state, and is incremented by the *test clock*, i.e., whenever the application of a test vector to the CUT is completed. The space compactor circuit is fed with the functional outputs $y_1, y_2, \ldots, y_m$ of the CUT, and the counter outputs $b_1, b_2, \ldots, b_{\lceil log_2 k \rceil}$.

Lemma 1 *Given a CUT C and a test set T, it is always possible to design a zero-aliasing space compactor with $q = 1$, using the scheme of Fig. 3, for all errors produced by T at the functional outputs of C.*

Proof: We design a compactor whose output z is given by the following Boolean function:

$$z = \bigvee_{t \in T} \left\{ \bigwedge_{i=1}^{\lceil log_2 k \rceil} b_i^*(t) \cdot \bigwedge_{j=1}^{m} y_j^*(t) \right\} \qquad (1)$$

where, $\bigvee(\bigwedge)$ denotes logical OR(AND) operation, and

$$b_i^*(t) = \begin{cases} \overline{b_i}, & \text{if the } i\text{-th bit of the counter state is 0 when the test t is applied,} \\ b_i, & \text{otherwise;} \end{cases}$$

$$y_j^*(t) = \begin{cases} \overline{y_j}, & \text{if the } j\text{-th output of the fault-free CUT is 0 under the test vector } t, \\ y_j, & \text{otherwise.} \end{cases}$$

Thus, $z = 1$, if and only if a test vector $t \in T$ when applied to the functional outputs of C. The identity of t is uniquely captured by the counter state. Hence, it achieves zero-aliasing single-output space compaction for all errors that are produced by T in C.

3 Two-stage compaction

We assume that for the CUT C, a test set $T = \{t_1, t_2, \ldots, t_k\}$, and the corresponding fault-free output response vectors $Y(t_i), 1 \leq i \leq k$, are available. The collection of all the response vectors is described by the *response matrix* RM $= Y[t_i, y_j], 1 \leq j \leq m$.

The proposed scheme of the compactor to achieve zero-aliasing with a single-output is shown in Fig. 3. If the CUT is fault-free, then the compactor will generate a *single* periodic (alternating) output z of 0's and 1's (e.g., 010101...), when the test vectors in T are applied to C in a pre-defined sequence.

The compactor operates in two-stages. In the *first stage*, m functional outputs $y_1, y_2, \ldots, y_m$, are compacted to p outputs $z_1, z_2, \ldots, z_{p-1}, z_p$, where $\lceil log_2(\alpha+1) \rceil \leq p \leq m$. We design this stage based on a new concept of distinguishing lines (D), and characteristic function (Ch). The outputs $z_1, z_2, \ldots, z_{p-1}$ correspond to the set D, and z_p implements the characteristic function (see Fig. 4). The outputs of this stage feeds the *second stage* compactor, which consists of the following four logic blocks:

(i) a binary counter with $\lceil log_2 k \rceil$ bits, whose state identifies the test vector currently in application;

(ii) a combinational logic block C_{map} (called mapping logic) which is driven by the counter state $b_1, b_2, \ldots, b_{\lceil log_2 k \rceil}$, and generates matching functions $g_1, g_2, \ldots, g_{p-1}, g_p$, corresponding to the p outputs of the first-stage compactor;

(iii) a comparator block having at most p EX-OR gates, where the output of the i-th gate G_i is given by $h_i = z_i \oplus g_i$, $1 \leq i \leq p$;

(iv) a self-checking CMOS code checker which is fed by the lines $h_1, h_2, \ldots, h_p$, and produces a single periodic output stream at z.

3.1 Design of the first-stage compactor

To all distinct fault-free responses to the CUT, we assign a unique signature with p bits, that can distinguish each response vector $Y(t_i)$ from all other distinct vectors in RM, as well as from all the vectors not contained in RM.

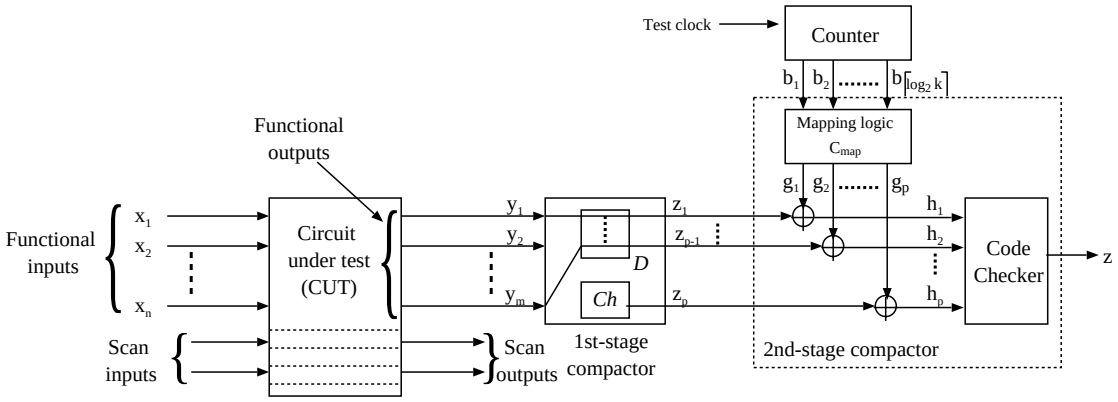

Figure 4: Scheme of space compactor with a single periodic output z.

Table 1: Distinguishing columns

Resp. vectors	CUT outputs						Disting. (columns)		
	y_1	y_2	y_3	y_4	y_5	y_6	y_1	y_2	y_6
$Y(t_1)$	0	0	0	1	1	0	0	0	0
$Y(t_2)$	0	1	0	1	1	1	0	1	1
$Y(t_3)$	1	0	0	0	1	1	1	0	1
$Y(t_4)$	0	0	0	1	1	0	0	0	0
$Y(t_5)$	0	1	1	0	0	0	0	1	0
$Y(t_6)$	1	1	1	0	1	1	1	1	1
$Y(t_7)$	1	0	0	0	0	0	1	0	0

Given the response matrix RM $= Y[t_i, y_j]_{k\times m}$, we choose a minimum set of columns (called *distinguishing columns*) to form a reduced sub-matrix $Y'_{k\times(p-1)}$, $(p-1) \leq m$, such that for any pair of test vectors t_i, t_l, $Y'(t_i) \neq Y'(t_l) \iff Y(t_i) \neq Y(t_l)$. In other words, each row vector (called *distinguishing vector*) in the reduced matrix $Y'_{k\times(p-1)}$ identifies the vector in the original RM uniquely, distinguishing it from all other distinct rows of RM. These $(p-1)$ columns of the response matrix which correspond to certain primary outputs of the CUT, are called *distinguishing outputs or lines (D)*; they directly form the outputs $z_1, z_2, \ldots, z_{p-1}$ of the first-stage compactor.

The p^{th} output of the first-stage compactor $z_p(y_1, y_2, \ldots, y_m)$, called the *characteristic function* is given by:

$$z_p = \begin{cases} 1, & \forall \text{ response vectors to tests } t_i, 1 \leq i \leq k; \\ d & (\textit{don't care}), \text{ if the distinguishing vector} \\ & \text{in } \{y_1, y_2, \ldots, y_m\} \text{ is different from all} \\ & \text{those in RM;} \\ 0, & \text{otherwise.} \end{cases}$$

In the limiting case, when $p-1 = m, z_p = 0$. The characteristic function z_p can be synthesized following the above description.

Example 1: Consider a CUT C with 6 outputs $y_1, y_2, \ldots, y_6$ whose fault-free response vectors $Y(t_i)$ for 7 tests $t_1, t_2, \ldots, t_7$, are shown in Table 1. In this example, the response vectors $Y(t_1)$ and $Y(t_4)$ are identical. The distinguishing columns are y_1, y_2, and y_6. These lines directly form the compactor outputs z_1, z_2, and z_3 respectively (see Fig. 5). The characteristic function z_4 is described in Table 2. For each of the distinct response vectors, z_4 is set to logic 1. Since, there are 6 distinct distinguishing vectors, the cubes corresponding to the remaining two $(001ddd, 110ddd)$, are set as don't cares (d). The logic value of z_4 for all the remaining inputs are set to 0. Thus, in the first-stage, the six outputs of the CUT are compacted to four outputs z_1, z_2, z_3, and z_4.

Table 2: Characteristic function z_4 for Example 1

Distg. columns			Remaining columns			
y_1	y_2	y_6	y_3	y_4	y_5	z_4
0	0	0	0	1	1	1
0	1	1	0	1	1	1
1	0	1	0	0	1	1
0	0	0	0	1	1	1
0	1	0	1	0	0	1
1	1	1	1	0	1	1
1	0	0	0	0	0	1
0	0	1	d	d	d	d
1	1	0	d	d	d	d
Everything else						0

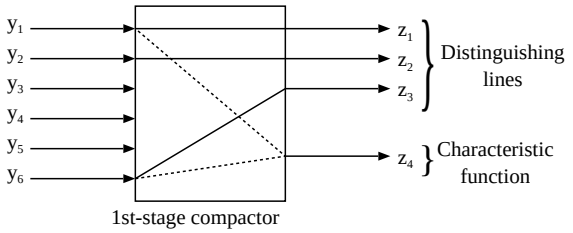

Figure 5: Distinguishing lines and z_p.

Lemma 2 *Given a CUT C with m functional outputs $y_1, y_2, \ldots, y_m$, and the fault-free responses to a test set T, the first-stage compactor with p outputs $z_1, z_2, \ldots, z_{p-1}, z_p$, $\lceil log_2(\alpha+1) \rceil \leq p \leq m$, designed as above, propagates all errors produced by T in C.*

Proof: For every correct response $Y(t_i)$ corresponding to a test t_i, the characteristic function at the compactor output will be 1, and the distinguishing outputs will show the signature of the concerned fault-free response vector. If an error occurs, the erroneous response vector $Y^f(t_i)$ must correspond to either another distinct vector in RM, or a vector $\notin$ RM. In the first case, the error will certainly propagate to the compactor output. In the second case, if the distinguishing vector of $Y^f(t_i)$ disagrees with those in RM, the error is carried through them, else the characteristic function assumes logic 0. Hence, the compactor is zero aliasing for all errors produced by T in C.

Synthesis of z_p using counter states

For efficient logic synthesis, the characteristic function z_p may be implemented in a slightly different fashion. We choose a set of columns randomly from the RM, and treat them similar to distinguishing columns for designing C_{map}. To define z_p, we keep the remaining columns of RM, and then to each row $Y'(t_i)$ of the reduced RM, we add the counter state $b_1, b_2, \ldots, b_{\lceil log_2 k \rceil}$ representing the test t_i. The truth table is then defined as before. This approach also preserves zero-aliasing. Thus, the size of the input set of variables defining z_p can be appropriately chosen to fit in available logic synthesis tools, providing us options for a flexible design by choosing the size of p. In the limiting case, no distinguishing lines may be used, i.e., the second-stage compactor is not required. All the m outputs of the CUT along with the counter bits will define z_p, and single-output compaction will be achieved. Thus, the scheme used in the proof of Lemma 1 is obtained. The other extreme case arises when all m outputs serve as distinguishing lines, and feed the second-stage compactor. In this case, the first-stage compactor is not needed. An example of the latter appears while analyzing the circuit s298 (see Table 6).

3.2 Design of the second-stage compactor

The second stage of the compactor takes $z_1, z_2, \ldots, z_{p-1}, z_p$ as well as the counter state $b_1, b_2, \ldots, b_{\lceil log_2 k \rceil}$, as inputs, and outputs an alternating string of 0's and 1's at z on application of the test vectors in a certain sequence.

Let the test set T be arbitrarily partitioned into two subsets T_0 and T_1 of almost equal size ($T = T_0 \cup T_1$, $T_0 \cap T_1 = \phi$, $|T_0| \geq |T_1| \geq 1$, $|T_0| - |T_1| \leq 1$). Let T_S denote the test sequence $\{t_{01}, t_{11}, t_{02}, t_{12}, \ldots\ldots\}$, where, $t_{0i} \in T_0$, and $t_{1l} \in T_1$. Thus T_S is formed by choosing consecutive vectors alternatingly from T_0 and T_1, until all $t \in T$ are exhausted.

3.2.1 Design of mapping logic C_{map}

The mapping logic C_{map} (see Fig. 4) generates matching outputs $g_i, 1 \leq i \leq p$, which are given by:
$g_i = z_i \oplus h_i$, $1 \leq i \leq p$, where

$$h_1 = h_2 = \ldots = h_p = \begin{cases} 0, & \forall\ test\ t_i \in T_0 \\ 1, & \forall\ t_i \in T_1 \end{cases}$$
$$\text{and } h_i = d(\text{don't care}), \quad otherwise. \tag{2}$$

The above condition implies that, if a test vector $t \in T_0$ ($t \in T_1$) is applied to the fault-free CUT, the logic values at the input lines $(h_1, h_2, \ldots, h_m)$ to the code checker will be simultaneously 0(1), and thus, on application of the test sequence T_S, these inputs receive alternatingly all-0 and all-1. Hence, it follows that, $\forall\ i, 1 \leq i \leq p$, the output function g_i of C_{map} is given by:

$$g_i(t) = \begin{cases} z_i(t) \oplus 0 = z_i(t), & \forall t \in T_0 \\ z_i(t) \oplus 1 = \overline{z_i}(t), & \forall t \in T_1 \\ = d \ \text{(don't care)}, & \text{otherwise.} \end{cases} \tag{3}$$

Eq. 3 can be readily used to synthesize the mapping logic C_{map} if a suitable partition of T into T_0 and T_1 is determined. For the $(2^{\lceil log_2 k \rceil} - k)$ additional counter states that do not correspond to the k tests, we set the logic values to $g_i, 1 \leq i \leq p$ as don't cares.

3.2.2 Determination of test sequence T_S

We use a simple heuristic to determine the test ordering with the goal of reducing the overhead of C_{map}.

Let $Z(Y(t))$ denote the vector $\{z_1, z_2, \ldots, z_p\}$ at the output of the first-stage compactor corresponding to the response vector $Y(t)$ to test t. We define, $N_1(Z) = \sum_{i=1}^{p} z_i$, which denotes the Hamming weight, i.e., the number of 1's in the vector $Z(Y(t))$.

We order ($\lrcorner$) the test vectors t's with respect to non-decreasing values of their $N_1(Z)$, i.e., $N_1(Z(Y(t_i))) < N_1(Z(Y(t_j)) \Rightarrow t_i < t_j$, and $t_i < t_j \Rightarrow N_1(Z(Y(t_i))) \leq N_1(Z(Y(t_j)))$.

$$\text{We choose,} \quad T_0 = \{t^1, t^2, \ldots, t^{\lceil \frac{k}{2} \rceil}\}, \quad \text{and}$$
$$T_1 = \{t^{\lceil \frac{k}{2} \rceil + 1}, \ldots, t^k\}, \quad \text{where}$$

$\lceil \frac{k}{2} \rceil$ is the smallest integer larger than or equal to $\frac{k}{2}$.

This makes $|T_0| = |T_1|$, if k is even, and $|T_0| = |T_1| + 1$, if k is odd, and the length of the test sequence T_S would be $(|T_0| + |T_1| = |T| = k)$. Hence, from Eqn. 2 & 3, it implies that the total number of 1's in the output description of C_{map} (# ON-sets) will be minimum.

Example 2: Consider the circuit of Example 1. Table 3 shows ordering of test vectors from which the sets T_0, and T_1 are determined. In Table 4, the test sequence $T_S = \{t_1, t_2, t_4, t_3, t_5, t_6, t_7\}$ is shown which is obtained by choosing consecutive vectors alternatingly from T_0 and T_1. The corresponding outputs of C_{map} to satisfy the desired checker inputs are shown in column 4. Assignment of counter states to the tests and the truth-table describing the outputs of C_{map} are

Table 3: Ordering of test vectors

Tests	1st-stage compactor outputs				$N_1(Z)$	Test set partition
	z_1	z_2	z_3	z_4		
t_1	0	0	0	1	1	
t_4	0	0	0	1	1	T_0
t_5	0	1	0	1	2	
t_7	1	0	0	1	2	
t_2	0	1	1	1	3	
t_3	1	0	1	1	3	T_1
t_6	1	1	1	1	4	

Table 4: Test sequencing and the required outputs of C_{map}

Test sequence	1st-stage compactor outputs				Required checker inputs				Required outputs of C_{map}			
T_S	z_1	z_2	z_3	z_4	h_1	h_2	h_3	h_4	g_1	g_2	g_3	g_4
t_1	0	0	0	1	0	0	0	0	0	0	0	1
t_2	0	1	1	1	1	1	1	1	1	0	0	0
t_4	0	0	0	1	0	0	0	0	0	0	0	1
t_3	1	0	1	1	1	1	1	1	0	1	0	0
t_5	0	1	0	1	0	0	0	1	0	1	0	1
t_6	1	1	1	1	1	1	1	1	0	0	0	0
t_7	1	0	0	1	0	0	0	0	1	0	0	1

Table 5: Counter states and the truth-table of C_{map}

Test sequence	Assigned counter state			Outputs of C_{map}			
T_S	b_3	b_2	b_1	g_1	g_2	g_3	g_4
t_1	0	0	0	0	0	0	1
t_2	0	0	1	1	0	0	0
t_4	0	1	0	0	0	0	1
t_3	0	1	1	0	1	0	0
t_5	1	0	0	0	1	0	1
t_6	1	0	1	0	0	0	0
t_7	1	1	0	1	0	0	1
—	1	1	1	d	d	d	d

shown in Table 5, which leads to the following Boolean functions for these outputs: $g_1 = b_2b_3 + b_1\bar{b}_2\bar{b}_3; g_2 = b_1b_2 + \bar{b}_1\bar{b}_2b_3; g_3 = 0; g_4 = \bar{b}_1$.

3.2.3 Comparator and the code checker

We use a special CMOS gate to implement the code checker as shown in Fig. 6. As observed in [10], this code checker can admit large fan-in. In test mode, all the p inputs to the checker, would be either 0 ($\forall t \in T_0$), or 1 ($\forall t \in T_1$). In the first case, all the p-MOS (n-MOS) transistors at the first level of the CMOS gate will be ON (OFF), and in the second case, they will be OFF (ON). Therefore, application of T_S to the CUT will generate a periodic output sequence 010101... at z, in the absence of any fault. Further, this checker is self-checking. The two code inputs (00...0, 11...1) are sufficient to detect all s-a-0, s-a-1 faults at the inputs/outputs of the checker, and all stuck-open faults in the transistors. All single transistor stuck-on faults except those in the inverter, are also detected by these two vectors.

3.2.4 Zero-aliasing space compaction

By Lemma 2, all the errors produced by T in the CUT are propagated to the outputs $Z = \{z_1, z_2, \ldots, z_{p-1}, z_p\}$ of the first-stage compactor. Design of C_{map} according to Eqns. (2) & (3), ensures that $\forall t \in T_0(T_1)$, the checker inputs $h_1 = h_2 = \ldots = h_p = 0(1)$, if and only if the vector Z is error free. When an error changes the vector Z under a test t, two cases may arise. If it causes the inputs to the checker to receive a vector consisting of 0's and 1's, then at the first level of CMOS gate, both the paths from V_{dd} to w, and w to GND (Fig. 6) will be open.

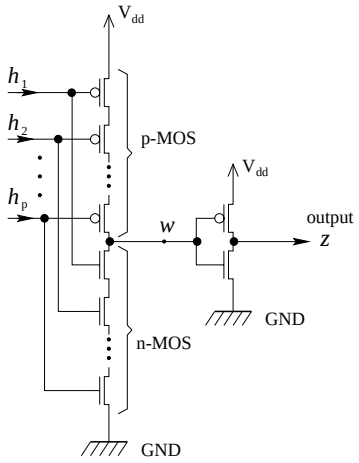

Figure 6: A CMOS code checker.

The logic value at w will be floating and will show the previous value. On the other hand, if the error inverts all the checker inputs simultaneously, (i.e., 11...1 instead of 00...0, or vice-versa), then also the output at z becomes opposite to its expected value. In both the cases, therefore, the alternating (periodicity) nature of the output at z is lost.

From the above discussion, the next theorem follows.

Theorem 1 *Given a test set T for a CUT C, the proposed design yields a single-output zero-aliasing space compactor for all errors produced by T in C.*

It may be noted that most of the errors propagate through the line z_p corresponding to the characteristic function. However, $\forall t \in T$, the logic value at z_p becomes 1, if no error is present. A stuck-at one fault at the line z_p will therefore never be detected by any test in T, and its presence could have a catastrophic masking effect on all the errors propagating through it. To overcome this, we add one additional vector t' in T, that produces a response vector not in RM, to enforce a 0 at z_p, prior to the synthesis procedure.

4 Experimental results

The proposed synthesis method was implemented on a SUN Ultra-5/Solaris workstation, and applied to several ISCAS-85 and ISCAS-89 benchmark circuits. In each case, we used compact test sets either generated by the MINTEST program [13] or by Hansen

Table 6: Hardware overhead of the compactor with $q = 1$, and zero-aliasing

Circuit	Number of inputs/outputs/ flip-flops	Circuit area	Test length $\|T\|$	Number of distingu. lines	Percentage area overhead		
					Charact. function	Mapping logic C_{map}	Total
c499	41/32/0	616	52	2	10.06	1.62	11.69
c6288	32/32/0	4800	12	4	3.54	0.56	4.10
s298	3/6/14	300	23	6	0	20.33	20.33
s344	9/11/15	329	13	5	16.41	7.29	23.70
s1423	17/5/14	1460	20	4	1.10	2.47	3.57
s1494	8/19/6	1417	100	16	4.30	38.39	42.70
s9234.1	36/39/211	8815	105	26	8.03	14.36	22.39
s35932	35/320/1728	35181	12	43	6.76	1.09	7.85
s38417	28/106/1636	38790	68	26	10.74	2.6	13.34

and Hayes [14]. A simple heuristic was used to determine a minimal set of distinguishing columns. ESPRESSO [11] followed by SIS [12], was used for synthesis of characteristic function and mapping logic. Since these tools are unable to handle very large benchmark circuits, we partitioned their outputs to ensure that the synthesis scripts ran to completion. For example, we partitioned the 320 functional outputs of s35932 into 10 groups of 32 outputs each.

The experimental results for zero-aliasing space compaction are shown in Table 6. We report the hardware overhead for two combinational and seven sequential circuits. These figures include the area for the mapping logic and the characteristic function. The area due to the code checker, counter, and comparators are not included since these are generic functional blocks that are not tailored to any CUT and can be reused across the complete design. The hardware overhead is clearly related to the test length—the synthesis approach is more efficient for short test sets. For example, the area overhead is only 4.10% for c6288 and 7.85% for s35932. Both these circuits possess very compact test sets. On the other hand, for s1494, the overhead is high since the test set contains 100 patterns and the circuit is relatively small.

5 Conclusion

We have shown analytically that the functional outputs of a scan-based sequential circuit can be compacted to a *single periodic output* with guaranteed *zero-aliasing*. A new synthesis procedure is then described for designing the space compactor. The proposed technique relies only on the knowledge of the fault-free responses of the circuit under test to a precomputed test set. It is therefore particularly suitable for compressing test responses at the functional outputs of IP cores. Experimental results for the ISCAS benchmark circuits demonstrate that the hardware overhead for the synthesized compactors is low to moderate in all cases.

References

[1] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded core based system chips," *Proc. International Test Conference*, 1998, pp. 130-143.

[2] M. Gössel, E. Sogomonyan, and A. Morosov, "A new totally error propagating compactor for arbitrary cores with digital interfaces," *Proc. VTS*, 1999, pp. 49-56.

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

[4] K. Chakrabarty, B. T. Murray, and J. P. Hayes, "Optimal zero-aliasing space compaction of test responses," *IEEE Trans. Comput.*, vol. 47, no. 11, pp. 1171-1187, Nov. 1998.

[5] M. Seuring and K. Chakrabarty, "Space compaction of test responses for IP cores using orthogonal transmission functions," *Proc. VTS*, 2000, pp. 213-219.

[6] J. Savir, "On shrinking wide compressors," *IEEE Trans. on CAD*, vol. 14, pp. 1379-1387, 1995.

[7] B. B. Bhattacharya, A. Dmitriev, and M. Gössel, "Zero aliasing space compaction using a single periodic output and its application to testing of embedded cores," *Proc. Int. Conference on VLSI Design*, January 2000, pp. 382-387.

[8] G. Hetheringten, et al., "Logic BIST for large industrial designs," *Proc. ITC*, 1999, pp. 358-367.

[9] H. Lang, J. Pfeiffer, and J. Maguire, "Using on-chip test pattern compression for full scan SoC designs," *Proc. ETW*, 2000, pp. 47-52.

[10] S. Kundu, E. S. Sogomonyan, M. Gössel, and S. Tarnick, "Self-checking comparator with one periodic output," *IEEE TC*, vol. 45, pp. 379-380, 1996.

[11] R. K. Brayton, et al., *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Boston, 1984.

[12] E. M. Sentovich, et al., "SIS: A system for sequential circuit synthesis," *Tech. Rep. UCB/ERL M92/41*, Electronic Research Lab., 1992.

[13] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithm for combinational circuits," *Proc. ICCAD*, Nov. 1998, pp. 283-289; (MINTEST-http://www.crhc.uiuc.edu/IGATE).

[14] M. C. Hansen and J. P. Hayes, "High-level test generation using physically-induced faults," *Proc. VTS*, 1995, pp. 20-28.