Aliasing-free compaction revisited

S.R. Das, A. Hossain, S. Biswas and E.M. Petriu

Abstract: The design of space-efficient support hardware for built-in self-testing is of immense significance in the synthesis of present day very large-scale integration circuits and systems, particularly in the context of design paradigm shift from system-on-board to system-on-chip (SOC). The authors revisit the general problem of designing zero-aliasing (or aliasing-free) space compression hardware in relation to embedded cores-based SOC for single stuck-line faults in particular, extending the well-known concepts of conventional switching theory and of incompatibility relation to generate maximal compatibility classes utilising graph theory concepts, based on optimal generalised sequence mergeability, as developed by them in earlier works. The authors briefly present the mathematical basis of selection criteria for merger of an optimal number of outputs of the module under test for realising maximum compaction ratio in the design, along with extensive simulation results on ISCAS 85 combinational and ISCAS 89 full-scan sequential benchmark circuits, with simulation programs ATALANTA, FSIM and COMPACTEST.

1 Introduction

As the electronics industry continues to grow, integration densities and system complexities continue to increase, the necessity for better and more efficient methods of testing to ensure reliable operations of chips, the mainstay of today’s many sophisticated devices and products, is being increasingly realised [1–4, 5]. The very concept of testing has a relatively broad applicability, and finding the most effective testing techniques that can guarantee correct system performance is of immense practical significance. Generally, the price of testing integrated circuits (ICs) is rather prohibitive, accounting for 35–55% of their total manufacturing cost. Besides, testing a chip is also time-consuming, taking up to about one-half of the total design cycle time. The amount of time available for manufacturing, testing and marketing a product, on the other hand, is on the decline. Moreover, as a result of diminishing trade barriers and global competition, customers now demand products of better quality at lower cost. In order to achieve this higher quality at lower cost, evidently the testing methods have to be improved. The conventional testing techniques of digital circuits require application of test patterns generated by a test pattern generator (TPG) to the module under test (MUT) and comparing the responses with known correct responses [5–9]. For large circuits, because of higher storage requirements for the fault-free responses, the customary test procedures thus become very expensive, and hence alternative approaches are sought to minimise the amount of needed storage [10].

Built-in self-testing (BIST) is a design process that provides the capability of solving many of the problems otherwise encountered in testing digital systems. It combines the concepts of both built-in test (BIT) and self-test (ST) into one termed BIST. In BIST, test generation, test application and response verification are all accomplished through built-in hardware, which allows different parts of a chip to be tested in parallel, thereby reducing the required testing time, besides eliminating the necessity for external test equipment. As the cost of testing is becoming the single major component of the manufacturing expense of a new product, BIST thus tends to reduce the manufacturing and maintenance costs through improved diagnosis [8, 9, 11, 12]. Several companies such as Motorola, AT&T, IBM and Intel have incorporated BIST in many of their products [13–18]. AT&T, for example, has incorporated BIST into more than 200 of their IC chips. The three large programmable logic arrays and microcode ROM in the Intel 80386 microprocessor were built-in-self-tested [19]. The general-purpose microprocessor chip, Alpha AXP21164 and Motorola microprocessor 68020, were also tested using BIST techniques [15]. More recently, Intel, for its Pentium Pro architecture microprocessor, with its unique requirements of meeting very high production goals, superior performance standards and impeccable test quality put strong emphasis on its design-for-test (DFT) direction [18]. A set of constraints, however, limits Intel’s ability to tenaciously explore DFT and test generation techniques, viz. full-scan or partial-scan or scan-based BIST [2].

AMD’s K6 processor is a reduced instruction set computer (RISC) core named enhanced RISC86 microarchitecture [17]. K6 processor incorporates BIST into its DFT process. Each RAM array of K6 processor has its own BIST controller. BIST executes simultaneously on all of the arrays for a predefined number of clock cycles that ensures completion for the largest array. Hence, BIST execution time depends on the size of the largest array [2]. AMD uses commercial automatic test pattern
BIST is widely used to test embedded regular structures that exhibit a high degree of periodicity such as memory arrays (SRAMs, ROMs, FIFOs and registers). This type of circuits does not require complex extra hardware for test generation and response compaction. Also, including BIST in these circuits can guarantee high fault coverage (FC) with zero-aliasing [5–7, 11, 12]. Unlike regular circuits, random-logic circuits cannot be adequately tested only with BIST techniques, since generating adequate on-chip test sets using simple hardware is a difficult task to be accomplished. Moreover, since test responses generated by random-logic circuits seldom exhibit regularity, it is extremely difficult to ensure zero-aliasing compaction. Therefore random-logic circuits are usually tested using a combination of BIST, scan design techniques and external test equipment.

A typical BIST environment, as shown in Fig. 1, uses a TPG that sends its outputs to an MUT, and output streams from the MUT are fed into a test data analyzer. A fault is detected if the test sequence is different from the response of the fault-free circuit. The test data analyzer is composed of a response compaction unit, storage for the fault-free responses of the MUT and a comparator. In order to reduce the amount of data represented by the fault-free and faulty MUT responses, data compression is used to create signatures (short binary sequences) from the MUT and its corresponding fault-free circuit. Signatures are compared and faults are detected if a match does not occur. BIST techniques may be used during normal functional operating conditions of the unit under test (online testing), as well as when a system is not carrying out its normal functions (off-line testing). In the case where detecting real-time errors is not that important, systems, boards and chips can be tested in off-line BIST mode. BIST techniques use pseudo-exhaustive or pseudorandom test patterns, or sometimes on-chip storing of reduced or compact test sets. Today, testing logic circuits exhaustively is seldom used, since only a few test vectors are needed to ensure full FC for single stuck-line faults [10, 19]. Reduced pattern test sets can be generated using existing algorithms such as FAN and others [1, 2]. BIT generators can often generate such reduced test sets at low cost, making BIST techniques suitable for on-chip self-testing.

The subject paper focuses primarily on the response compaction process of BIST techniques that basically formulate into realising appropriate means of reducing the test data volume coming from the MUT to a signature. Instead of comparing bit-by-bit the fault-free responses to the observed outputs of the MUT as in conventional testing methods, the observed signature is only compared with the correct one, thus reducing the storage needed for the correct circuit responses [20–28, 10]. The response compaction in BIST is carried out through a space compaction unit followed by time compaction. In general, P input sequences coming from an MUT are fed into a space compactor, providing L output streams of bits such that L ≪ P; most often, test responses are compressed into only one sequence (L = 1). Space compaction brings a solution to the problem of achieving high-quality BIST of complex chips without the necessity of monitoring a large number of internal test points, thereby reducing the testing time and area overhead by merging test sequences coming from these internal test points into a single stream of bits [29–34]. This single bit stream of length H is ultimately fed into a time compactor, and a shorter sequence of length B (B < H) is obtained at the output [35–44]. The extra logic representing the compaction network, however, must be as simple as possible, to be easily embedded within the MUT, and should not introduce signal delays to affect either the test execution time or the normal functionality of the module being tested. Moreover, the length of the signature must be as short as possible in order to minimise the amount of memory needed to store the fault-free response signatures. Also, signatures derived from faulty output responses and their corresponding fault-free signatures should not be the same, which unfortunately is not the case always. A fundamental problem with compaction techniques is error masking or aliasing [45, 10] which occurs when the signatures from faulty output responses map into the fault-free signatures,
usually calculated by identifying a good circuit, applying test patterns to it, and then having the compaction unit generate the fault-free references. Alasing causes loss of information, which affects the test quality of BIST and reduces the FC (the number of faults detected, after compaction, over the total number of faults injected). Several methods have been suggested in the literature for computing the aliasing probability. The exact computation of this aliasing probability is known to be an NP-hard problem [19, 46, 47]. In practice, high fault coverage, over 99%, is required, and hence any compression technique that maintains more percentage error coverage information is considered worthy of investigation.

A major challenge in realising efficient space compaction in BIST is the development of techniques that are simple, suitable for on-chip self-testing, require low area overhead and have little adverse impact on the MUT performance. With this perspective in focus, this paper revisits the general problem of designing zero-aliasing BIST support hardware with applications targeted towards embedded cores-based system-on-chip (SOC) [48–50], extending the well-known concepts of conventional switching theory, particularly those of cover table and frequency ordering commonly utilised in the simplification of switching functions and of incompatibility relation as employed in the minimisation of incomplete sequential machines, using graph theoretic concepts in the design [51–56], based on optimal generalised sequence mergeability as developed and applied by the authors in earlier works, for detectable single stuck-line faults of the MUT. The advantages of zero-aliasing compaction as discussed herein over earlier techniques are self-evident — zero-aliasing is achieved without any modifications of the original MUT, whereas the area overhead and signal propagation delay are relatively low compared with conventional linear compactors. In addition, the approach works equally well with both deterministic compacted and pseudorandom test sets. The method also involves less computation in the successive stages compared with the existing techniques based on exhaustive search for merger of only a pair of output lines of the MUT to achieve zero-aliasing [57]. This paper makes use of mathematically sound sequence criteria of merger of an optimal number of output lines of the MUT to decide on the logic for zero-aliasing, achieving maximal compaction ratio in the process, as is evident from extensive simulation experiments conducted on ISCAS 85 combinational and ISCAS 89 full-scan sequential benchmark circuits.

The paper is organised into six sections. The mathematical basis in the realisation of zero-aliasing compaction is presented in Section 2. Section 3 introduces the pertinent graph theoretic concepts utilised in the implementation of the design approach, and Section 4 provides the details of the algorithms development. In Section 5, simulation results are given, whereas concluding remarks are presented in Section 6.

2 Implementation of zero-aliasing space compression: mathematical basis

The mathematical basis [48–50, 46] underlying the realisation of proposed zero-aliasing space compaction is outlined in the following for the sake of clear understanding and completeness.

Property 1: Let A and B represent two of the outputs of an MUT. Let these MUT outputs be merged by a gate from the logic family AND/NAND, OR/NOR and XOR/XNOR, and let the gate output be \( z_1 \). Then, we might envisage the undermentioned possible scenarios:

Case 1: Fault-free (FF) outputs \( = \) faulty (F) outputs (outputs subject to the condition of MUT having faults), viz. FF = F \( \Rightarrow \) outputs A and B of the MUT do not detect any faults, and faults are hence not detectable at \( z_1 \).

Case 2: Only the faults that occur at A and B (subject to the condition of MUT having faults) are detectable at \( z_1 \). FF \( \neq \) F.

Case 3: Faults occur at A and B but either all or some are not detectable at \( z_1 \) \( \Rightarrow \) FF \( \neq \) F. In this case, the faults missed at \( z_1 \) are detected additionally at other outputs of the MUT (besides A and B).

Definition 1: Let \( A, B, C, \ldots \) be the different outputs of an \( n \)-input \( m \)-output MUT. Let the faults detected at the MUT outputs \( A, B, C, \ldots \) be \( \theta \), where \( \theta \leq \beta \), the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests \( \tau, \tau \leq 2^n \), \( \tau \) might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the two outputs A and B conforms to conditions of cases 1 and 2 above (but not case 3). If the MUT outputs A and B are now merged by an AND/NAND gate, we define output lines A and B to be strongly AND/NAND compatible, written as

\[(AB) \text{s-AND/NAND compatible.} \]

Definition 2: Let \( A, B, C, \ldots \) be the different outputs of an \( n \)-input \( m \)-output MUT. Let the faults detected at the MUT outputs \( A, B, C, \ldots \) be \( \theta \), where \( \theta \leq \beta \), the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests \( \tau, \tau \leq 2^n \), \( \tau \) might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the two outputs A and B conforms to conditions of case 3 (but not cases 1 and 2). If the MUT outputs A and B are now merged by an AND/NAND gate, we define output lines A and B to be weakly AND/NAND compatible, written as

\[(AB) \text{w-AND/NAND compatible.} \]

Definition 3: Let \( A, B, C, \ldots \) be the different outputs of an \( n \)-input \( m \)-output MUT. Let the faults detected at the MUT outputs \( A, B, C, \ldots \) be \( \theta \), where \( \theta \leq \beta \), the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests \( \tau, \tau \leq 2^n \), \( \tau \) might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the two outputs A and B conforms to none of the conditions as specified by cases 1–3. If the MUT outputs A and B are now merged by an AND/NAND gate, we define output lines A and B to be strongly AND/NAND incompatible, written as

\[(AB) \text{AND/NAND incompatible.} \]

Definition 4: Let \( A, B, C, \ldots \) be the different outputs of an \( n \)-input \( m \)-output MUT. Let the faults detected at the MUT outputs \( A, B, C, \ldots \) be \( \theta \), where \( \theta \leq \beta \), the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests \( \tau, \tau \leq 2^n \), \( \tau \) might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at
the two outputs $A$ and $B$ conforms to conditions of cases 1 and 2 (but not case 3). If the MUT outputs $A$ and $B$ are now merged by an OR(NOR) gate, we define output lines $A$ and $B$ to be strongly OR(NOR) compatible, written as

$$(AB) \text{ w-OR(NOR)} \text{ compatible}.$$ 

Definition 5: Let $A, B, C, \ldots$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A, B, C, \ldots$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\tau, \tau \leq 2^n, \tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the two outputs $A$ and $B$ conforms to conditions of case 3 (but not cases 1 and 2). If the MUT outputs $A$ and $B$ are now merged by an OR(NOR) gate, we define output lines $A$ and $B$ to be weakly OR(NOR) compatible, written as

$$(AB) \text{ s-OR(NOR)} \text{ compatible}.$$ 

Definition 6: Let $A, B, C, \ldots$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A, B, C, \ldots$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\tau, \tau \leq 2^n, \tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the two outputs $A$ and $B$ conforms to conditions of cases 1–3. If the MUT outputs $A$ and $B$ are merged under these conditions by AND(NAND), OR(NOR) or XOR(XNOR) gate, then, we define output lines $A$ and $B$ to be simply AND(NAND), OR(NOR) or XOR(XNOR) compatible, written as

$$(AB) \text{ AND(NAND), OR(NOR) or XOR(XNOR)} \text{ compatible}.$$ 

Theorem 1: Let $A, B, C, \ldots$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A, B, C, \ldots$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\tau, \tau \leq 2^n, \tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A$ and $B$ conforms to conditions of cases 1 and 2 above, so that the outputs $A$ and $B$ are AND(NAND) compatible. Similarly, let the outputs $B$ and $C$ be s-AND(NAND) compatible, and the outputs $A$ and $B$ be s-AND(NAND) compatible. Then the outputs $(ABC)$ are s-AND(NAND) compatible and all faults are detectable at $z_1$.

Theorem 2: Let $A_1, A_2, \ldots, A_m$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A_1, A_2, \ldots, A_m$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\tau, \tau \leq 2^n, \tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A_1, A_2, \ldots, A_m$ conforms to conditions of cases 1 and 2 above, so that the outputs $(A_1A_2\ldots A_m)$ are s-AND(NAND) compatible. Then, all faults are detectable at $z_1$.

Theorem 3: Let $A, B, C, \ldots$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A, B, C, \ldots$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\tau, \tau \leq 2^n, \tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A$ and $B$ conforms to conditions of cases 1 and 2 above, so that the outputs $A$ and $B$ are s-OR(NOR) compatible. Similarly, let the outputs $B$ and $C$ be s-OR(NOR) compatible, and the outputs $A$ and $C$ be s-OR (NOR) compatible. Then the outputs $(ABC)$ are s-OR(NOR) compatible and all faults are detectable at $z_1$. 

IET Circuits Devices Syst., Vol. 2, No. 1, February 2008 169
Theorem 4: Let $A_1, A_2, \ldots, A_m$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A_1, A_2, \ldots, A_m$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\pi, \tau \leq 2^n$, $\tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A_1, A_2, \ldots, A_m$ conforms to conditions of cases 1 and 2 above, so that the outputs $(A_1 A_2 \ldots A_m)$ are s-XOR(XNOR) compatible. Then, all faults are detectable at $z_1$.

Theorem 5: Let $A, B, C, \ldots$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A, B, C, \ldots$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\pi, \tau \leq 2^n$, $\tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A$ and $B$ conforms to conditions of cases 1 and 2 above, so that the outputs $(A B)$ are s-OR(NOR) compatible. Then, all faults are detectable at $z_1$.

Theorem 6: Let $A_1, A_2, \ldots, A_m$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A_1, A_2, \ldots, A_m$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\pi, \tau \leq 2^n$, $\tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A_1, A_2, \ldots, A_m$ conforms to conditions of cases 1 and 2 above, so that the outputs $(A_1 A_2 \ldots A_m)$ are s-XOR(XNOR) compatible. Then, all faults are detectable at $z_1$.

Corollary 1: Let $A_1, A_2, \ldots, A_m$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A_1, A_2, \ldots, A_m$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\pi, \tau \leq 2^n$, $\pi$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A_1, A_2, \ldots, A_m$ conforms to conditions of case 3 above, so that the outputs $(A_1 A_2 \ldots A_m)$ are w-AND(NAND) compatible. Similarly, let the outputs $B$ and $C$ be w-XOR(XNOR) compatible, and the outputs $A$ and $C$ be w-AND(NAND) compatible. Then the outputs $(ABC)$ are w-AND(NAND) compatible and all faults may or may not be detectable at $z_1$.

Theorem 7: Let $A, B, C, \ldots$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A, B, C, \ldots$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\pi, \tau \leq 2^n$, $\tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A$ and $B$ conforms to conditions of cases 1 and 2 above, so that the outputs $(A B)$ are w-OR(NOR) compatible. Then, all faults may or may not be detectable at $z_1$.

Corollary 2: Let $A_1, A_2, \ldots, A_m$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A_1, A_2, \ldots, A_m$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\pi, \tau \leq 2^n$, $\tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A_1, A_2, \ldots, A_m$ conforms to conditions of case 3 above, so that the outputs $(A_1 A_2 \ldots A_m)$ are w-OR(NOR) compatible. Then, all faults may or may not be detectable at $z_1$.

Corollary 3: Let $A_1, A_2, \ldots, A_m$ be the different outputs of an $n$-input $m$-output MUT. Let the faults detected at the MUT outputs $A_1, A_2, \ldots, A_m$ be $\theta$, where $\theta \leq \beta$, the total number of detectable faults at the MUT outputs when subjected to a compacted set of deterministic tests $\pi, \tau \leq 2^n$, $\tau$ might not be a minimal or non-minimal but complete set of tests, or to pseudorandom tests. Assume that the fault situation at the outputs $A_1, A_2, \ldots, A_m$ conforms to conditions of case 3 above, so that the outputs $(A_1 A_2 \ldots A_m)$ are w-OR(NOR) compatible. Then, all faults may or may not be detectable at $z_1$.

In actual situations, we do not know (and also it is rather difficult to know) whether the merged outputs conform to conditions specified by cases 1–3 as discussed, and as such we have to deal exclusively with the case of simply compatible. However, very recently, a novel approach to the solution of the problem utilising the concept of fault grading [58, 59] has been proposed, which renders the developed mathematical basis underlying the notion of strong and weak compatibilities really meaningful. But since this paper does not address the theory underlying that approach, it becomes necessary to check here every possible MUT output pair in a group for being simply compatible (AND/NAND, OR/NOR or XOR/XNOR) to form a larger maximal or non-maximal compatibility class (compatibility, in general, being a non-transitive relation). On the contrary, if a set of MUT outputs satisfies the condition of being strongly compatible (AND/NAND, OR/NOR or XOR/XNOR), they could all be merged by a single gate.
of the appropriate logic family, knowing that all the faults at the logic inputs will be detectable at its output, thus eliminating the need to use any heuristic in developing the compaction networks with maximal compaction ratio. In the current approach, there is no way of knowing if any MUT output pair is strongly compatible, and as such, for any compatibility class comprising 2 lines, when merged by a gate from the given logic family, the gate output needs to be tested for detection of all input faults by real experimentation.

3 Graph theoretic concepts and implementation of design approach

An important problem in relation to designing zero-aliasing space compression networks as proposed herein is to first find the sets of maximal compatibility classes (MCCs) of response data outputs of the MUT for logic families AND/NAND, OR/NOR and XOR/XNOR, given the information of the corresponding pairs of incompatibles. There exist many different methods in the literature to find the sets of maximal compatibility classes, based on information of either the incompatible pairs or compatible pairs. In this paper, use has been made of available graph theoretic approaches in the solution of the problem. Some relevant basic concepts of graph theory as used in the paper in this regard might be relevant here for the sake of completeness [51, 53–56].

3.1 Approach based on generation of maximal complete subgraphs or cliques of undirected graphs using the Bron–Kerbosch algorithm

Some important basic definitions are given as follows.

Definition 11: A symmetric or an undirected graph \( A = (V, E) \) is defined as an ordered pair consisting of a finite set \( V \) of nodes or vertices and a set of unordered pairs \((v, w)\) of distinct vertices called edges. Any two vertices \( v \) and \( w \) in \( A \) are said to be adjacent to each other if \((v, w) \in E\). A set \( S \) of vertices of \( A \) is a complete subgraph if \((v, w) \in E\) for all pairs of distinct vertices \( v, w \in S \). A maximal complete subgraph or clique of an undirected graph \( A \) is a complete subgraph that is not contained in any other complete subgraph of \( A \). The complement of an undirected graph \( A = (V, E) \) is the graph \( \bar{A} = (V, \bar{E}) \), where \( \bar{E} = \{(v, w) | v, w \in A, v \neq w, (v, w) \notin E\} \). A set \( S \) of vertices in \( A \) is said to be internally stable or independent if \((v, w) \notin E\) for all \( v, w \in S \).

That is, given a set \( S \subseteq V \) of vertices of \( A \), if none of the pairs of distinct vertices in \( S \) is adjacent to each other, then \( S \) is an independent set of \( A \), and any such independent set \( S \) of vertices of \( A \), which is not contained in any other independent set of \( A \), is called a maximal independent set of \( A \).

Note here that a set \( S \subseteq V \) of vertices of \( A \) is a maximal independent set of \( A \), if and only if, \( S \) is a clique of \( A \), and hence the problem of listing all cliques of a given graph \( A \) is simply equivalent to that of listing all maximal independent sets of its complementary graph \( \bar{A} \), and vice versa.

The clique problem of undirected graphs, in its many different aspects, has been intensively discussed in the literature, and many authors have developed a variety of fascinating approaches for its solution. It is important to observe here that this clique detection problem of graph theory is identical to the problem of deriving the collection of MCCs in a set of elements with compatibility relation. The cliques of an undirected graph are thus not mutually exclusive and can have non-void intersections. The maximal compatible problem as a counterpart of the clique problem has again been investigated by many authors in various disciplines. It is appropriate to remark here that the clique generation problem like some of the classical problems of combinatorics is an NP-complete problem [47, 52], and as such is quite intractable.

Bron et al. [51] developed two backtracking algorithms for generating all cliques, using a branch-and-bound technique that cuts off branches that cannot lead to a clique. These algorithms were subsequently reported by Bron and Kerbosch [53] and commonly known as Bron–Kerbosch algorithm in the literature. Their first version is a straightforward implementation of the basic algorithm and generates cliques in a lexicographic order. The second version is derived from the first and generates cliques in an unpredictable order in an attempt to minimise the number of branches to be traversed. The authors implemented their algorithms with others. Based on the results of implementation, the authors resolve that both versions of their algorithms perform quite well. For the Moon–Moser graphs, the authors’ second test case, the processing time for the first version was found proportional to \( 4^k \), whereas for the second version of the algorithms, it was proportional to \( 3.14^k \), for some constant \( k \) characteristic of the graphs.

The algorithms need at most \( \frac{1}{2}(M + 3) \) storage locations to contain arrays of small integers, where \( M \) is the size of the largest connected component in the input graph. In our proposed approach for zero-aliasing space compaction, use has been made specifically of this well-known Bron–Kerbosch algorithm for the generation of maximal compatible (cliques) of response data outputs for logic families AND/NAND, OR/NOR and XOR/XNOR, based on the information of their pairs of incompatibles.

3.2 Approach based on generation of maximal minimally strongly connected subgraphs: concepts

Definition 12: Consider an undirected graph \( A \) with \( n \) vertices, \( v_i, i = 1, 2, \ldots, n \). Two subgraphs \( A_1 \) and \( A_2 \) of \( A \) are said to be complementary to each other, if and only if, both \( A_1 \) and \( A_2 \) have the same set of vertices and one has edges connecting between those pairs of vertices that are not connected by edges in the other.

Definition 13: Consider a vertex \( v_i \) in an undirected graph \( A \). The degree of \( v_i \), \( d(v_i) \), is the number of edges of \( A \) incident in \( v_i \). The degree complement of a vertex \( v_i \), \( d'(v_i) \), is the degree of the vertex \( v_i \) in the complementary graph \( \bar{A} \). Two vertices \( v_i \) and \( v_j \) in \( A \) are said to be minimally strongly connected (MSC), if and only if \( d(v_i) \) is reachable from \( v_j \) by a path of length 1. Otherwise, the vertices, if connected, are said to be non-MSC (NMSC). The degree complement of an NMSC pair of vertices \((v_i, v_j)\) in \( A \) is written as \( d'(v_i, v_j) = (k_1, k_2) \), where \((v_i, v_j) = k_1, d'(v_i) = k_2 \).

Definition 14: A subgraph \( A_i \) of \( A \) is said to be MSC, if and only if every possible pair of vertices in \( A_i \) is MSC. The subgraph \( A_i \) is said to be maximally MSC (MMSC) if there does not exist any vertex outside of \( A_i \), which is MSC with all the vertices of \( A_i \).

Definition 15: Let \((v_i, v_j)\) be an NMSC pair of vertices in \( A \). Then splitting \( A \) into two subgraphs \( A_i \) and \( A_j \) such that \( A_i \) contains the vertex \( v_i \) and \( A_j \) contains the vertex \( v_j \) is to obtain two subgraphs \( A_i \) and \( A_j \) from \( A \) such that \( A_i \) contains all the vertices of \( A \) except \( v_j \) and \( A_j \) contains all the vertices of \( A \) except \( v_i \), both \( A_i \) and \( A_j \) having all the existing edges of
A connecting between relevant pairs of vertices. Obviously, 
\( A_1 \subseteq A; A_2 \subseteq A \).

**Definition 16:** For any two distinct NMSC pairs of vertices 
\((v_{11}, v_{12})\) and 
\((v_{21}, v_{22})\) in \( A \), let 
\( d(v_{11}, v_{12}) = (k_1, k_2) \) and 
\( d(v_{21}, v_{22}) = (r_1, r_2) \). If \( k_1 + k_2 > r_1 + r_2 \), then an 
ordering of the degree complements of the pairs of vertices 
can be made as 
\( d(v_{11}, v_{12}) \geq d(v_{21}, v_{22}) \), whereas if 
\( k_1 + k_2 < r_1 + r_2 \), the ordering of the degree complements 
of the pairs of vertices can be made as 
\( d(v_{21}, v_{22}) \geq d(v_{11}, v_{12}) \). If, however, \( k_1 + k_2 = r_1 + r_2 \), the 
ordering can be made either as 
\( d(v_{11}, v_{12}) \geq d(v_{21}, v_{22}) \) or as 
\( d(v_{21}, v_{22}) \geq d(v_{11}, v_{12}) \). This kind of ordering \((\geq)\) that can 
be established among degree complements of different 
NMSC pairs of vertices in an undirected graph is called 
the magnitude ordering of the degree complements of the 
pairs of vertices.

**Theorem 10:** Let \( A \) be an undirected graph, and let 
\((v_i, v_j)\) be an NMSC pair of vertices in \( A \). Let the graph \( A \) be split 
around \((v_i, v_j)\) into two subgraphs \( A_1 \) and \( A_2 \) and let this 
process of splitting around NMSC pairs of vertices be iteratively 
applied to \( A_1 \) and \( A_2 \) and to all their subgraphs until in the 
resulting subgraphs there exist no more NMSC pairs of vertices. 
The final set of these subgraphs then includes all the 
MMS subgraphs of \( A \).

**Theorem 11:** Let \( A \) be an undirected graph, and let 
\((v_i, v_j)\) be an NMSC pair of vertices of \( A \) having the highest degree 
complement in the magnitude ordering. If now the graph 
\( A \) is split around \((v_i, v_j)\) into two subgraphs \( A_1 \) and \( A_2 \), then in the 
resulting subgraphs the number of NMSC pairs of vertices will always be 
less than that when \( A \) will be split into subgraphs around any other NMSC 
pairs having non-highest degree complement in the magnitude 
ordering.

**Theorem 12:** In the process of successively splitting an 
undirected graph \( A \) into subgraphs around NMSC pairs of 
vertices, let \( A_i \) and \( A_j \) be any two obtained at 
different stages such that \( A_j \subseteq A_i \), but \( A_i \) is not derived 
from \( A_j \). Then, in finding only MMS subgraphs, the 
subgraph \( A_i \) may be discarded in general.

4 **Algorithm development**

The developed zero-aliasing space compression approach 
consists of a set of algorithms: The first algorithm is for 
computing set of incompatible pairs [50, 46] of response 
data outputs of the MUT for logic AND/NAND, OR/ 
NOR and XOR/XNOR, while the second and third algorithms 
are for finding their MCCs from the incompatible 
pairs based on the two different graph theoretic approaches 
as discussed. The final algorithm constructs the space compaction 
networks using the information of the generated 
maximal compatibility classes. All the different algorithms 
are presented below.

4.1 **Algorithm 1**

This algorithm computes all incompatible pairs of the MUT 
output lines (pairs that do not produce 100% fault coverage) 
for logic AND/NAND, OR/NOR and XOR/XNOR.

**Step 1.** Get the total number of output lines of the MUT.

**Step 2.** Generate all possible combinations \((v_i, v_j)\) of the 
MUT output lines, taking two at a time, and store all pairs of 
the output lines \((v_i, v_j)\).

**Step 3.** Select the first pair from the list of all combined 
output lines \((v_i, v_j)\). 

**Step 4.** Merge the selected pair of output lines \((v_i, v_j)\) using 
logic gates AND/NAND, OR/NOR and XOR/XNOR, 
respectively, using only one type of logic gate at a time.

**Step 5.** Add a new output line to the original MUT 
corresponding to the outputs \((v_i, v_j)\), one at a time.

**Step 6.** Discard the output lines \((v_i, v_j)\) from the original 
MUT and generate a new modified MUT.

**Step 7.** Inject stuck-at logic faults into the newly 
generated MUT and apply test patterns.

**Step 8.** If the FC \(< 100\%\), then store the output pair \((v_i, v_j)\) 
in the incompatible pairs database of logic AND/NAND, 
OR/NOR and XOR/XNOR, respectively.

**Step 9.** Delete the pair just considered, from the list of all 
combined output lines \((v_i, v_j)\), and select the next pair.

**Step 10.** Go to step 4 and continue until all pairs are 
exhausted.

4.2 **Algorithm 2**

This algorithm is an implementation of the well-known 
graph theory technique of Bron and Kerbosch for computing 
all cliques in an undirected graph [51, 53]. We employ this as one graph theoretic approach for computing 
the MCCs of response data outputs of the MUT for logic families 
AND/NAND, OR/NOR and XOR/XNOR. In the 
process, we use information of the incompatible pairs of 
the MUT output lines as generated by applying Algorithm 1 as given above. The algorithm is now described 
as follows.

**Step 1.** Calculate the total number of vertices in the undirected 
graph.

**Step 2.** Find the connected diagonal elements of the graph.

**Step 3.** Select a candidate point.

**Step 4.** Merge the selected candidate to a set called 
compsub, which is to be extended by a new point or 
shrinked by a point on travelling along a branch of the back 
tracking tree.

**Step 5.** Generate a new set called candidates, which is the 
set of all points that will in due time serve as an extension to 
the present configuration of compsub.

**Step 6.** Create another set called not, which is the set of all 
points that, at an earlier stage, already served as an extension 
of the present configuration of compsub and are now 
explicitly excluded.

**Step 7.** Remove all points not connected to the selected 
candidate, keeping the old sets intact.

**Step 8.** Call the extension operator to perform on the 
newly generated sets.

**Step 9.** Remove the selected candidate from the compsub 
and add it to the old set not after returning.

4.3 **Algorithm 3**

This algorithm also finds the MCCs from the same set of 
incompatible pairs of the MUT outputs as obtained by 
Algorithm 1 above, based on the implementation of the 
other graph theoretic approach as outlined previously 
[54–56]. The algorithm is provided below.

**Step 1.** From the undirected graph \( A \) (compatibility graph) 
representative of the incompatible pairs, find the magnitude 
ordering of degree complements of the NMSC pairs of 
outputs of the MUT in \( A \).

**Step 2.** Select an NMSC pair of outputs \((v_i, v_j)\) in \( A \), where 
\((v_i, v_j)\) has the highest degree complement in the magnitude
ordering. If more than one pair of outputs has the highest degree complement, select any one of these output pairs \((v_i, v_j)\). Split \(A\) around \((v_i, v_j)\) into two subgraphs \(A_i\) and \(A_j\) such that \(A_i\) contains all the outputs (vertices) of \(A\) except \(v_i\) and \(A_j\) contains all the outputs (vertices) of \(A\) except \(v_j\).

(a) Consider the subgraph \(A_i\); check if there exists a subgraph \(A_k\) from which \(A_i\) is not derived, containing \(A_j\). If so, discard the subgraph \(A_i\); if not, take \(A_i\) and go to step 1.

(b) Consider the subgraph \(A_j\); check if there exists a subgraph \(A_m\) from which \(A_j\) is not derived, containing \(A_i\). If so, discard the subgraph \(A_j\); if not, take \(A_j\) and go to step 1.

Step 3. Follow steps 1 and 2 iteratively until in all the resulting subgraphs there does not exist any NMSC pair of outputs. The final set of subgraphs then includes all the MMSC subgraphs (MCCs) of \(A\).

Step 4. In the set of subgraphs obtained after step 3, check if any subgraph is contained in another subgraph for possible cancellation of non-MMSC subgraphs. The resultant set, after cancellation, if any, gives all the MMSC subgraphs (MCCs) of \(A\).

4.4 Algorithm 4

This algorithm utilises the knowledge of MCCs as obtained from either Algorithm 2 or Algorithm 3 to construct zero-aliasing space compactors for the MUT. The final algorithm is now given as follows.

Step 1. Define the possible maximum number of stages in the space compaction trees at the MUT output.

Step 2. Get the total number of output lines in the MUT. Continue the following steps until there is only a single output line (possibly).

Step 3. Find the sets of all MCCs from the MUT for logic AND/NAND, OR/NOR and XOR/XNOR, employing Algorithm 2 or Algorithm 3.

Step 4. Select an MCC with large (possibly largest) number of output lines from the sets of MCCs. Select the next large class during subsequent iteration, if 100% FC is not achieved in the previous iteration from the same MUT.

Step 5. Merge selected output lines of the MCC, using appropriate logic gates (AND/NAND, OR/NOR or XOR/XNOR).

Step 6. Add a new output line corresponding to the selected merged outputs of MCC.

Step 7. Discard those MUT output lines that are already used in MCC.

Step 8. Search another MCC from the remaining output lines.

Step 9. Merge the selected output lines in MCC using appropriate logic gates.

Step 10. Add a new output line corresponding to the selected merged outputs of MCC.

Step 11. Discard the output lines that are already used in MCC.

Step 12. Go to step 8 as long as there are MCCs in the sets, and enough output lines.

Step 13. Find all the remaining output lines that do not belong to any of the selected MCCs.

Step 14. Merge all these remaining lines with XOR/ XNOR gate.

Step 15. Add a new output line corresponding to these selected merged outputs.

Step 16. Inject stuck-at logic faults into the newly generated MUT (original MUT with COMPACTOR hardware).

Step 17. Compute FC by applying input test patterns.

Step 18. If FC = 100%, then replace the old MUT with the new MUT and go to step 2 for generating the next stage of the compactor.

Step 19. If FC < 100%, then merge all the remaining output lines with two-input XOR/XNOR gates, two output lines at a time.

Step 20. Add new output lines corresponding to the selected merged outputs.

Step 21. Inject stuck-at logic faults into the newly generated MUT (original MUT with COMPACTOR hardware).

Step 22. Compute FC by applying input test patterns.

Step 23. If FC < 100%, then continue to work on the same MUT. Go to step 4 for selecting a new MCC.

Step 24. If FC = 100%, then replace the old MUT with the new MUT, and go to step 2 for computing the next and subsequent stages of the compactor.

5 Experimental results

Extensive simulations runs were conducted to demonstrate the feasibility of the proposed zero-aliasing space compaction scheme using ISCAS 85 combinational benchmark circuits and ISCAS 89 full-scan sequential benchmark circuits. In our design experimentation, we used ATALANTA [60] (fault simulation program developed at the Virginia Polytechnic Institute and State University) as test generation tool to produce the fault-free output sequences needed to construct our space compactor circuits and to test the benchmark circuits using reduced test sets. We also used FSIM fault simulation program [61] that generates pseudorandom test sets, and COMPACTEST [62] program to generate the reduced test sets that detect most detectable single stuck-line faults for all the benchmark circuits. For each circuit, we determined the FC without the compactor, FC with the compactor, number of test vectors used to construct the compaction tree, simulation CPU time, number of test vectors applied, hardware overhead, and compaction ratio by running ATALANTA and FSIM programs on a

Fig. 2 Compactor circuit 1 for c432

Fig. 3 Compactor circuit 2 for c432
SUN SPARC 5 workstation, and COMPACTEST program on IBM AIX machine.

Figs. 2 and 3 illustrate two separate three-stage zero-aliasing compaction networks for c432 benchmark circuit. For designing each and every stage of a compactor, Algorithm 1 is first run on the original MUT for generating all incompatible pairs of output lines. Some results for logic AND/NAND and OR/NOR for circuit c432 are shown in Table 1. However, there is no incompatible pair for logic XOR/XNOR for this circuit. These incompatible pairs of output lines are next used to determine MCCs by employing Algorithm 2 or Algorithm 3. Table 2 depicts some of the results obtained from an implementation of those algorithms. Table 3 incorporates two different compaction trees generated from the data in Table 2 by applying Algorithm 4, with maximal compaction ratio. After generating the first stage of the compactor, a new MUT is formed, which is used for generating the second stage of the compactor by following the same sequence of computations. As can be seen from Figs. 2 and 3, there are only two outputs (434 and 435) after the second stage, which are combined

### Table 1: Incompatible pairs for logic AND/NAND and OR/NOR

<table>
<thead>
<tr>
<th>AND/NAND</th>
<th>OR/NOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>out1, out2</td>
<td>out1, out2</td>
</tr>
<tr>
<td>223, 421</td>
<td>223, 421</td>
</tr>
<tr>
<td>329, 421</td>
<td>223, 432</td>
</tr>
<tr>
<td>329, 421</td>
<td>329, 432</td>
</tr>
<tr>
<td>370, 421</td>
<td>329, 432</td>
</tr>
<tr>
<td>421, 430</td>
<td>329, 432</td>
</tr>
<tr>
<td>421, 431</td>
<td>370, 421</td>
</tr>
<tr>
<td>421, 432</td>
<td>370, 432</td>
</tr>
<tr>
<td>430, 431</td>
<td>421, 430</td>
</tr>
<tr>
<td>430, 432</td>
<td>421, 431</td>
</tr>
<tr>
<td>431, 432</td>
<td>421, 432</td>
</tr>
<tr>
<td>430, 432</td>
<td>430, 432</td>
</tr>
<tr>
<td>431, 432</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2: Maximal compatibility classes

<table>
<thead>
<tr>
<th>AND/NAND</th>
<th>OR/NOR</th>
<th>XOR/XNOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>223, 329, 370, 431</td>
<td>223, 329, 370, 431</td>
<td></td>
</tr>
<tr>
<td>223, 329, 370, 432</td>
<td>421</td>
<td></td>
</tr>
<tr>
<td>421</td>
<td></td>
<td></td>
</tr>
<tr>
<td>432</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3: Compaction trees with maximal compaction ratio

<table>
<thead>
<tr>
<th>Compaction tree: 1</th>
<th>Compaction tree: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>433 = AND (223, 329, 370, 432)</td>
<td>433 = OR (223, 329, 370, 431)</td>
</tr>
<tr>
<td>434 = XOR (430, 431)</td>
<td>434 = XOR (421, 430)</td>
</tr>
<tr>
<td>435 = OR (421, 433)</td>
<td>435 = XOR (432, 433)</td>
</tr>
<tr>
<td>436 = XOR (434, 435)</td>
<td>436 = XOR (434, 435)</td>
</tr>
<tr>
<td>OUTPUT (436)</td>
<td>OUTPUT (436)</td>
</tr>
</tbody>
</table>

### Table 4: Simulation results of ISCAS 85 combinational benchmark circuits using ATALANTA without space compactors

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Applied test vectors</th>
<th>Number of faults injected</th>
<th>Number of outputs</th>
<th>Fault coverage, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>c17</td>
<td>7</td>
<td>22</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>c432</td>
<td>76</td>
<td>520</td>
<td>7</td>
<td>100.00</td>
</tr>
<tr>
<td>c499</td>
<td>66</td>
<td>750</td>
<td>32</td>
<td>100.00</td>
</tr>
<tr>
<td>c880</td>
<td>107</td>
<td>942</td>
<td>26</td>
<td>100.00</td>
</tr>
<tr>
<td>c1355</td>
<td>105</td>
<td>1566</td>
<td>32</td>
<td>100.00</td>
</tr>
<tr>
<td>c1908</td>
<td>184</td>
<td>1870</td>
<td>25</td>
<td>100.00</td>
</tr>
<tr>
<td>c2670</td>
<td>182</td>
<td>2630</td>
<td>140</td>
<td>100.00</td>
</tr>
<tr>
<td>c3540</td>
<td>253</td>
<td>3291</td>
<td>22</td>
<td>100.00</td>
</tr>
<tr>
<td>c5315</td>
<td>197</td>
<td>5291</td>
<td>123</td>
<td>100.00</td>
</tr>
<tr>
<td>c6288</td>
<td>53</td>
<td>7710</td>
<td>32</td>
<td>100.00</td>
</tr>
<tr>
<td>c7552</td>
<td>376</td>
<td>7419</td>
<td>108</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### Table 5: Simulation results of ISCAS 85 combinational benchmark circuits using FSIM without space compactors

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Applied test vectors</th>
<th>Number of faults injected</th>
<th>Number of outputs</th>
<th>Fault coverage, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>c17</td>
<td>32</td>
<td>22</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>c432</td>
<td>544</td>
<td>520</td>
<td>7</td>
<td>100.00</td>
</tr>
<tr>
<td>c499</td>
<td>1312</td>
<td>750</td>
<td>32</td>
<td>100.00</td>
</tr>
<tr>
<td>c880</td>
<td>5480</td>
<td>942</td>
<td>26</td>
<td>100.00</td>
</tr>
<tr>
<td>c1355</td>
<td>2124</td>
<td>1566</td>
<td>32</td>
<td>100.00</td>
</tr>
<tr>
<td>c1908</td>
<td>29472</td>
<td>1870</td>
<td>25</td>
<td>100.00</td>
</tr>
<tr>
<td>c2670</td>
<td>6378144</td>
<td>2630</td>
<td>140</td>
<td>100.00</td>
</tr>
<tr>
<td>c3540</td>
<td>38848</td>
<td>3291</td>
<td>22</td>
<td>100.00</td>
</tr>
<tr>
<td>c5315</td>
<td>4576</td>
<td>5291</td>
<td>123</td>
<td>100.00</td>
</tr>
<tr>
<td>c6288</td>
<td>128</td>
<td>7710</td>
<td>32</td>
<td>100.00</td>
</tr>
<tr>
<td>c7552</td>
<td>100000000</td>
<td>7419</td>
<td>108</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### Table 6: Simulation results of ISCAS 85 combinational benchmark circuits using COMPACTEST without space compactors

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Applied test vectors</th>
<th>CPU simulation time, s</th>
<th>Number of outputs</th>
<th>Fault coverage, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>c17</td>
<td>4</td>
<td>0.01</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>c432</td>
<td>44</td>
<td>5.09</td>
<td>7</td>
<td>99.430</td>
</tr>
<tr>
<td>c499</td>
<td>65</td>
<td>8.36</td>
<td>32</td>
<td>98.990</td>
</tr>
<tr>
<td>c880</td>
<td>30</td>
<td>1.85</td>
<td>26</td>
<td>100.00</td>
</tr>
<tr>
<td>c1355</td>
<td>96</td>
<td>2.54</td>
<td>32</td>
<td>99.480</td>
</tr>
<tr>
<td>c1908</td>
<td>137</td>
<td>13.39</td>
<td>25</td>
<td>99.230</td>
</tr>
<tr>
<td>c2670</td>
<td>138</td>
<td>96.78</td>
<td>140</td>
<td>95.520</td>
</tr>
<tr>
<td>c3540</td>
<td>110</td>
<td>278.45</td>
<td>22</td>
<td>95.920</td>
</tr>
<tr>
<td>c5315</td>
<td>55</td>
<td>35.74</td>
<td>123</td>
<td>98.890</td>
</tr>
<tr>
<td>c6288</td>
<td>16</td>
<td>68.16</td>
<td>32</td>
<td>99.330</td>
</tr>
<tr>
<td>c7552</td>
<td>85</td>
<td>164.23</td>
<td>108</td>
<td>98.440</td>
</tr>
</tbody>
</table>
by an XOR gate to obtain the single output (436) at the final stage.

The complete results on ISCAS 85 combinational and ISCAS 89 full-scan sequential benchmark circuits are listed in Tables 4–10. The circuits with the compressors were tested with the same fault injection and test vectors for all the simulation programs FSIM, COMPACTEST and ATALANTA. Tables 4–6 show that the FC obtained from ISCAS 85 benchmark circuits without employing any space compactor. For almost all the circuits (viz. at the MUT outputs, before the use of space compression hardware), the FC is either 100% or very close to it (using ATALANTA, FSIM and COMPACTEST), but the number of output lines varies from 2 to 140. However, the results in Tables 7–9 show that 100% FC is not only achieved in all cases with the space compactors regardless of the simulation programs (ATALANTA and FSIM), but also the number of output lines equals one in all cases with the exception of only a single circuit (c2670). Similar results were also obtained from ISCAS 89 full-scan sequential benchmark circuits, as shown in Table 10 using the designed space compactors. The FC is considered 100%, if the faults detected at the MUT outputs and COMPACTOR outputs are the same, thereby implying that the COMPACTOR did not introduce any information loss.

Fig. 4 gives the estimates of the hardware overhead for ISCAS 85 combinational benchmark circuits using ATALANTA simulation program. For estimating the hardware overhead, we used the ratio of the weighted gate count metric, viz. average fanins multiplied by the number of

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Applied test vectors</th>
<th>Number of faults injected</th>
<th>Number of outputs</th>
<th>Fault coverage, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>c17</td>
<td>10</td>
<td>22</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c432</td>
<td>124</td>
<td>520</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c499</td>
<td>169</td>
<td>750</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c880</td>
<td>223</td>
<td>940</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c1355</td>
<td>220</td>
<td>1566</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c1908</td>
<td>313</td>
<td>1870</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c2670</td>
<td>496</td>
<td>2630</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>c3540</td>
<td>270</td>
<td>3291</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c5315</td>
<td>692</td>
<td>5291</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c6288</td>
<td>65</td>
<td>7710</td>
<td>1</td>
<td>100.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Applied test vectors</th>
<th>Number of faults injected</th>
<th>Number of outputs</th>
<th>Fault coverage, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>c17</td>
<td>45</td>
<td>22</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c432</td>
<td>752</td>
<td>520</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c499</td>
<td>10 929 363</td>
<td>750</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c880</td>
<td>97 055 712</td>
<td>940</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c1355</td>
<td>100 000 000</td>
<td>1566</td>
<td>1</td>
<td>94.994</td>
</tr>
<tr>
<td>c1908</td>
<td>96 283 712</td>
<td>1870</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c2670</td>
<td>100 000 000</td>
<td>2630</td>
<td>3</td>
<td>98.889</td>
</tr>
<tr>
<td>c3540</td>
<td>301 824</td>
<td>3291</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c5315</td>
<td>1 316 134 912</td>
<td>5291</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>c6288</td>
<td>224</td>
<td>7710</td>
<td>1</td>
<td>100.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit name</th>
<th>Number of faults injected</th>
<th>Fault coverage, %</th>
<th>Number of outputs</th>
<th>Fault coverage, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>s27</td>
<td>32</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s208</td>
<td>214</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s298</td>
<td>306</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s344</td>
<td>340</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s349</td>
<td>348</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s382</td>
<td>397</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s713</td>
<td>921</td>
<td>100.00</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>s838</td>
<td>187</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s953</td>
<td>81</td>
<td>100.00</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>s1196</td>
<td>1025</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>s1238</td>
<td>1035</td>
<td>100.00</td>
<td>1</td>
<td>100.00</td>
</tr>
</tbody>
</table>
gates or gate count of the COMPACTOR and that of the total circuit composed of the MUT and COMPACTOR. It can be seen from the above figure that the hardware overhead of the compressor for different benchmark circuits is as small as 0.71–8.99%, and equals 14.29% for circuit c17 in the case of ATALANTA. Fig. 5, on the other hand, gives compaction ratio for ISCAS 89 full-scan sequential benchmark circuits using ATALANTA.

The calculated FC and hardware overhead are evidently different when different simulation programs were employed. This is because of the variations of the embedded algorithms in the respective computer-aided design programs. For instance, the compression, fault injection, and fault analysis algorithms employed in ATALANTA or COMPACTEST are obviously completely different from those used in FSIM.

6 Concluding remarks

This paper revisits zero-aliasing space compaction problem of response data outputs of MUT with application specifically targeted towards digital embedded cores-based SOCs. The technique utilises conventional switching theory concepts, viz. those of cover table, frequency ordering and compatibility relation together with those of strong and weak compatibilities of response data outputs, in the selection of specific gates for merger of an arbitrary but optimal number of output bit streams from the MUT. This is novel in the sense that zero-aliasing is achieved without any prior modification of the original circuit (MUT), as all the modification for testing is done in software mode, whereas maximal compaction is realised in most cases in reasonable time utilising some simple heuristics. The technique, illustrated with details of design of space compactors for ISCAS 85 combinational and ISCAS 89 full-scan sequential benchmark circuits with ATALANTA, FSIM and COMPACTEST simulation programs, confirms the usefulness of the suggested approach, its simplicity, resulting low area overhead, and full FC for single stuck-line faults, making it suitable in a VLSI design environment as BIST support hardware. It might be fair to mention here that the test vectors used for detecting single stuck-line faults in digital BIST these days, irrespective of whether the circuits are combinational or full-scan sequential, are either deterministic, minimal or non-minimal, complete, or simply reduced but not necessarily complete, or pseudorandom [1–3]. If minimal complete test sets are available (viz. MinTest for ISCAS 85 and ISCAS 89 full-scan circuits [2]), these obviously would be the best choice in testing any circuit for complete FC using minimal time and resources.

On the other hand, programs such as ATALANTA [60] or COMPACTEST [62] as used in the paper generate reduced test sets that detect most single stuck-line faults for both ISCAS 85 and ISCAS 89 full-scan circuits (the programs, in general, not providing complete sets of tests), whereas FSIM [61] generates pseudorandom test vectors which give very good fault coverage, not necessarily 100%, for most of the ISCAS benchmark circuits. All these types of test vectors are absolutely compatible with the approach developed in this paper. Only restriction is, however, that we are working under the constraint of a single stuck-fault model. Notwithstanding this, the proposed theoretical framework, although was simulated on the assumption of single stuck-line faults, is amenable to modification to take care of multiple fault situations as well, if programs are available to simulate multiple stuck-line faults. But, multiple faults are extremely difficult to analyse in view of their sheer number. Besides, modern day circuits are extremely reliable not to exhibit simultaneous multiple stuck-line faults, although there are occasions where these faults need to be considered. However, in this work, we did not, as in the case of most other studies. Besides, we do not consider faults such as stuck-open faults, short faults, bridging faults, intermittent faults or transient faults in our analysis, the last two types being very difficult to investigate theoretically, requiring discrete or continuous Markov modelling. In the sequel, it is evident from the experimental results that the suggested approach, although relies on restricted use of heuristics, still could be considered simple and robust enough in its design methodology for single stuck-line faults of the
MUT. With advances in computational resources, evidently this heuristic space compaction algorithm might be improved upon for better efficiency in respect of time and storage.

7 Acknowledgments

The authors would like to thank the anonymous reviewers for their very valuable comments that immensely helped in the preparation of the revised version of the paper. Thanks are also due to the guest editor and to Ms Ann Thompson of the IET Editorial office for their immense help and kind understanding. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant A 4750. A preliminary version of this paper was presented as an invited contribution at the 3rd International Conference on Computers and Devices for Communication (CODEC), Calcutta, India, 18–20 December 2006.

8 References


56 Lee, S.Y.: ‘On a family of novel clique detection algorithms and some studies on their performance characteristics and order of complexities in the context of existing algorithms’, PhD thesis, National Chiao Tung University, Department of Electronics, Hsinchu, Taiwan, ROC, 1982


