Universität Potsdam - Institut für Informatik

Output Space Compaction for Testing and Concurrent Checking

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)
in der Wissenschaftsdisziplin Technische Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

von
Markus Seuring

Potsdam, im Oktober 2000

Contents

1 Preface

Testing digital circuits is an important part of the efforts, which are necessary for their design, their production, and, finally, their usage. As the electronics industry evolved from discrete components, through the first integrated circuits containing only a single gate or flipflop, to more and more increasingly high levels of integration, the demands on safety, availability, reliability, and accuracy require the continual development of more sophisticated testing and checking techniques. Due to the increasing number of circuit outputs and the increasing size of test sets for thousands or millions of transistors, large quantities of test response data must be compared with the fault-free responses. Therefore, for highly integrated circuits bit-by-bit comparison is not feasible. To handle this problem, the output values of the circuit are usually compacted and compared with the compacted test responses of the fault-free circuit. Since any compaction means a loss of information, output compaction often results in fault masking, i. e., erroneous values at the functional outputs are compacted to the fault-free compacted test responses, so that no error can be observed at the outputs of the compactor.

The approach of output compaction can be divided into two basic concepts: Time compaction and space compaction. Time compaction means to capture the output responses over several time or test steps and to compute from the test responses a signature. It uses sequential logic to combine the test responses during one clock cycle with the test responses from previous clock cycles. After the end of the test the obtained signature is compared to the fault-free signature. In contrast, a space compactor reduces the number of streams of response bits which must be observed. Normally, it uses combinational logic to merge a large number of response streams into a smaller number of streams. So it reduces the number of bits of test responses that are generated every clock cycle. On the other hand, if no time compaction is added, then the amount of compacted response data increases according to the length of the test. Furthermore, space compaction can also be used for concurrent checking. Concurrent checking provides fault detection during the regular operation of a circuit.

There are also techniques for output compaction which can be conceived as a mixture of both concepts. Often test responses observed at the functional outputs are compacted by a space compactor and the compacted response streams feed a time compactor. Since in most cases the objectives for the design of space compactors are independent of the existence of an additional time compaction step, this work only focuses on space compaction and its usage for testing or concurrent checking.

Tests of digital circuits are performed by applying a set of precomputed input vectors to the circuit. Such a test can be carried out for design, diagnosis or after the production of a circuit. Furthermore, if the circuit is already in use, it can be tested by utilizing its initialization phase or any other time interval while the circuit is idle or which is explicitly reserved for testing. In many cases, the precomputed test vectors are generated targeting a certain class of faults in the circuit and may additionally detect non-modeled faults.

As mentioned above, while the precomputed test set is applied to the inputs of the circuit, a space compactor can be used to compact the test responses. The test results are compared with the results of a fault-free circuit. If any difference can be observed, then the tested circuit is faulty. Figure 1.1

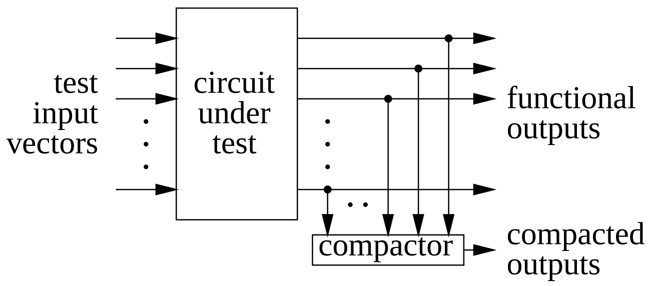


Figure 1.1: Circuit under test with output compactor

schematically shows a circuit under test with an output compactor.

Usually, a circuit is tested in an automatic test equipment (ATE) before it is delivered to the customers. In this case, the input vectors of a test are generated by the ATE and applied to the circuit. The ATE accesses the responses of the circuit, and these responses are compared with the fault-free test responses. In order to reduce the amount of test response data, the size of memory required for storing the fault-free test responses, and the number of comparisons, the output values of the circuit can be compacted as described in this work.

For several years the costs of automatic test equipments have been increasing and issues related to timing accuracy and support of diagnosis have become more important [43], so that other solutions are used in combination with ATEs or even instead of ATEs. Such a well known solution is built-in self test (BIST).

BIST requires a special test mode. Additional logic is used to switch between this mode and the normal operation mode of the circuit. The test input vectors are generated on the chip and also the test responses are compacted there. In several approaches the fault-free results are also stored on the chip. In this case, after all test vectors have been applied and the responses have been compared, a test response at the chip output pins indicates whether a fault has been detected or not. If the fault-free results are not stored on the chip, then the test responses are compacted, so that only a small number of bits must be processed by the environment. Again, space compactors can be used for the compaction of the test responses.

Although in the following only the compaction of functional outputs is described, space compactors can also be used to compact the test responses derived from observation points. Observation points are often inserted to improve the testability of the circuit under test [40, 28, 22, 66]. If a space compactor is added, the quantity of test data obtained from the observation points can be reduced [30]. It is also possible to compact the values of the functional outputs and the observation points jointly using a single space compactor.

A different technique for detecting faults of circuits is concurrent checking. It is also called on-line testing. The basic idea of concurrent checking is to monitor the function of a circuit during regular operation. Thus, the circuit does not switch to a special test mode.

For concurrent checking two additional circuits are required: coder and code checker. During the regular operation the coder computes redundant information from the circuit inputs, so that the vector, which consists of the output of the coder and of the circuit under check, is encoded into some specific error code. In order to check if the circuit and the coder produce code words, an additional circuit called code checker is used. If the presence of a fault modifies the circuit under check or the coder so that a

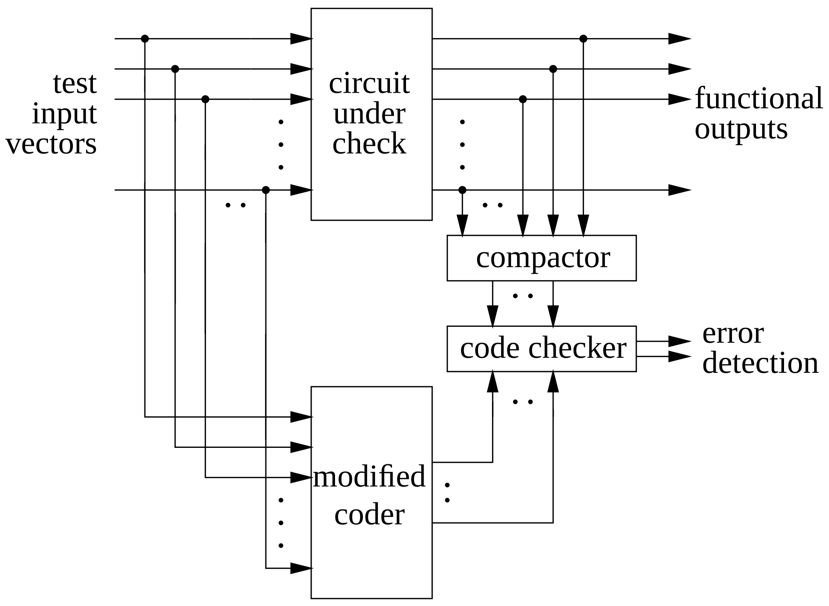


Figure 1.2: Concurrent checking combined with output compaction

word is generated, which does not belong to the code, then the error is detected by the code checker.

In most cases, the number of bits of the code words is at least as high as the number of the functional outputs of the circuit under check. Then the size of the code checker may be very large according to the number of its inputs. To reduce the size of the code checker, a space compactor can be inserted at the functional outputs to obtain an error code consisting of shorter words. This is shown in Figure 1.2.

In many cases the usage of short code words considerably reduces the size of the code checker, which must be modified according to the new code. Then the reduction of the area required for modified coder and code checker predominates the additional area overhead caused by the space compactor. The disadvantage of this method is, that faults of the circuit may be detected with some delay or may even be totally masked.

About this Work

The objective of this thesis is to provide new space compaction techniques for testing or concurrent checking of digital circuits. In particular, the work focuses on the design of space compactors that achieve high compaction ratio and cause minimal loss of testability of the circuits. The introduction of each chapter briefly explains how the space compactors can be included in environments for testing or concurrent checking.

Basically, the thesis is divided into two parts: Chapter 2 and 3 consider a circuit, which is described by a netlist of gates, so that an appropriate linear space compactor can be derived from an analysis of the circuit structure. In Chapter 4, it is assumed, that the structure of a circuit is unknown, but a precomputed test set and the fault-free test responses of the circuit are given. Under this condition space compactors are designed without knowledge of the underlying fault model of the test set. Each chapter starts with an introduction into the basics of the corresponding subject followed by a brief summary of the state of the art.

In the first part of this work the proposed compactor design is based on the knowledge of the circuit structure. In particular, the circuit is described at the logical level by a netlist of gates. The netlist is processed by an algorithm, which analyzes the propagation of single stuck-at faults from an arbitrary fault site to different functional outputs of the circuit. Several analysis algorithms are introduced and discussed in this work. The results of the analysis are used to design a linear space compactor with minimized fault masking. The generation of the linear space compactor allows an arbitrary choice of the number of compacted outputs and therefore also an arbitrary compaction ratio. In Chapter 2, design methods are discussed, which are based on various new analysis algorithms for combinational circuits. Analogically, such methods are described for sequential circuits in Chapter 3.

Both chapters include the presentation of experimental results. The experiments were performed to determine the accuracy of the various heuristic algorithms. The analysis algorithms achieving the highest accuracy were chosen to design linear space compactors according to the analysis results. Then further experiments are used to investigate the frequency of fault masking caused by these space compactors. For testing and concurrent checking experimental results were derived by simulations of several benchmark circuits in the presence of single stuck-at faults or transition faults.

The experimental results show, that for the investigated benchmark circuits the proposed approaches lead to compactors with high compaction ratio, which cause minimal or no loss of testability. Since the complexity of each design procedure is linear with respect to the number of gates of the circuit, the determination of compactors for the investigated benchmark circuits required only a few seconds and the presented design methods are applicable to large circuits.

Chapter 2 includes comparisons with other approaches which were proposed for the output compaction of combinational circuits. The comparisons show, that in most cases the best results are obtained by the methods presented in this work. In Chapter 3 no comparisons with other approaches are given, since it introduces the first structural approach for output compaction for sequential circuits, which takes into account fault propagation from faulty states to the outputs of the circuit.

In Chapter 4, the underlying conditions for the design of space compactors are different in many ways compared to those of the previous chapters. It is assumed that the structure of the circuit is unknown. For example, in practice often intellectual property (IP) cores are taken from a core vendor to implement special functionalities of a large chip without time consuming and expensive development. Such an IP core can not be easily tested, since normally the structural information of this circuit is not revealed to protect the intellectual property. To solve this issue, often the core vendor gives an appropriate set of test patterns. The space compaction approach described in Chapter 4 is applicable to such circuits and requires only the knowledge of the fault-free test responses for a precomputed test set. Furthermore, no additional information about the underlying fault model is necessary.

The proposed compactor design guarantees zero-aliasing with respect to the precomputed test set. The space compactors are non-linear or only partly linear. Since the compactors are zero-aliasing, the test responses can not be compacted to an arbitrary number of compacted outputs. Lower bounds on the number of compacted outputs are proven in Chapter 4. The smaller the number of distinct test responses is, the smaller is the lower bound. Therefore, this approach is especially applicable to circuits for which small test sets are available, because in this case the proposed design results in space compactors with high compaction ratio.

The basic design method produces space compactors with the minimal number of compacted outputs according to the proven bound. Thus, it achieves the optimum compaction ratio. In order to further reduce the number of compacted outputs, a method is discussed which is based on two different compaction functions. For each test response the two compaction functions are used in two time steps which may increase the testing time. The determination of the compaction functions of the two time steps is based on orthogonal transmission functions.

For a circuit with a large number of functional outputs the synthesis of the proposed space compactor design is not feasible because the higher the number of inputs of the compactor is, the more complex is the function. A method to synthesize such a space compactor is introduced in Chapter 4. Then the presentation of experimental results obtained for several benchmark circuits follows. The area overhead of each space compactor of these circuits is given and the dependency between the number of distinct fault-free test responses and size of the compactor is discussed. The results show, that the compactors for the investigated circuits produce only low area overhead.

The last chapter reviews the major contributions of the thesis and briefly discusses possible extensions of this work. The appendix briefly introduces the simulator `Minos` which was implemented for the experimental investigations of this thesis.

Acknowledgments

Most of the work presented in this dissertation has been done at the Fault Tolerant Computing Group of the Institute of Computer Science at the University of Potsdam. The fourth chapter has been done in part during my stays at the Department of Electrical and Computer Engineering at the Duke University in Durham, North Carolina.

First of all I want to thank my supervisor Prof. Dr. Michael Gössel, head of the Fault Tolerant Computing Group of the Institute of Computer Science, for his patient guidance and support during all the time this work was done, for his cooperation, and also for the freedom he gave me in my research activities.

I want to thank Prof. Dr. Egor Sogomonyan from the Institute of Control Sciences at the Russian Academy of Sciences in Moscow for many helpful discussions on Chapter 2.

Further, I would like to thank Prof. Dr. Krishnendu Chakrabarty for the opportunity to work at the Department of Electrical and Computer Engineering at the Duke University in Durham and for his collaboration on Chapter 4.

I want to thank my colleagues of the Institute of Computer Science and special thanks to my colleagues of the Fault Tolerant Computing Group who supported my work at the University of Potsdam. I want to thank Hendrik Hartje and Dorothea Müller for carefully reading the manuscript.

Special thanks to my parents, my brother and my friends who supported me all the time.

2 Structural Output Space Compaction for Combinational Circuits

In this chapter, the design of output space compactors based on a circuit analysis is described. The following section briefly introduces its usage for testing and concurrent checking and the two fault models, which are considered. Then approaches that are already known for space compaction for combinational circuits are discussed. The type of compactors considered in this chapter are linear space compactors. Therefore, the basic concept of these compactors is described. Then the two main steps for the proposed design of a structural space compactor are explained: One section introduces algorithms which analyze the structure of the circuit and another section describes the second step, i. e. a heuristic algorithm which derives the compaction function from the results of the analysis algorithm. Finally, the last two sections give experimental results and a summary.

Parts of the work described in this chapter have been published in [77, 79].

2.1 Introduction

The following short description explains how output space compaction can be applied to testing and concurrent checking. Afterwards fault models, known design approaches, and in particular linear space compactors are discussed.

2.1.1 Testing and Output Space Compaction

Figure 1.1 schematically shows a circuit under test with an output compactor. While a set of precomputed test patterns is applied to the circuit inputs, the output of the circuit is compacted by a space compactor.

For built-in self test the compacted outputs instead of the functional outputs can be connected to a time compactor. For example, the time compaction can be done by a multi-input signature analyzer which computes a signature while the test patterns are applied. After the test is completed, the final signature is compared to the corresponding signature of the fault-free circuit. The insertion of the compactor drastically reduces the size of the time compactor but may increase fault masking due to the additional stage of test response compaction.

If the test responses are propagated to a test access mechanism, the insertion of a space compactor reduces the amount of test response data. Thus, for the comparison between the obtained signature and the fault-free signature a smaller comparator can be used and the size of memory required to store the fault-free test responses is smaller. Furthermore, if the test responses should be observable in parallel at the chip output pins, then the number of required pins is less, otherwise the time required to serially shift out the test responses is reduced. On the other hand, the disadvantage is that the output space compactor may map an erroneous output to a fault-free compacted output so that the corresponding fault is masked.

Although below the design of space compactors is only described for functional outputs, these space compactors can also be applied to compress the test data of observation points. Furthermore, a single space compactor can be used for both, functional outputs and outputs of observation points. If the outputs of the observation points are conceived as functional outputs, then the design described below can be easily used to obtain an extended compactor for the test responses derived from observation points and functional outputs. In order to simplify the description of the design approach, the description of the space compactor design is restricted to the application to functional outputs.

Basically, the input patterns for testing a digital circuit can be obtained in two different ways: The test patterns can be computed by an automatic test pattern generator (ATPG) or a pseudo random pattern generator can be used.

An automatic test pattern generator processes the netlist of the circuit, which has to be tested. It computes for a given fault class a set of input patterns, which tests as many faults as possible. After that, the automatic test pattern generator tries to reduce the size of the test set, so that the fault coverage achieved by the test set is not decreased.

If output space compaction is used, the generation of a deterministic test set can be modified as follows: The automatic test pattern generator processes the netlist including the output compactor. Therefore, it takes into account the additional compaction stage during the generation of test patterns and generates also test patterns for faults which are located in the space compactor.

In some cases the fault coverage, which can be achieved by this method, is smaller than the fault coverage obtained without output compaction. Due to the additional space compactor, faults that are previously testable can become hard to detect or even untestable at the observed outputs, so that the problem of test generation is more complex than without output compaction. In practice, in many cases the runtime required for test generation increases considerably [10].

Pseudo random test means, that a large number of pseudo random vectors are applied to the circuit, so that as much faults as possible are tested by at least one input pattern. But it may happen that a testable fault remains undetected after the pseudo random test: Then only one or a few vectors or a certain sequence of vectors out of the set of all possible input vectors make the fault observable and the required vectors are not generated by the pseudo random pattern generator. On the other hand, pseudo random test inputs can test a large set of unmodeled faults. Therefore, it is important, to choose an appropriate pseudo random pattern generator, so that a moderate number of input patterns achieves a high fault coverage and tests many unmodeled faults.

If the output of the circuit under pseudo random test is compacted, then an erroneous vector at the functional outputs can be masked, that means that due to the loss of information it is mapped to the fault-free compacted test response. If the input patterns of a pseudo random test make a fault observable at the functional outputs, but fault masking of the compactor occurs for all these input patterns, then the fault is undetectable at the compacted outputs.

Deterministic testing and pseudo random testing can be combined. In this case, a deterministic test set is generated and a sequence of pseudo random test patterns is chosen, so that a large subset of the deterministic test set is part of the sequence. Then additional logic is used to modify several pseudo random test patterns in such a way that all patterns of the deterministic test set are included in the modified sequence. If a space compactor is used to reduce the amount of test response data, then the compactor should be included in the generation of the deterministic test set as explained above.

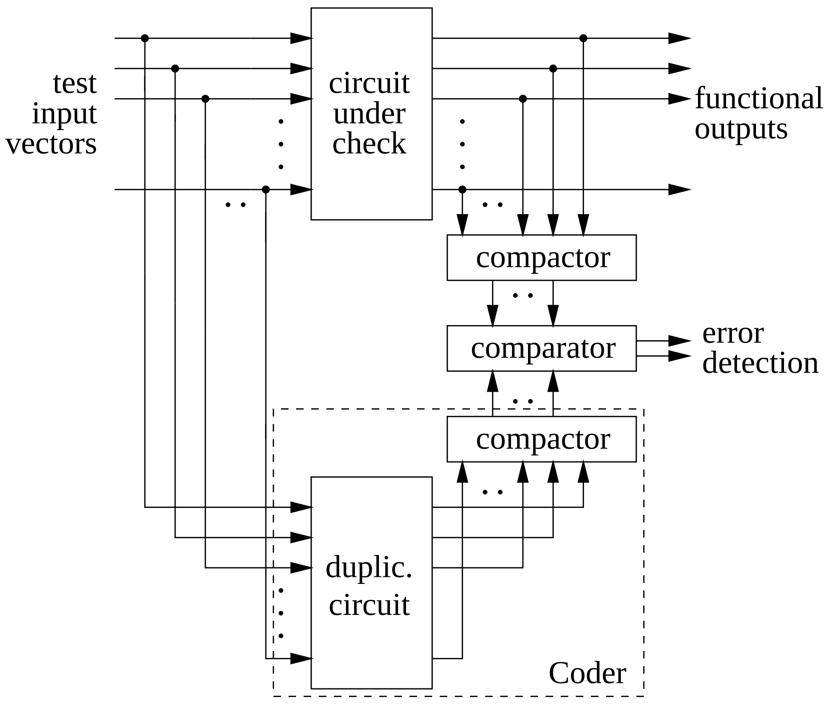


Figure 2.1: Duplication and comparison combined with output compaction

2.1.2 Concurrent Checking and Output Space Compaction

As described in the preface of this thesis, output compaction can also be used for concurrent checking. During the regular operation of the circuit, a coder is used to encode the output values of the circuit into some specific error code. In many cases, a space compactor can be added, so that the output of the compactor instead of the functional output values can be encoded, and the size of the additional logic required for concurrent checking is smaller.

Figure 2.1 shows an example of the use of space compactors on concurrent checking. Without a compactor the approach of duplication and comparison consists of the circuit under check, the coder, which equals the duplicated circuit, and the code checker comparing the output vectors of the two circuits. Figure 2.1 shows the modification: The outputs of the original circuit and of the duplicated circuit are compacted by two compactors implementing the same compaction function. Thus, the size of the code checker is smaller. The code checker can be implemented by a self-checking comparator, which compares the output vectors of the two compactors. In many cases the small comparator and the two additional compactors require less chip area than the large comparator, which is necessary for the method of duplication and comparison without space compaction. In order to further reduce the area overhead, the two parts of the coder, i. e. the duplicated circuit and the second compactor, can be jointly implemented. This is indicated by the dashed box around these parts.

Again, faults can be masked by the compactor. In such a case, a fault in the circuit under check results in erroneous values at the functional outputs but these values are mapped to fault-free compacted values. Therefore, the fault is not detected, although the functional output is erroneous. If another input vector exists, which propagates the fault to the functional outputs and the space compactor causes no fault masking for this vector, then the fault is detected with some delay. Otherwise the fault will never be detected. Since concurrent checking is often used for digital circuits in critical environments, the usage of an output space compactor for concurrent checking should depend on the question, whether the risk can be accepted, that certain faults are only observed with delay or not detected at all.

2.1.3 Fault Models

An important issue is, how physical defects located in the circuit can be modelled as logic functions, that characterize the effects of the defect on the behaviour of the circuit. The representative characterization of a certain effect of a physical defect is a fault model.

The underlying fault model is very important for testing and concurrent checking: Usually test sets are generated targeting one or more fault models, the quality of a test set is judged by the achievable fault coverage, and in general but also in this chapter compactors are designed, so that fault masking is minimized for certain fault models.

One of the oldest and most frequently used fault models is the stuck-at fault model [29]. In the stuck-at fault model it is assumed, that a faulty line of the circuit is permanently set to either the value of 0 or 1 instead of the correct value. The corresponding faults are called stuck-at-0 and stuck-at-1, respectively. A single stuck-at fault (SSF) is located at exactly one input or output of a circuit gate and the remaining gates are fault-free. Test sets, which are generated targeting at stuck-at faults, are also effective in detecting faults of other fault models (for example bridging faults [54]) or many real manufacturing defects [11]. In this chapter, the space compactors are designed so that with high probability a single stuck-at fault, which produces erroneous functional output values, is also propagated by the compactor so that fault masking is less likely.

Although the approach described in this chapter is optimized for the propagation of single stuck-at faults, below experimental results are also presented for the transition fault model. The transition fault model is based on the assumption that a faulty circuit line switches either from 0 to 1 or from 1 to 0 with a delay larger than the nominal delay. The corresponding faults are called slow-to-rise and slow-to-fall, respectively. Under the transition fault model, the extra delay caused by the fault is assumed to be so large, that the transition is prevented from reaching any circuit output at the time of observation. Therefore, the delay fault can be observed independently of whether the transition propagates through a short or long path to an output.

An advantage of the transition fault model is, that the number of faults in the circuit is linear in the number of gates. On the other hand, the assumption, that the delay fault only affects one gate in the circuit, might not be realistic, since several smaller faults together can result in a performance degradation. Also, the expectation that the delay fault is large enough to propagate from the fault site through any path might not always be true, because short paths may have a large slack.

If a space compactor is added to the circuit, then the differences between the slacks of various paths might become even larger. To avoid this, the space compactor should be designed in such a way, that the lengths of all paths through the space compactor are almost equal.

In order to detect delay faults the circuit must be clocked at the same speed as during regular operation. This is always true for concurrent checking, since the circuit function is monitored during regular operation.

Since the approach for output space compaction described below should be applicable to large circuits, it is demanded, that for an arbitrary circuit the design of a space compactor requires at most polynomial runtime in terms of the number of inputs, outputs and gates of the circuit under test.

2.1.4 State of the Art

Linear space compactors were suggested for the first time in [3]. These compactors are used very often; for example, in [32, 17] it is shown, that high percentages of single stuck-at faults of typical circuits are propagated through single parity tree compactors. Additionally, a large number of papers are published discussing, how the usage of a linear space compactor consisting of more than one parity tree can further improve the obtained fault coverage. Some of these papers are mentioned below.

An approach of output data compaction for on-line detection was proposed for the first time by Sogomonyan [81]. In [81] a zero-aliasing linear space compactor is designed based on groups of independent outputs. Two outputs of the circuit are independent, if they depend upon disjoint sets of circuit inputs. This approach can also be applied for off-line testing. In general, additional logic required for concurrent checking can also be used in test mode as proposed in [75].

This structural approach is further developed by Gössel and Sogomonyan [36]. Instead of considering independent outputs, it is shown, how groups of weakly independent outputs can be determined for a given test set. Two outputs are weakly independent with respect to a fault, if an input pattern exists so that in the presence of the fault exactly one (and not both) of the outputs is erroneous.

Further approaches for designing a zero-aliasing linear space compactor according to a given test set are based on covering procedures applied to a table, that list detection probabilities of parity trees as proposed by Tarnick [83] or that list the sensitization of pairs of outputs as introduced by Chakrabarty [19]. Furthermore, Chakrabarty describes in [16], how to model the space compaction process as a graph. The design of such a compactor is related to the graph colouring problem. All these methods result in zero-aliasing compactors, but suffer from the drawback, that the design of the space compactor depends on the chosen test set.

Chakrabarty and Hayes [18] propose another structural method, which achieves 100% fault coverage for stuck-at faults and which is applicable to large circuits: First a test set is computed using an automatic test pattern generator. Then all the functional outputs are compacted by a single parity tree. Applying the test set, the faults, which are not observable at the output of the parity tree, are determined. To achieve that these faults are also detectable under the test set, single functional outputs are successively chosen and added to the output of the parity tree until a 100% fault coverage is obtained. The additional outputs are not compacted. A second approach proposed in [18] is based on insertion of observation points in the circuit under test instead of adding single functional outputs.

Savir introduces in [71] for the first time a structural method for space compaction for BIST which is not based on a given test set: In an initial step, groups of outputs, which depend upon disjoint sets of circuit inputs are determined. The parities of these groups are the compacted outputs. If the number of these groups, i. e. the number of compacted outputs, is too large, then a simulation-based heuristic procedure is used to re-group circuit outputs until a given fault coverage is achieved or a certain time limit is reached. Thus, a successful generation of a space compactor is not guaranteed.

A method for the design of linear output space compactors using a heuristic algorithm is introduced by Böhlau [9]. The algorithm uses path lengths to determine the compaction function. Again, this method is independent of any precomputed test set. In Section 2.4.3 the experimental results of the approach based on path lengths are compared to the results obtained for the methods proposed in this chapter.

Reddy, Saluja, and Karpovsky suggest several approaches for space compactors, which are based on codes used in code theory [64, 67]. For example, in [64] a Hamming code based on the parities of groups of outputs is successfully used for space compaction. In contrast to structural methods, these design techniques do not take into account the structure of the circuit.

There are also several methods to design output space compactors which are in general non-linear. Ivanov, Tsuji, and Zorian [44] propose an approach, which uses genetic algorithms to obtain such output space compactors. Since the design of the space compactors is based on estimated probabilities of error detection at the functional outputs, this method avoids extensive fault simulation.

For a precomputed test set the design of zero-aliasing space compactors achieving an optimal compaction ratio is formulated as a graph colouring problem by Chakrabarty, Murray and Hayes [20]. A non-linear space compactor is obtained by colouring the graph and interpreting the colours as logic values.

Pouya and Touba propose a zero-aliasing space compactor design consisting of trees of AND-, OR-, NAND-, or NOR-gates in [61]. The trees of a compactor, which is designed for testing are synthesized by adding one gate at a time without introducing redundancy. Therefore, the synthesis requires checks for the introduction of redundancy, which is based on calculations using 5-valued logic and automatic test pattern generation. Thus, the runtime of the design algorithm increases more than polynomially with respect to the size of the circuit structure.

Other methods which are based on nonlinear output space compactors are hybrid space compaction [53], dynamic space compaction [47], modified dynamic space compaction [25], and the quadratic function method [48].

2.1.5 Basic Notations

In the following, a circuit C with m circuit inputs $x_1, \ldots, x_m$ and n circuit outputs $y_1, \ldots, y_n$ is considered. The space compactor SC maps the functional outputs $y_1, \ldots, y_n$ to the compacted outputs $z_1, \ldots, z_k$ yielding $k < n$. SC implements the boolean functions $z_i = f_i(y)$, with $i = 1, \ldots, k$ and $y = y_1, \ldots, y_n$.

Definition 2.1 (Compaction Ratio) *Let SC be a space compactor with n inputs and k outputs. Then $\frac{n}{k}$ is called the compaction ratio of SC.*

In the following, linear space compactors are considered. The functions implemented by such a space compactor are linear:

Definition 2.2 (Linear Boolean Function) *A boolean function $f(y_1, \ldots, y_n)$ is linear, if and only if coefficients $c_i \in \{0, 1\}$, $i = 0, \ldots, n$, exist so that f can be described by*

$$f(y_1, \ldots, y_n) = c_0 \oplus c_1 y_1 \oplus \ldots \oplus c_n y_n.$$

Definition 2.3 (Linear Space Compactor) *Let SC be a space compactor. If the boolean functions, which are used to compact the data input of SC, are linear, then SC is called a linear space compactor.*

Since the boolean functions $z_i = f_i(y), i = 1, \ldots, k$, implemented by the linear space compactor SC are linear, the compactor can be implemented using only XOR gates and no other gate type. Because the XOR operation "$\oplus$" is commutative, it is sufficient to describe each boolean function $f_i(y)$, $i = 1, \ldots, k$, by a group $Y_i = \{y_{i,1}, \ldots, y_{i,n_i}\}$ of circuit outputs. Then the parity of the outputs included in the group Y_i equals the function of output z_i of the linear space compactor SC, i. e. $f_i(y) = y_{i,1} \oplus \ldots \oplus y_{i,n_i}$.

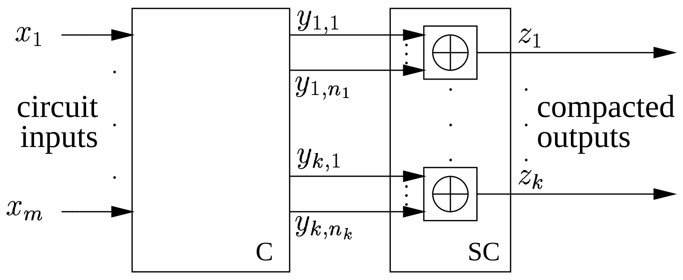


Figure 2.2: Linear output space compactor

A circuit C and a linear space compactor SC are shown by Figure 2.2. The groups of outputs $Y_i = \{y_{i,1}, \dots, y_{i,n_i}\}$, $i = 1, \dots, k$, are called parity groups and these groups are disjoint. The parity of a group equals an output of SC. It is computed by a tree consisting of XOR gates, which is not shown in detail.

In general, parity groups do not have to be disjoint. A special case of a linear output compactor is SC_1, which computes the parity of all circuit outputs. SC_1 maps all outputs of the circuit to a single output.

2.1.6 Characteristics of Linear Output Space Compactors

The usage of space compactors for testing or concurrent checking causes additional area overhead. The number of 2-input XOR gates required to form a parity tree of a linear space compactor depends on the number of circuit outputs. If n_i outputs are included in the parity group Y_i, then $n_i - 1$ XOR gates with two gate inputs are necessary to compute the parity of the outputs included in Y_i. Thus, the number of additional gates of the linear space compactor can exactly be determined. For example, if each of the n outputs of C is included in exactly one of the parity groups Y_i, $i = 1, \dots, k$, then the linear space compactor can be built using $n - k$ 2-input XOR gates.

The additional gates used for output compaction cause delays, which could reduce both, the test performance and also the performance during normal operation. If a tree of XOR gates, which corresponds to $Y_i = \{y_{i,1}, \dots, y_{i,n_i}\}$ is fully balanced, then the depth of this tree is $\lceil log_2(n_i) \rceil$.

A space compactor designed for testing and concurrent checking should not only achieve a high compaction ratio, but it should also propagate input errors and internal errors of the space compactor. If the linear space compactor consists of parity trees implemented by XOR gates, then any internal fault of the compactor which modifies the output of a single XOR gate also modifies the output of the compactor. Thus, single internal faults are easily observable, which is an important benefit of linear space compactors.

The next theorem describes which errors at the circuit outputs, i. e. at the inputs of the linear space compactor, are not detectable at the compacted outputs:

Theorem 2.4 *Let C be a circuit and let SC be a space compactor of the functional outputs $y_1, \dots, y_n$ derived from the parity groups $Y_i = \{y_{i,1}, \dots, y_{i,n_i}\}$, $i = 1, \dots, k$. An error at the functional outputs of C is not propagated to the output of the compactor, if and only if for each group Y_i the number of erroneous outputs in the group is even or zero.*

Proof:

1. If for each group Y_i the number of erroneous outputs in the group is even then the parity of each group does not change. Therefore, the outputs of the compactor are not modified and the error is not propagated.
2. If the error is not propagated, then the parity of each group is equal to the parity in the fault-free case. Therefore, for each parity group Y_i the number of erroneous outputs in the group is even or zero. □

It follows:

Corollary 2.5 *Let C be a circuit and SC be a space compactor of the functional outputs $y_1, \dots, y_n$ derived from the disjoint parity groups $Y_i = \{y_{i,1}, \dots, y_{i,n_i}\}$ with $i = 1, \dots, k$ and $Y_1 \cup \dots \cup Y_k = \{y_1, \dots, y_n\}$. Then any error at the functional outputs, which is propagated by the parity of all outputs $y_1, \dots, y_n$ (i. e. compactor SC_1), is also propagated by SC.*

Proof:
If the error is propagated by SC_1 then an odd number of circuit outputs must be erroneous. Because each circuit output is exactly included once in one of the parity groups Y_i, there must exist at least one group with an odd number of erroneous outputs. It follows that the parity of this group is modified by the error and therefore the error is propagated by SC. □

The reversion of the corollary is not true: For example, if there are exactly two parity groups with an odd number of erroneous outputs then the error is propagated through the compactor SC. If no other functional outputs are erroneous then the sum of all erroneous outputs is even. Thus, the parity of all circuit outputs is not changed by the error.

It follows from Theorem 2.4, that fault masking can be minimized, if the probability is low, that an even number of outputs of the same group are affected by a random error. This issue can be simplified by targeting to a low probability, that two outputs, which are included in the same group, are simultaneously erroneous. The simplification might cause bad results for some circuits: If there exist three outputs which are simultaneously erroneous for almost all observable SSF, then the three outputs can be assigned to the same group resulting in a small loss of propagation probability. In contrast, since for these outputs the probability that the outputs are pairwisely erroneous is very high, the outputs might not be included in the same group. The question is discussed below, whether it is likely that faults are simultaneously propagated to three or more outputs.

The following experiments were performed: The combinational circuits of the ISCAS '85 benchmarks suite [13] were simulated using the fault simulator `Minos` which is briefly described in the appendix of this work. Table 2.1 lists the ten combinational circuits including columns that show the number of gates, inputs, and outputs of each circuit.

The application of 100000 pseudo random input patterns and the propagation of each possible single stuck-at fault (SSF) were simulated. The number of erroneous outputs was counted for each input pattern and for each fault, which could be detected after applying a pseudo random input pattern. The results are presented in Table 2.2.

The very left column of Table 2.2 lists the investigated circuits. The next columns present the percentages of the combinations of applied input patterns and detectable SSF, which result in $1, 2$ or 3 simultaneously erroneous outputs. The very right column gives the percentage for more than 3 simultaneously erroneous outputs.

Combinational	Number of		
Circuit	gates	inputs	outputs
c432	160	36	7
c499	202	41	32
c880	357	60	26
c1355	514	41	32
c1908	880	33	25
c2670	1161	233	140
c3540	1667	50	22
c5315	2290	178	123
c6288	2416	32	32
c7552	3466	207	108
average	1311.3	91.1	54.7

Table 2.1: ISCAS '85 benchmark circuits

The experimental results show, that there is only one circuit (c3540) out of ten circuits with a high probability, that a SSF is simultaneously propagated to three outputs. The corresponding percentage is 33.84 %. But in general, there is a low probability that three or even more outputs are simultaneously erroneous. The average probabilities for the ten benchmark circuits are 8.48 % and 7.39 %, respectively.

In [57] the average probability of three or more simultaneously erroneous outputs is considerably lower ($<$ 3 %). The reason is, that mainly small circuits with a low number of circuit outputs have been investigated.

Because errors, which modify many outputs are less likely, this chapter is focused on minimizing the probability, that two outputs included in the same parity group are simultaneously erroneous. This leads to the question, how to derive from the netlist for each output pair (y_i, y_j), $1 \leq i \neq j \leq n$, of the circuit C a value, which estimates the reduction of fault propagation, if the outputs y_i and y_j are included in the same parity group. The next section describes several approaches, which determine such values based on an analysis of the circuit structure.

2.2 Analysis of the Circuit Structure

In the following the analysis of the circuit structure requires the knowledge of the circuit netlist. This knowledge is utilized to estimate for two given circuit outputs the loss of fault propagation, if these two outputs are included in the same parity group of the linear space compaction function.

The loss of fault coverage can exactly be determined by fault simulations or by algebraic methods. A simulation of the circuit must apply all possibles input patterns to the circuit inputs and thus the runtime increases exponential with respect to the number of circuit inputs. Therefore, an analysis based on simulations is not applicable to large circuits. The problem of exponential runtime is similar for algebraic methods. These methods might be practically successful for small and several medium sized circuits - for example, if the methods are based on Binary Decision Diagrams (BDD) [50, 51] - but fail for many large circuits, because they require a large amount of memory and exponential runtime.

In general, the problem to decide whether a single stuck-at fault becomes redundant, if two outputs

Combinational Circuit	# erroneous outputs 1	2	3	≥ 4
c432	48.89	17.56	14.58	18.97
c499	90.97	7.30	1.38	0.35
c880	91.58	6.09	1.39	0.94
c1355	94.96	4.21	0.66	0.17
c1908	77.50	11.19	3.46	7.85
c2670	69.62	18.60	5.88	5.90
c3540	41.60	10.07	33.84	14.49
c5315	70.37	15.30	7.13	7.20
c6288	53.43	23.15	11.53	11.89
c7552	77.10	11.85	4.91	6.14
average	71.60	12.53	8.48	7.39

Table 2.2: Percentages of n simultaneously erroneous outputs

are included in the same group, is equivalent to the problem of the detection of single stuck-at faults in logic circuits, which is NP-complete [41].

Therefore, the methods introduced in the following sections only estimate the reduction of fault propagation, if two functional outputs are assigned to the same parity group. The estimations of the reduction were developed in such a way, that for an arbitrary circuit the analysis algorithms always require polynomial runtime with respect to the number of inputs, outputs and gates of the circuit. The results of the analysis are values assigned to each pair (y_1, y_2) of functional outputs. The larger the value is, the higher is the estimated reduction of fault propagation, if the two outputs y_1 and y_2 are included in the same parity group. Even if each of the following methods does not compute the exact reduction of fault coverage, the results of these methods can be used to form convenient groups of outputs.

The next section presents the analysis based on common gates ratio, then the following section describes the analysis based on propagation distances, and then analysis algorithms based on approximated probabilities are discussed. The analysis algorithms which are introduced in the following are restricted to circuits whose netlists only consist of inverters, AND-, OR-, NAND-, NOR-, XOR-, and XNOR-gates. However, the algorithms can be extended, so that gates implementing other functions can also be processed.

2.2.1 Analysis Based on Common Gates Ratio

Let y_i and y_j be two outputs of the given circuit C. Let these two outputs are assigned to the same parity group of the linear space compactor designed for C. If this assignment reduces the fault propagation, then the parity of the two outputs remains unmodified under the presence of an error at the functional outputs. An error is masked by the parity of y_i and y_j, if and only if both outputs are erroneous, which means that a fault is propagated simultaneously to both outputs. Hence there must exist at least one fault which can be propagated to both outputs.

Therefore, the number of single stuck-at faults which can be propagated to y_i and y_j influences the reduction of fault propagation, if these outputs are included in the same parity group. This number

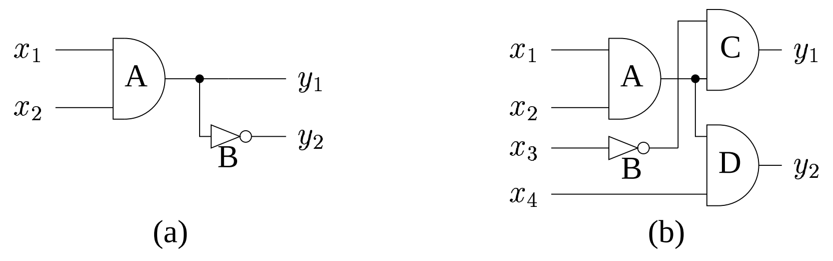


Figure 2.3: Example circuits for the common gates ratio approach

depends on the number of gates included in the intersection of the transitive fanins of the two outputs y_i and y_j. The transitive fanin $F_{in,g}$ of a gate g denotes the set of all inputs, gates, and outputs, that are part of any path from a circuit input to the gate g.

In [81] Sogomonyan introduces pairs of independent outputs, which are two outputs with an empty intersection of the corresponding transitive fanins. Approaches based on groups of independent outputs are introduced for self-checking and self-testing circuits in this paper. For many circuits the number of pairs of independent outputs is very small and thus these pairs can not be used to achieve a high compaction ratio. However, a good measurement of the loss of fault propagation should assign 0 to a pair of independent outputs, since no SSF can modify both outputs.

The common gates ratio approach meets this constraint: The ratio $r_{i,j}$ is the estimation of loss of fault propagation assigned to the pair of outputs (y_i, y_j). It is the number of gates of the intersection of the transitive fanins of y_i and y_j divided by the number of gates which are included in the union of the transitive fanins of y_i and y_j. Thus, the larger the number of SSF that are detectable at both outputs, the larger is the numerator of $r_{i,j}$. On the other hand, if (y_i, y_j) is a pair of independent outputs, which means that the transitive fanins of y_i and y_j are completely disjoint, then it holds $r_{i,j} = 0$.

The common gates ratio measurement $r_{i,j}$ depends on the size of the union of the transitive fanins as mentioned above, because it is

$$r_{i,j} = \frac{|F_{in,y_i} \cap F_{in,y_j}|}{|F_{in,y_i} \cup F_{in,y_j}|}.$$

If the faulty gate is included in the intersection $F_{in,y_i} \cap F_{in,y_j}$ and the union of the transitive fanins contains many gates, then there might be many gates included in the paths from the faulty gate to the outputs y_i and y_j. Furthermore, it is more likely, that one or more gates do not propagate the SSF and therefore the SSF is only propagated to one output: either y_i or y_j. In this case the fault is not masked by the parity.

Figure 2.3 shows two small sample circuits with two outputs. In particular, for the circuit of Figure 2.3 (a) the common gates ratio measurement can be computed as follows:

$$\begin{aligned}
F_{in,y_1} &= \{x_1, x_2, A\} \\
F_{in,y_2} &= \{x_1, x_2, A, B\} \\
F_{in,y_1} \cap F_{in,y_2} &= \{x_1, x_2, A\} \\
F_{in,y_1} \cup F_{in,y_2} &= \{x_1, x_2, A, B\}
\end{aligned}$$

and thus $r_{1,2} = \frac{3}{4} = 0.75$. Obviously, an SSF at any gate of the intersection $F_{in,y_1} \cap F_{in,y_2}$ is always masked if y_1 and y_2 are included in the same group.

Figure 2.3 (b) shows another sample circuit. $F_{in,y_1} \cap F_{in,y_2}$ is the same set as above, but the union $F_{in,y_1} \cup F_{in,y_2} = \{x_1, x_2, x_3, x_4, A, B, C, D\}$ is now larger yielding a smaller value $r_{1,2} = \frac{3}{8} = 0.375$. In contrast to the previous example, an SSF at any gate of the intersection $F_{in,y_1} \cap F_{in,y_2}$ can result in an erroneous value of the parity $y_1 \oplus y_2$: The fault is detected, if the fault is observable at the output of the AND-gate A and it is either $x_3 = 0$ or $x_4 = 1$.

These two examples show, that the larger the union of the transitive fanins of the two corresponding outputs is, the more likely is a small probability of faults propagated simultaneously to both outputs.

2.2.2 Analysis Based on Propagation Distances

The loss of fault propagation caused by the assignment of two circuit outputs y_i and y_j to the same parity group is roughly estimated, if only the number of gates in the intersection $F_{in,y_i} \cap F_{in,y_j}$ is taken into account. In the presence of a fault at one of the gates in the intersection, the fault is not masked by the compactor, if the fault is propagated to exactly one of the outputs y_i or y_j and not to both. This means that for one output no sensitized path exists from the faulty site to the output.

For example, the propagation of errors along a certain path can be prevented by a controlling value at the input of an AND-, OR-, NAND-, or NOR-gate. The value assigned to an input of a gate is a controlling value, if it uniquely determines the output value of the gate regardless of the values assigned to the other inputs. If the output value of an AND-, OR-, NAND-, or NOR-gate of the path is determined by a controlling value of an input line which is not included in the path, then the path is not sensitized. Therefore, the probability of a sensitized path depends on the number of AND-, OR-, NAND-, or NOR-gates along the path. The smallest number of these gates along paths from a gate to an output is called the propagation distance between the gate and the output.

Definition 2.6 (Propagation Distance) *Let C be a circuit consisting of the following gates: inverters, AND-, OR-, NAND-, NOR-, XOR-, and XNOR-gates. The smallest number of AND-, OR-, NAND-, or NOR-gates along a path from a gate g to an output y is called the propagation distance $d_p(g, y)$ between gate g and output y.*

The definition of the propagation distance d_p is based on the smallest number, since for propagating a fault it is sufficient that at least one sensitized path exists and that path might be the “shortest” one. This measurement utilizes the simplification, that it is sufficient to consider separately each path existing from the faulty site to the output. But if there are several paths, then these paths must reconverge since they all end at the same output line. Furthermore, the propagation of the fault through two or more paths can be mutually eliminated so that the propagation ends at the point of reconvergence. This is not taken into account by the propagation distance.

Figure 2.4 (a) shows an example for two sensitized paths eliminating each other. There are two paths from the circuit input x_2 to the circuit output y_2: Path p_a includes the AND-gate A and path p_b passes through the inverter B. For $x_1 = 1$ both paths p_a and p_b propagate the value at x_2 to the OR-gate C. In this case, the value at the output of the inverter B is the inverted value of that assigned to the output of the AND-gate A and thus always one of the input lines of OR-gate C is 1 resulting in $y_2 = 1$. Therefore, for $x_1 = 1$ no fault at x_2 is propagated to the circuit output y_2.

For $x_1 = 0$ only the path p_b is sensitized and the fault can be propagated through the gates B and C to the circuit output y_2.

For each gate of the circuit the propagation distances to the different functional outputs can be obtained by the algorithm `Distances` shown in Figure 2.5. The input of the algorithm is the netlist

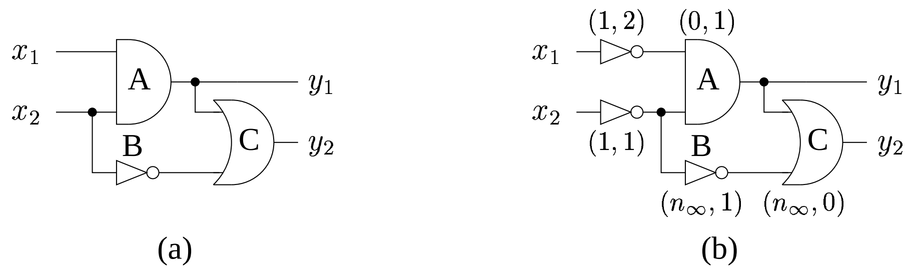


Figure 2.4: Example circuits for the propagation distance approach

of the circuit. After the initialization, the algorithm computes for each gate and for each circuit output the propagation distance starting at the circuit outputs and proceeding to the circuit inputs. Each gate g of the circuit is visited once by the algorithm, which assigns to the gate an n-tuple of propagation distances $v_g = (d_p(g, y_1), \dots, d_p(g, y_n))$.

In an initial step the algorithm `Distances` assigns an n-tuple to each gate. If the gate is directly connected to the i-th circuit output, then the n-tuple assigned to the gate is

$$(n_\infty, \dots, 0, \dots, n_\infty).$$

The i-th component of the n-tuple is 0 and all other components are n_∞, which is a symbolic value for an infinite large number. In the following it is assumed, that any value added to n_∞ yields n_∞ again.

While the algorithm `Distances` proceeds to the circuit inputs, the circuit is processed in a reversed topological order. The n-tuple of distances $v_g = (d_p(g, y_1), \dots, d_p(g, y_n))$, which is assigned to a gate g, is used to derive the n-tuples of the gates, which are connected to the inputs of gate g in the following way: First an auxiliary n-tuple v_{aux} is computed. If g is an AND-, OR-, NAND-, or NOR-gate, then v_{aux} is $(d_p(g, y_1) + 1, \dots, d_p(g, y_n) + 1)$, since g increases the distance. Otherwise it is $v_{aux} = v_g$. Then the algorithm computes from $v_{aux} = (v_{aux,1}, \dots, v_{aux,n})$ the different n-tuples assigned to every gate g' which is connected to the input of gate g. If the tuple $(n_\infty, \dots, n_\infty)$ is assigned to g', then the new tuple is $v_{g'} = v_{aux}$. Otherwise the distances of the n-tuple $v_{g'}$ are set to the minimum of the distances of the n-tuple v_{aux} and the previously computed n-tuple $v_{g'}$. v_{aux} corresponds to all paths passing through gate g, and $v_{g'}$ corresponds to paths that are not passing through gate g. It is:

$$v_{g'} = (min(v_{aux,1}, d_p(g', y_1)), \dots, min(v_{aux,n}, d_p(g', y_n))).$$

The algorithm `Distances` ends, after it has reached the circuit inputs.

Figure 2.4 (b) shows an extension of the circuit of Figure 2.4 (a). Inverters have been added at both circuit inputs to obtain a more instructive example for computing the n-tuples of propagation distances. First the n-tuples $(0, n_\infty)$ and $(n_\infty, 0)$ are assigned to the gates A and C which are connected to the circuit outputs y_1 and y_2, respectively. (n_∞, n_∞) is assigned to all other gates.

The first gate of the reversed topological order of circuit gates is the OR-gate C. The auxiliary tuple $v_{aux} = v_C + (1, \dots, 1) = (n_\infty, 1)$ is computed from $v_C = (n_\infty, 0)$. Since it is $v_B = (n_\infty, n_\infty)$, the tuple v_{aux} is assigned to the inverter B which is connected to C. The AND-gate A is also connected to the inputs of C and the final n-tuple assigned to A is $v_A = (min(n_\infty, 0), min(1, n_\infty)) = (0, 1)$. The propagation distances of the remaining gates are analogically computed and the obtained n-tuples are shown in Figure 2.4 (b).

```
1.  for each gate $g$ do
2.      assign to gate $g$ the $n$-tuple $v_g = (v_{g,1}, \dots, v_{g,n})$ with
            $v_{g,i} := \begin{cases} 0 & \text{if } g \text{ is directly connected to output } y_i; \\ n_\infty & \text{otherwise;} \end{cases}$
3.  endfor;
4.  for each gate $g$ of the circuit do
    // Loop starts at the gates which are connected to a circuit output and proceeds to
    // the circuit inputs in a reversed topological order, so that the $n$-tuple $v_g$ assigned
    // to $g$ is already correctly determined by the algorithm
5.      if gate $g$ is an AND-, NAND-, OR-, or NOR-gate then
6.          $v_{aux} := (v_{g,1} + 1, \dots, v_{g,n} + 1)$
7.      else $v_{aux} := (v_{g,1}, \dots, v_{g,n})$;
8.      for each gate $g'$ connected to the input of $g$ do
9.          $v_{g'} := (min(v_{aux,1}, v_{g',1}), \dots, min(v_{aux,n}, v_{g',n}))$;
10.     endfor;
11. endfor;
12. end.
```

Figure 2.5: The algorithm `Distances`

Obviously, the complexity of the algorithm `Distances` is linear with respect to the number of gates of the circuit. If the number of gates is $|G|$ and the number of outputs is n then the complexity is bounded by $O(|G| \cdot n)$.

The propagation distances d_p computed by the algorithm `Distances` are used to estimate the loss of fault propagation if two circuit outputs y_i and y_j are assigned to the same parity group. Thus, this estimation corresponds to the common gates ratio $r_{i,j}$ described in the previous section.

Given the pair of outputs (y_i, y_j), the estimation $d_{i,j}$ based on propagation distances is defined as follows:

$$d_{i,j} = \sum_{g \in F_{in,y_i} \cap F_{in,y_j}} 2^{-(d_p(g,y_i) + d_p(g,y_j))}$$

The definition of $d_{i,j}$ can be explained as follows: Let g be a faulty gate in the intersection of the transitive fanins of y_i and y_j and let y_i and y_j be included in the same parity group of the linear output compactor. A fault at the gate g does not change the value of $y_i \oplus y_j$, if the fault is simultaneously propagated to both outputs. Therefore, the distances $d_p(g, y_i)$ and $d_p(g, y_j)$ are added to describe this case. If it is assumed for simplification, that most gates are 2-input gates and the probability is 0.5, that a fault is propagated from the inputs of an AND-, OR-, NAND-, or NOR-gate to the output of such a gate, then the following term is derived: $2^{-(d_p(g,y_i)+d_p(g,y_j))}$. This term, which is based on propagation distances, estimates the loss of fault propagation caused by masking a fault of gate g. Finally, $d_{i,j}$ is the sum of these terms computed for all gates, which are included in the intersection of the transitive fanins of y_i and y_j.

If (y_i, y_j) is a pair of independent outputs, which means that the transitive fanins of y_i and y_j are completely disjoint, then $F_{in,y_i} \cap F_{in,y_j}$ is empty and it follows $d_{i,j} = 0$.

For example, the outputs y_1 and y_2 of the circuit shown in Figure 2.4 (b) are considered. The two inverters at the inputs and gate A form the intersection of the transitive fanins of the outputs. Therefore, it is $d_{1,2} = 2^{-(1+2)} + 2^{-(1+1)} + 2^{-(0+1)} = \frac{7}{8}$. The analysis of the circuits shown in Figures 2.3 (a) and (b) on page 17 yields $d_{1,2} = 2^{-(0+0)} = 1$ and $d_{1,2} = 2^{-(1+1)} = \frac{1}{4}$, respectively.

Since this method requires a more complex step computing the propagation distances, the runtime of the analysis may be longer compared to the approach based on common gates ratio. Nevertheless, the complexity is still linear in the number of gates of the circuit. The benefit of the usage of the propagation distance measurement is, that the analysis is more accurate and thus better results can be expected compared to the approach, which is based on the common gates ratio.

2.2.3 Analysis Based on Approximated Observation Probabilities

The analysis introduced in the following is based on the approximation of probabilities, that stimulated single stuck-at faults are propagated to different functional outputs so that they can be observed. This analysis can be used to derive output compaction functions achieving a high compaction ratio and a small loss of fault propagation.

A precondition for the analysis is that the circuit C is given as a netlist. It is supposed, that the gates are AND-, OR-, NAND-, NOR-, XOR-, XNOR-gates, and inverters. An error at a line s of C is denoted by $e(s)$ and the set of all lines of the considered circuit is S.

In order to reduce the complexity of the analysis several assumptions will be made to compute the value $p(s \rightsquigarrow y)$, which is an approximation of the observation probability $o_C(s, y)$:

Definition 2.7 (Observation Probability) *Let C be a circuit which contains the circuit line s and the circuit output y. Then the observation probability $o_C(s, y)$ is the probability that, after applying a random input out of the uniformly distributed set of input patterns, a sensitized path from s to y exists, so that an erroneous value of s is observable at the output y.*

This definition is based on the assumption that the values of the inputs x_i, $1 \leq i \leq m$, of C are equally distributed with the probabilities $p(x_i = 1) = p(x_i = 0) = \frac{1}{2}$. Under the same assumption the analysis algorithm computes the value $p(s_i \rightsquigarrow y_j)$, which estimates the probability that an error at line s_i results in an erroneous output y_j.

Approximated Propagation Probabilities of a Single Gate To illustrate how the corresponding probabilities for the existence of sensitized paths can be approximated, first a single gate g with two inputs is considered. Let v_1 and v_2 be the input values of g. Let v_2 be erroneous which means that a fault is at the second input line. In the terminology of the D-algorithm [65, 31], this fault could be a stimulated single stuck-at-0 fault D or a stimulated single stuck-at-1 fault $\bar{D}$.

If $c(g)$ and $\bar{c}(g)$ denote the controlling and the non-controlling values of g, then the erroneous input value v_2 is propagated (is not propagated) to the output of gate g if $v_1 = \bar{c}(g)$ ($v_1 = c(g)$), respectively. For the purpose of simplification it is assumed, that the value v_1 is randomly chosen with the probabilities $p(v_1 = 1) = p(v_1 = 0) = \frac{1}{2}$. Under this assumption the probability $p(g)$, that the error is observable at the output of g, equals the probability $p(v_1 = \bar{c}(g))$. This probability depends on the boolean function implemented by the gate g. It is $p(g) = \frac{1}{2}$ for two-input AND-, OR-, NAND-, and NOR-gates. Since both values 0 and 1 are non-controlling values of XOR- and XNOR-gates, it is

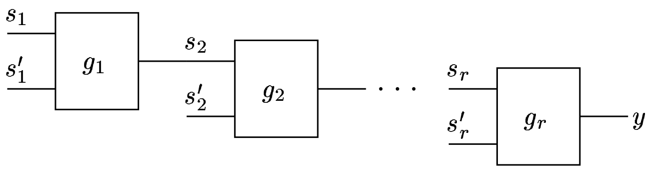


Figure 2.6: Chain of 2-input gates

$p(g) = 1$ for these gate types. Obviously, for inverters it is $p(g) = 1$, because an inverter has only one input line.

In general, to simplify the analysis, it is assumed that for any value v_i assigned to a line s_i of the circuit it is $p(v_i = 1) = p(v_i = 0) = \frac{1}{2}$. Therefore, the existence of lines whose values are much more frequent "0" than "1" or vice versa, is left out of consideration. From this simplification it is obtained, that the approximated propagation probability through a gate is $p(g) = \frac{1}{2^{k-1}}$ for any k-input AND-, OR-, NAND- or NOR-gate.

Approximated Propagation Probabilities of a Chain of Gates As shown in Figure 2.6 a chain of 2-input gates $g_1, \ldots, g_r$ is considered in the following. An error $e(s_1)$, i. e. a stimulated fault D or $\bar{D}$ at the input line s_1 of gate g_1, is propagated to the output y, if each value of the inputs $s_1', \ldots, s_r'$ is a non-controlling value of the corresponding gates $g_1, \ldots, g_r$. If the values of the lines $s_1', \ldots, s_r'$ are randomly and equally distributed, then the approximated probability $p(s_1 \rightsquigarrow y)$ that the error $e(s_1)$ is observable at the output y is

$$p(s_1 \rightsquigarrow y) = p(g_1) \cdot \ldots \cdot p(g_r).$$

Let $e(y)$ be an error at the output y and $e(s_i)$, $1 \leq i \leq r$, an error at the line s_i of Figure 2.6. Then the approximated probability $p(s_i \rightsquigarrow y)$, that the path from s_i to y is sensitized, so that an error of s_i is observable at the output, can be computed as follows:

$$\begin{aligned}
p(y \rightsquigarrow y) &= 1 \\
p(s_r \rightsquigarrow y) &= p(g_r) \\
p(s_{r-1} \rightsquigarrow y) &= p(g_{r-1}) \cdot p(s_r \rightsquigarrow y) \\
&\vdots \\
p(s_i \rightsquigarrow y) &= p(g_i) \cdot p(s_{i+1} \rightsquigarrow y) \\
&\vdots \\
p(s_2 \rightsquigarrow y) &= p(g_2) \cdot p(s_3 \rightsquigarrow y) \\
p(s_1 \rightsquigarrow y) &= p(g_1) \cdot p(s_2 \rightsquigarrow y)
\end{aligned} \tag{2.1}$$

Obviously, the computation of the approximated probabilities follows a reversed topological order of the gates.

Now the example circuit given in Figure 2.7 is considered. For the chain of gates $g_1, \ldots, g_4$ it is $p(g_1) = p(g_2) = p(g_4) = \frac{1}{2}$ and $p(g_3) = 1$. For $1 \leq i \leq 5$ the approximated probabilities $p(s_i \rightsquigarrow y)$

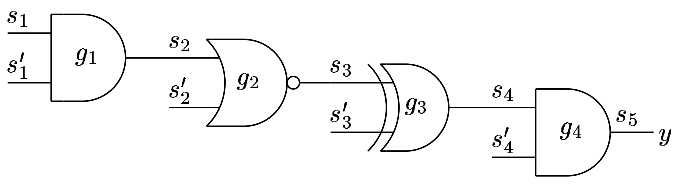


Figure 2.7: Example chain of gates

are:

$$\begin{aligned}
p(s_5 \rightsquigarrow y) &= 1, \\
p(s_4 \rightsquigarrow y) &= p(g_4) \cdot p(s_5 \rightsquigarrow y) = \frac{1}{2} \cdot 1 = \frac{1}{2}, \\
p(s_3 \rightsquigarrow y) &= p(g_3) \cdot p(s_4 \rightsquigarrow y) = 1 \cdot \frac{1}{2} = \frac{1}{2}, \\
p(s_2 \rightsquigarrow y) &= p(g_2) \cdot p(s_3 \rightsquigarrow y) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}, \\
p(s_1 \rightsquigarrow y) &= p(g_1) \cdot p(s_2 \rightsquigarrow y) = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}.
\end{aligned}$$

The approximated probabilities are determined using Equation (2.1) by proceeding gradually backwards from s_5 to s_1.

Approximated Probabilities of Observation at Circuit Outputs Now the basic algorithm is described considering a combinational circuit C with n outputs $y_1, \ldots, y_n$ and without reconvergent fanout. The analysis assigns to each internal line s_i of C the n-tuple of approximated probabilities $(p(s_i \rightsquigarrow y_1), \ldots, p(s_i \rightsquigarrow y_n))$, for which each component corresponds to one output. These probabilities are recursively computed by use of Equation (2.1) under the assumption, that the probability of error propagation through a gate g is $p(g)$.

Figure 2.8 illustrates the proposed method for the computation of the approximated probabilities $p(s_i \rightsquigarrow y_1), \ldots, p(s_i \rightsquigarrow y_n)$ for a simple combinational circuit without reconvergent fanout. The approximated probabilities along the bold path starting at the output y_2 are determined as follows: An error at the output of gate A is propagated with probability 1 to the functional output y_2 and with probability 0 to the functional outputs y_1 and y_3. Therefore, the 3-tuple (0,1,0) is assigned to this line. The probability that an error at one of the input lines of gate A is propagated to the output is approximated by $p(g) = \frac{1}{2}$. Such an error is observable at the outputs y_1 and y_3 with probability 0. This is expressed by assigning $(0, \frac{1}{2}, 0)$ to both input lines of A. The tuple $(0, \frac{1}{2}, 0)$ can easily be obtained by multiplying $(0, 1, 0)$ by $p(g) = \frac{1}{2}$.

The input line x_2 combines input lines of gate A and B. In this case, the corresponding tuples are simply added so that the tuple assigned to x_2 is $(0, \frac{1}{2}, 0) + (0, 0, \frac{1}{2}) = (0, \frac{1}{2}, \frac{1}{2})$. In a similar way the other tuples $(p(s_i \rightsquigarrow y_1), p(s_i \rightsquigarrow y_2), p(s_i \rightsquigarrow y_3))$ in Figure 2.8 are determined.

Approximated Probabilities of Reconvergent Paths The examples given above did not show the case, that often more than one path exist which can be sensitized from the faulty site to one certain output. Therefore, now a reconvergent fanout as shown in Figure 2.9 is considered. An error $e(s_i)$ at

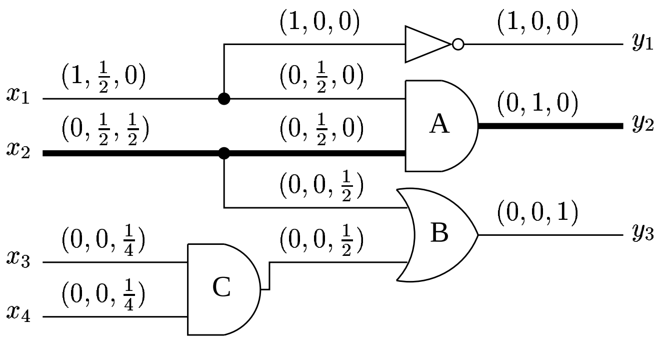


Figure 2.8: Example circuit with approximated observation probabilities

the fanout point s_i can be propagated to the point of reconvergence s_j through the two paths p_1 and p_2. p_1 and p_2 denote the paths from $s_{i,1}$ to s_1 and from $s_{i,2}$ to s_2, respectively. If the value of input s_2 of g is the non-controlling value and if the path p_1 is sensitized, then the error $e(s_i)$ can be propagated through the path p_1 from s_i to s_j. On the other hand, the error $e(s_i)$ can also be propagated through the path p_2, if p_2 is sensitized and if the non-controlling value of g is applied to s_1. Finally, the error can also simultaneously be propagated through both the paths p_1 and p_2. In this case the error will not be propagated from s_i to s_j, if g is an XOR- or an XNOR-gate, or if the number of inverters along the two paths is different modulo 2.

In many cases the probability, that the path p_1 from $s_{i,1}$ to s_1 is sensitized, and the probability, that the path p_2 from $s_{i,2}$ to s_2 is sensitized, are correlative. For example, the following two opposite cases are possible: 1. The paths are always simultaneously propagating a fault; 2. for all inputs either p_1 or p_2 or none of the two paths is sensitized. However, an exact analysis of the correlation for all reconvergent paths would extend the given bound on the runtime complexity.

If the probabilities of sensitized paths p_1 and p_2 are independent, then the probability of a simultaneous propagation equals the product of the probabilities of single sensitizations of paths p_1 and p_2. Furthermore, if the number of gates of the reconvergent fanout is not too small, then this product term is much lower than the probability, that only one of these paths is sensitized.

Because the exact computation of the correlative probabilities is as complex as the exact computation of all probabilities and in order to simplify the case of independent probabilities, the analysis approximates the observation probability for reconvergent fanouts as follows: Given the values $p(s_{i,1} \rightsquigarrow s_1)$ and $p(s_{i,2} \rightsquigarrow s_2)$, the probability that the error $e(s_i)$ is propagated from the fanout point s_i to the reconvergence point s_j, is approximated by $p(s_i \rightsquigarrow s_j)$ as

$$p(s_i \rightsquigarrow s_j) = p(g) \cdot (p(s_{i,1} \rightsquigarrow s_1) + p(s_{i,2} \rightsquigarrow s_2)).$$

If a path exists from s_j to the circuit output y, it follows:

$$\begin{aligned} p(s_i \rightsquigarrow y) &= p(s_i \rightsquigarrow s_j) \cdot p(s_j \rightsquigarrow y) \\ &= (p(s_{i,1} \rightsquigarrow s_1) + p(s_{i,2} \rightsquigarrow s_2)) \cdot p(g) \cdot p(s_j \rightsquigarrow y) \\ &= p(s_{i,1} \rightsquigarrow s_1) \cdot p(g) \cdot p(s_j \rightsquigarrow y) + p(s_{i,2} \rightsquigarrow s_2) \cdot p(g) \cdot p(s_j \rightsquigarrow y) \\ &= p(s_{i,1} \rightsquigarrow y) + p(s_{i,2} \rightsquigarrow y). \end{aligned} \tag{2.2}$$

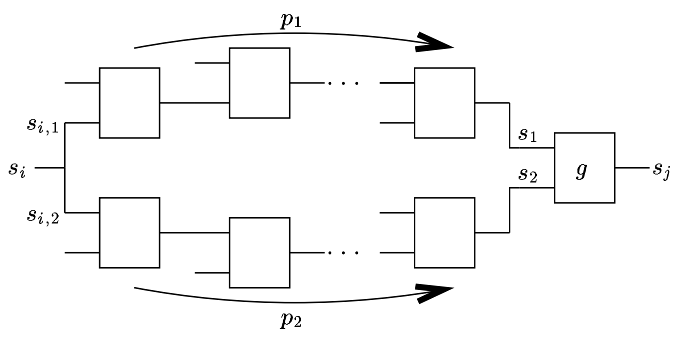


Figure 2.9: Example of reconvergent fanout

Therefore, the probabilities of propagation through reconvergent paths can be approximated as follows: During the analysis following a reversed topological order of the gates, the approximated probabilities $p(s_{i,1} \rightsquigarrow y)$ and $p(s_{i,2} \rightsquigarrow y)$ obtained by the computation are simply added at the fanout point s_i. This implies, that the probability of the case, that propagation could be prevented because both paths p_1 and p_2 are simultaneously sensitized, is neglected.

The Algorithm Now the corresponding algorithm called `Observabilities` is presented. The algorithm is used to compute approximations of the probabilities that an error of a line is propagated to different functional outputs. It assigns to each line $s_i \in S$ an n-tuple $(p(s_i \rightsquigarrow y_1), \ldots, p(s_i \rightsquigarrow y_n))$. The components of these tuples are approximated values of the probabilities $o_C(s_i, y_1), \ldots, o_C(s_i, y_n)$ that an error $e(s_i)$ at line s_i is observable at the functional outputs $y_1, \ldots, y_n$, respectively.

In an initial step the n-tuples $(1, 0, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$ of probabilities are assigned to the functional outputs. Then the algorithm `Observabilities` computes for each line s_i the corresponding n-tuple of probabilities proceeding in a reverse topological order from the circuit outputs to the circuit inputs.

The steps of `Observabilities` are described in Figure 2.10. The algorithm proceeds through the circuit only once without any backtracking. For each line it computes the corresponding n-tuple. Therefore, the complexity of the algorithm `Observabilities` computing the approximated observation probabilities $p(s \rightsquigarrow y)$ is linear with respect to the product of the number of lines $|S|$ and n, which is the number of circuit outputs of C. Since the maximal fanout of the gates is very often bounded by a constant depending on the technology of the implementation, it follows, that the runtime of the algorithm is linear in the product of the number of gates and n. The polynomial complexity of the analysis algorithm allows to apply the presented algorithm to large circuits.

Loss of Fault Propagation for Pairs of Outputs The approximated observation probabilities $p(s \rightsquigarrow y)$ computed by the algorithm `Observabilities` are used to derive a measurement $p_{i,j}$ of the loss of fault propagation caused by the assignment of two circuit outputs y_i and y_j to the same parity group of the linear output compactor. Thus, this measurement corresponds to the common gates ratio $r_{i,j}$ and the propagation distance $d_{i,j}$ introduced in the previous sections.

Given a pair of outputs (y_i, y_j), the measurement $p_{i,j}$ is the approximated probability that, after applying a random input vector, the two outputs y_i and y_j are simultaneously erroneous in the presence

```
1.  for each functional output line $y_i$, $1 \le i \le n$ do
2.      assign to line $y_i$ the $n$-tuple $(p(y_i \leadsto y_1), \dots, p(y_i \leadsto y_n))$ with
            $p(y_i \leadsto y_j) := \begin{cases} 1 \text{ for } i = j; \\ 0 \text{ otherwise;} \end{cases}$
3.  endfor;
4.  for each output line $s_g$ of a gate $g$ of the circuit do
    // Loop starts at the circuit outputs and proceeds to the circuit inputs in a reversed
    // topological order, so that either the $n$-tuple assigned to $s_g$ is already determined or
    // it is a fanout stem and the $n$-tuples assigned to the branches are already determined
5.      if $s_g$ is a fanout stem branching to the lines $s_{g,1}, \dots, s_{g,g_n}$ then
6.          assign to $s_g$ the $n$-tuple $(\sum_{i=1}^{g_n} p(s_{g,i} \leadsto y_1), \dots, \sum_{i=1}^{g_n} p(s_{g,i} \leadsto y_n))$;
7.      endif;
8.      if gate $g$ is a $k$-input AND-, NAND-, OR- or NOR-gate then
9.          $p(g) := \frac{1}{2^{k-1}}$
10.     else $p(g) := 1$;
11.     endif;
12.     assign to each input of gate $g$ the $n$-tuple $(p(g) \cdot p(s_g \leadsto y_1), \dots, p(g) \cdot p(s_g \leadsto y_n))$;
13. endfor;
14. end.
```

Figure 2.10: The algorithm `Observabilities`

of a fault. For simplification it is assumed, that for all $s \in S$ the approximated probabilities $p(s \leadsto y_i)$ and $p(s \leadsto y_j)$ are independent. $p(s \leadsto y_i)$ and $p(s \leadsto y_j)$ are the approximated probabilities that an error $e(s)$ is propagated to the circuit outputs y_i and y_j, respectively. Then the product of these two values is the approximated probability, that an error $e(s)$ is simultaneously observable at the two outputs y_i and y_j. The sum of all products related to the lines is divided by the number of lines included in the circuit, which yields the new approximation $p_{i,j}$:

$$p_{i,j} = \frac{1}{|S|} \sum_{s \in S} p(s \leadsto y_i) \cdot p(s \leadsto y_j).$$

If (y_i, y_j) is a pair of independent outputs, which means that the transitive fanins of y_i and y_j are completely disjoint, then $p(s \leadsto y_i) = 0$ or $p(s \leadsto y_j) = 0$ or both values are zero. For all three cases it follows $p_{i,j} = 0$.

For example, the values $p_{i,j}$ of the circuit shown in Figure 2.8 are computed as follows: The intersection of the lines of the transitive fanins of y_1 and y_2 only contains the single line x_1. Therefore, it is $p(s \leadsto y_1) \cdot p(s \leadsto y_2) = 0$ for all $s \in S - \{x_1\}$ yielding

$$p_{1,2} = \frac{1}{12} \cdot p(x_1 \leadsto y_1) \cdot p(x_1 \leadsto y_2) = \frac{1}{12} \cdot 1 \cdot \frac{1}{2} = \frac{1}{24}.$$

Similar considerations concerning $p_{2,3}$ yield

$$p_{2,3} = \frac{1}{12} \cdot p(x_2 \leadsto y_2) \cdot p(x_2 \leadsto y_3) = \frac{1}{12} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{48}.$$

Since the fanin cones of y_1 and y_3 are disjoint, it is $p(s \rightsquigarrow y_1) \cdot p(s \rightsquigarrow y_3) = 0$ for all $s \in S$, and furthermore

$$p_{1,3} = 0.$$

For the purpose of comparison the circuit shown in Figure 2.8 was simulated. The results of the simulation of all possible single stuck-at faults are: Eight input patterns result in simultaneously sensitized paths from x_1 to the circuit outputs y_1 and y_2. There are six input patterns simultaneously enabling fault propagation from x_2 to the circuit outputs y_2 and y_3. And there is no fault, which is simultaneously detectable at y_1 and y_3. The ratio of the results obtained from the fault simulations for the output pairs (y_1, y_2) and (y_2, y_3) is $\frac{3}{4}$. It differs slightly from $\frac{1}{2}$ which is the corresponding ratio obtained by the analysis based on approximated observation probabilities.

The complexity of the computation of the values $p_{i,j}$ exceeds the complexity of the algorithm `Observabilities`. Thus, the complexity of the complete analysis algorithm is depending on the complexity of the second step estimating the loss of fault propagation for each output pair.

$O(n^2)$ different pairs exist for the set of n circuit outputs. In the worst case, for each pair (y_i, y_j) and for each line s the product $p(s \rightsquigarrow y_i) \cdot p(s \rightsquigarrow y_j)$ must be computed in order to obtain $p_{i,j}$. It follows that the complexity of the analysis is $O(|S| \cdot n^2)$. Because the complexity is linear in the product of the circuit size and in the square of the number of outputs, the analysis based on approximated observation probabilities can be used to process efficiently large circuits.

2.2.4 Modifications of the Approximated Probabilities Analysis

The basic analysis introduced in the previous section approximates the probabilities, that a fault is propagated to different functional outputs, so that it can be observed. Because the exact computation of these probabilities would cause exponential runtime for many circuits, the analysis is simplified. These simplifications are based on reasonable assumptions implying, that only approximations and no exact probabilities are computed. This leads to the question, whether under the condition of polynomial runtime other methods can be developed for approximating the probabilities, so that more accurate results are achieved.

A large set of varying algorithms can be created in order to approximate the probabilities in polynomial runtime. Instead of investigating a confusingly large number of methods, in this work only two out of these methods are selected and described below, since no method with polynomial runtime can guarantee the best approximation of values for all circuits. The first of the two methods is a small modification of the underlying simplifications, while the second one can be conceived as an extension of the basic analysis introduced in the previous section.

2.2.4.1 Independent Propagation Probabilities of Reconvergent Paths

The difference between the analysis introduced here and the basic analysis described in the previous section is the modified approximation of propagation probabilities through reconvergent paths.

Again, the reconvergent fanout shown in Figure 2.9 is considered. As explained above, a fault at the fanout point s_i can be propagated to the point of reconvergence s_j either through the path p_1 from $s_{i,1}$ to s_1 or through p_2 from $s_{i,2}$ to s_2 or through both paths or the fault is not propagated. If g is an XOR- or an XNOR-gate or if the number of inverters along the paths is different modulo 2, then simultaneous propagation through both paths prevents the fault propagation to s_j. As mentioned above, this case is

```
5.    if $s_g$ is a fanout stem of the branches $s_{g,1}, \ldots, s_{g,g_n}$ then
6a.       assign the $n$-tuple $(0, \ldots, 0)$ to $s_g$;
6b.       for each branch $s_{g,i}$ do
6c.           for each output line $y_j$
                $p(s_g \rightsquigarrow y_j) := p(s_g \rightsquigarrow y_j) + p(s_{g,i} \rightsquigarrow y_j) - p(s_g \rightsquigarrow y_j) \cdot p(s_{g,i} \rightsquigarrow y_j)$;
6d.       endfor;
7.    endif;
```

Figure 2.11: Modified steps of the algorithm `IndependentProbabilities`

neglected if the approximated probabilities of propagation through the paths p_1 and p_2 are added and the sum of these approximations is simply assigned to the line s_i according to Equation (2.2).

The case of prevented fault propagation because both paths are sensitized is not neglected, if the probabilities of propagation through the paths p_1 and p_2 to an output y are conceived as independent probabilities. If the approximated probabilities $p(s_{i,1} \rightsquigarrow y)$ and $p(s_{i,2} \rightsquigarrow y)$ are given, then the probability that the error $e(s_i)$ at the fanout point s_i is observable at the output y is computed by the new method based on independent propagation probabilities as follows:

$$p(s_i \rightsquigarrow y) \quad = \quad p(s_{i,1} \rightsquigarrow y) + p(s_{i,2} \rightsquigarrow y) - (p(s_{i,1} \rightsquigarrow y) \cdot p(s_{i,2} \rightsquigarrow y)). \tag{2.3}$$

If two paths at a fanout point s_i do not reconverge, then the results of Equations (2.2) and (2.3) are equal since for each circuit output y at least one of the values $p(s_{i,1} \rightsquigarrow y)$ or $p(s_{i,2} \rightsquigarrow y)$ is 0. Thus, it is $p(s_{i,1} \rightsquigarrow y) \cdot p(s_{i,2} \rightsquigarrow y) = 0$ for each output y.

Only the sixth step of the basic algorithm `Observabilities` must be modified to obtain an algorithm that computes the observation probabilities assuming the independent propagation through reconvergent paths. The new steps which replace the sixth step of the basic algorithm and which lead to the new algorithm `IndependentProbabilities` are shown in Figure 2.11.

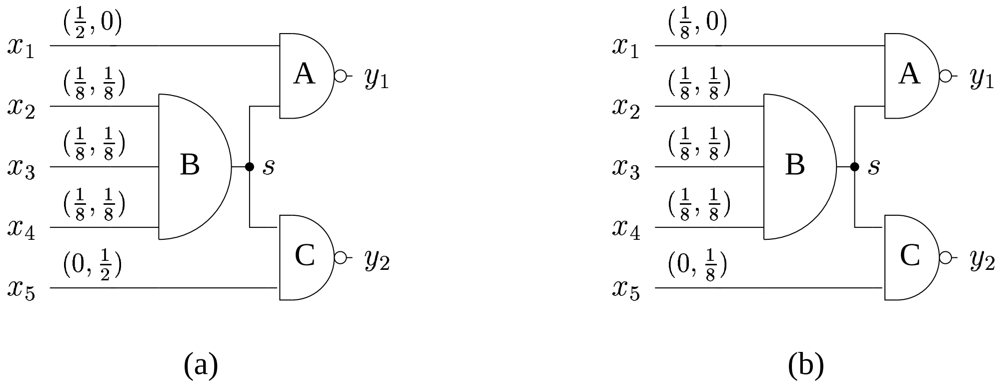
Obviously, the runtime of the algorithm `IndependentProbabilities` is equal to the runtime of the basic algorithm. It is linear in the product of the number of gates and the number of circuit outputs of C.

The approximated probabilities $p_{i,j}$, that a random input vector is applied in the presence of a fault so that the two outputs y_i and y_j are simultaneously erroneous, are computed as described in the previous section. Therefore, the complexity of the complete analysis consisting of the algorithm `IndependentProbabilities` and the computation of the values $p_{i,j}$ is linear in the product of the circuit size and in the square of the number of circuit outputs.

2.2.4.2 Analysis Including Approximated Signal Probabilities

In the previous sections, the observation probabilities have been approximated based on the assumption, that for any value v assigned to a line s of the circuit it is $p(v = 1) = p(v = 0) = \frac{1}{2}$. This means that for any line the probability, that the application of random values to the circuit inputs results in the assignment of “1” to the line, is $\frac{1}{2}$.

Instead of this assumption in the following the approximated probability $p_1(s)$ is computed in order to improve the approximation of observation probabilities. $p_1(s)$ is the approximated value of the signal



x_1
x_2
x_3
x_4
x_5
$(\frac{1}{2}, 0)$
$(\frac{1}{8}, \frac{1}{8})$
$(\frac{1}{8}, \frac{1}{8})$
$(\frac{1}{8}, \frac{1}{8})$
$(0, \frac{1}{2})$
A
B
C
s
y_1
y_2
(a)
$(\frac{1}{8}, 0)$
$(\frac{1}{8}, \frac{1}{8})$
$(\frac{1}{8}, \frac{1}{8})$
$(\frac{1}{8}, \frac{1}{8})$
$(0, \frac{1}{8})$
(b)

Figure 2.12: Example circuit for approximating propagation probabilities (a) without and (b) with considering signal probabilities

probability $sp_C(s)$ which is defined as follows:

Definition 2.8 (Signal Probability) *Let C be a circuit, which contains the circuit line s. Let the input assignments of the circuit be arbitrary, but fixed, and independently distributed for each of the circuit inputs. Then the signal probability $sp_C(s)$ denotes the probability, that applying a random input pattern results the value "1" assigned to line s.*

The theoretical complexity of the problem whether the signal probability of a given line is 0 or 1 is equivalent to the problem of Boolean Satisfiability, which implies, that it falls in the class of NP-complete problems [33]. Therefore, there is no algorithm with polynomial runtime, which computes the exact signal probabilities for all circuits.

Figure 2.12 shows a small example circuit with varying signal probabilities. The value assigned to the line s which is the output of gate B is "1" if and only if the values applied to the circuit inputs x_2, x_3, and x_4 are "1". Therefore, the signal probability $sp_C(s)$ is very low, it is $sp_C(s) = \frac{1}{8}$. In contrast, the signal probabilities of the circuit outputs are high: It is $sp_C(y_1) = sp_C(y_2) = \frac{15}{16}$.

For circuits with a large number of lines with signal probabilities, which differ significantly from the average value $\frac{1}{2}$, the basic analysis assuming the signal probability $\frac{1}{2}$ for each circuit line may generate poor values. For example, Figure 2.12 (a) shows the tuples assigned to the circuit inputs after running the algorithm `Observabilities` and part (b) of Figure 2.12 gives the exact observation probabilities of the same circuit. The tuples assigned to the inputs x_1 and x_5 given in (b) differ considerably from those given in (a).

Brglez, Pownall, and Hum give in [14] an algorithm for approximating signal probabilities $p_1(s)$ for each line s of a circuit C. This algorithm, which was introduced for testability analysis is described in the following. The computation of these values starts at the circuit inputs and proceeds to the circuit outputs in a topological order of the netlist. First $\frac{1}{2}$ is assigned to each circuit input which means, that with probability $\frac{1}{2}$ the value "1" is applied to the circuit inputs. If the set of input patterns is not uniformly distributed, but for each circuit input the probability that the input line is set to "1" can be determined independently of the other circuit inputs, then the corresponding probability is assigned to each circuit input. In order to simplify the following descriptions it is assumed, that the set of input patterns is uniformly distributed and $\frac{1}{2}$ is assigned to each circuit input.

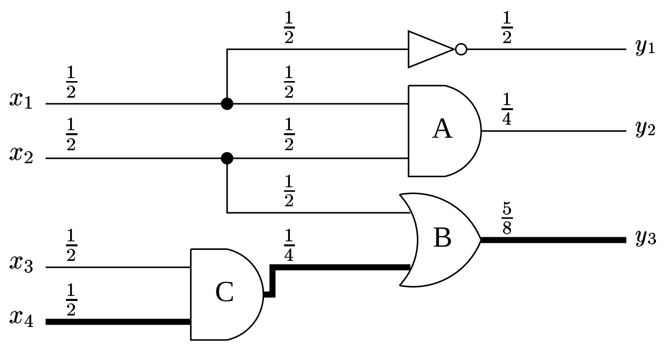


Figure 2.13: Example circuit with approximated signal probabilities

To illustrate how the algorithm determines the other values $p_1(s)$ after the initial values are assigned to the circuit inputs, a single gate g with k inputs $s_1, \ldots, s_k$ and the output s_g is considered. Since the algorithm proceeds from the circuit inputs to the circuit outputs, the approximated signal probabilities $p_1(s_1), \ldots, p_1(s_k)$ of the input lines are already determined if the value $p_1(s_g)$ assigned to the gate output is computed.

- Obviously, if the gate g is an inverter, then there is only one input line s_1. The value of the gate output is set to $p_1(s_g) = 1 - p_1(s_1)$.
- If g is an AND-gate, then the output value of the gate is “1”, if and only if “1” is assigned to all inputs of the gate. Thus, it is: $p_1(s_g) = p_1(s_1) \cdot \ldots \cdot p_1(s_k)$. Analogically, if g is an OR-gate, the output value of the gate is “0”, if and only if “0” is assigned to all inputs of the gate. This results in: $p_1(s_g) = 1 - (1 - p_1(s_1)) \cdot \ldots \cdot (1 - p_1(s_k))$.
- If g is a two-input XOR-gate, then the output value of the gate is “1”, if and only if values assigned to the input lines are not equal. Thus, the approximated signal probability $p_1(s_g)$ is determined by the sum $((1 - p_1(s_1)) \cdot p_1(s_2)) + (p_1(s_1) \cdot (1 - p_1(s_2)))$. The value $p_1(s_g)$ can be analogically derived for XOR-gates with more than two inputs.
- A NAND-gate (NOR-gate, XNOR-gate) is handled like an AND-gate (OR-gate, XOR-gate) followed by an inverter.
- Finally, the value assigned to a fanout stem is also assigned to its branches.

After the algorithm has reached the circuit outputs, the approximated signal probability of each line is determined and thus the algorithm ends. The complexity of the algorithm is linear with respect to the number of lines. Since the maximal fanout of the gates is very often bounded by a constant, which depends on the technology of the circuit implementation, the complexity of the algorithm is also linear in the number of gates.

Figure 2.13 illustrates for the proposed algorithm the computation of the approximated signal probabilities $p_1(s)$ for the small combinational circuit already shown in Figure 2.8. Now, the bold path shown in Figure 2.13 is considered starting at circuit input x_4. First of all $\frac{1}{2}$ is assigned to each circuit input. The output of the AND-gate C is “1” if both inputs are “1”. Therefore, the probability that the value of the gate output is “1” is $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. The next gate along the bold line is the OR-gate B. Since the output of the OR-gate is not “1”, if and only if both inputs of the gate are not “1”, it is $p_1(y_3) = 1 - ((1 - \frac{1}{2}) \cdot (1 - \frac{1}{4})) = \frac{5}{8}$.

The proposed algorithm computes no exact signal probabilities. If two or more paths reconverge in a gate, then the value assigned to the output of the gate might be not exact. An example is a single circuit input, which is directly connected to both inputs of a two-input NOR-gate g with output line s_g. Since firstly the value $\frac{1}{2}$ is assigned to the circuit input, $p_1(s_g) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ is obtained for s_g. However, in fact the NOR-gate is operating as an inverter and therefore it is $sp_C(s_g) = \frac{1}{2}$. Thus, the assumption, that it is $p(v = 1) = p(v = 0) = \frac{1}{2}$ for a value v assigned to any line s, results in the exact value. This example shows, that in some cases the simple assumption used to approximate the observation probabilities so far is more accurate than an additional approximation of signal probabilities.

In contrast to the example above, the computation of the values $p_1(s)$ in an initial step and the approximation of the observation probabilities including these values should lead to more accurate results for many circuits. If the approximated signal probabilities should be taken into account, then the algorithm `Observabilities` has to be modified. Only the lines 9 and 11 of the algorithm `Observabilities` given in Figure 2.10 must be replaced in the following way: For each input $s_1, \ldots, s_k$ of a k-input AND-, NAND-, OR- or NOR-gate the probability, that an error at an input line $s_i, 1 \leq i \leq k$, of the gate g is propagated to the output s_g, must be separately approximated. Then k approximated propagation probabilities $p(g_1), \ldots, p(g_k)$ instead of a single value $p(g)$ are obtained. If the outputs of the circuit are $y_1, \ldots, y_n$ then the modified algorithm assigns to the input line $s_i, 1 \leq i \leq k$ the n-tuple

$$(p(g_i) \cdot p(s_g \rightsquigarrow y_1), \ldots, p(g_i) \cdot p(s_g \rightsquigarrow y_n)).$$

The approximated probabilities $p(g_1), \ldots, p(g_k)$ can be computed in a straightforward way and depend only on the boolean function implemented by the gate. For example, given a k-input AND-gate g with inputs $s_1, \ldots, s_k$ and gate output s_g, an error at the first input s_1 of the gate can be propagated to the gate output if and only if the values of all the other gate inputs are “1”. The probability that “1” is assigned to the input line $s_i, 1 \leq i \leq k$, is approximated by $p_1(s_i)$. Therefore, the approximated probability, that an error at the first input s_1 is observable at the output of the k-input AND-gate, is: $p(g_1) = p_1(s_2) \cdot \ldots \cdot p_1(s_k)$. The approximated propagation probabilities of a NAND-, OR- or NOR-gate can analogically be determined.

The new algorithm `SignalObservabilities` consists of the approximation of the signal probabilities and the modified algorithm derived from `Observabilities` as described above. The complexity of the algorithm `SignalObservabilities` is equal to the complexity of the basic algorithms, i. e., it is linear with respect to the product of the number of gates and the number of circuit outputs of C.

Figures 2.13 and 2.14 present an example for the approximations of signal and observation probabilities of the same circuit obtained by the algorithm `SignalObservabilities`. The signal probabilities which are approximated in the first part of the algorithm are shown in Figure 2.13. The 3-tuples $(p(s \rightsquigarrow y_1), p(s \rightsquigarrow y_2), p(s \rightsquigarrow y_3))$ of all circuit lines s are given in Figure 2.14. The results can be compared to the approximation of the observation probabilities obtained by the basic algorithm `Observabilities` given in Figure 2.8. The 3-tuples assigned along the path marked by the bold line in Figure 2.14 differ from the results computed by the algorithm `Observabilities`.

Another example for the results computed by the algorithm `SignalObservabilities` is given in Figure 2.12 (b). For this circuit the approximated observation probabilities of the circuit inputs are

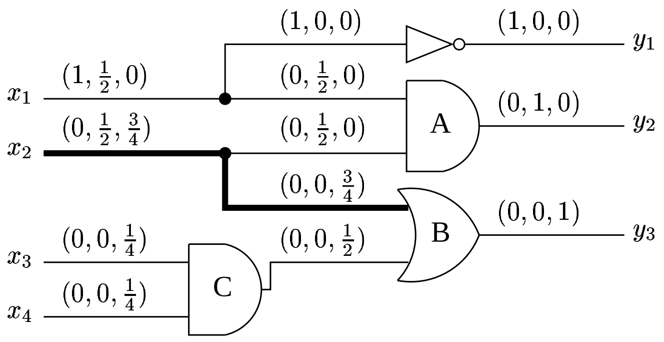


Figure 2.14: Example circuit with approximated signal and observation probabilities

equal to the exact values.

The algorithm `SignalObservabilities` can be modified so that it includes the concept of independent propagation probabilities of reconvergent paths as described in the previous section. Again, the sixth step must be replaced by the corresponding steps of the algorithm `IndependentProbabilities`. The new steps are shown in Figure 2.11.

Again, the final step of the circuit analysis is the computation of the approximated probabilities $p_{i,j}$, that under the presence of a fault a random input vector produces two simultaneously erroneous outputs y_i and y_j. This is done as described in Section 2.2.3. Therefore, the complexity of the complete analysis consisting of the algorithm `SignalObservabilities` and the computation of the values $p_{i,j}$ is the same as above, even if in practice in many cases the runtime increases compared to the runtime of the basic analysis. Nevertheless, the complexity is linear in the product of the circuit size and the square of the number of circuit outputs of the circuit.

For example, the computation of the values $p_{i,j}$ of the circuit of Figure 2.14 is as follows:

$$\begin{aligned}
p_{1,2} &= \frac{1}{12} \cdot p(x_1 \rightsquigarrow y_1) \cdot p(x_1 \rightsquigarrow y_2) = \frac{1}{12} \cdot 1 \cdot \frac{1}{2} = \frac{1}{24};\\
p_{2,3} &= \frac{1}{12} \cdot p(x_2 \rightsquigarrow y_2) \cdot p(x_2 \rightsquigarrow y_3) = \frac{1}{12} \cdot \frac{1}{2} \cdot \frac{3}{4} = \frac{1}{32};\\
p_{1,3} &= 0.
\end{aligned}$$

The value $p_{2,3}$ differs from the value obtained by the analysis of the same circuit using the algorithm `Observabilities` (see Equation 2.3): It is $p_{2,3} = \frac{1}{32}$ instead of $\frac{1}{48}$.

As mentioned above, fault simulations of the circuit given in Figure 2.14 show that eight input patterns result in simultaneously sensitized paths from x_1 to the circuit outputs y_1 and y_2. Six input patterns simultaneously enable fault propagation from x_2 to the circuit outputs y_2 and y_3. Thus, for the fault simulations the ratio of the result of the output pair (y_1, y_2) to the result of (y_2, y_3) is $\frac{3}{4}$. It is equal to the corresponding ratio of the results computed by the analysis based on approximated signal and observation probabilities. Hence for the circuit given in Figure 2.14 it is shown, that the analysis based on approximated signal and observation probabilities is more accurate than the basic analysis, which does not take into account varying signal probabilities.

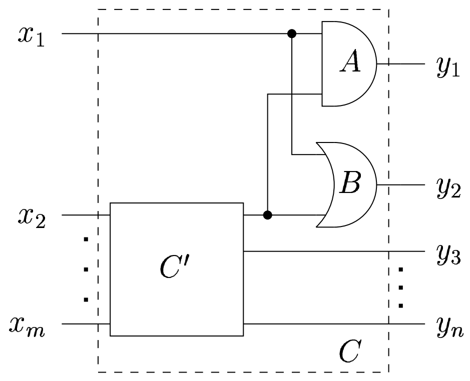


Figure 2.15: Example circuit for the estimation of the analysis error

2.2.5 Estimation of the Error of the Analysis Results

The analysis methods introduced above estimate the reduction of fault propagation if two outputs of the given circuit are assigned to the same parity group of the linear compactor. Based on comparisons of the results for different pairs of circuit outputs, the parity groups of a linear compactor are determined, so that the resulting compactor causes only a small reduction of the fault coverage. Since these values do not estimate the absolute number for any probability or the absolute percentage of any fault coverage reduction, the error of the estimations of the analysis methods is hard to determine.

Therefore, now the difference between exact and estimated values assigned to single pairs of circuit outputs is considered. Let C be a circuit with circuit inputs $x_1, \dots, x_m$, $m \geq 2$, and circuit outputs $y_1, \dots, y_n$, $n \geq 2$, and with the following characteristic: There is no input vector, which causes a simultaneous propagation of any single stuck-at fault to the two outputs y_1 and y_2. For such a circuit the reduction of fault propagation is 0, if the two circuit outputs y_1 and y_2 are assigned to the same parity group. The estimated value assigned to the output pair (y_1, y_2) by any analysis algorithm should be as low as possible, so that the value is considerably lower than the estimations assigned to other pairs of circuit outputs, which in fact can be simultaneously erroneous.

In contrast, circuits can be designed in such a way, that the estimated values of the analysis algorithms introduced in the previous sections are very high while the exact value is 0. An example circuit is shown in Figure 2.15. It shows in detail only this part of the circuit, which is important for the further considerations. The remaining part of the circuit is given as C'. Obviously, no single stuck-at fault located in C' can be simultaneously propagated to y_1 and y_2: For $x_1 = 0$ the AND-gate A does not propagate the error to y_1; for $x_1 = 1$ the OR-gate B prevents error propagation to y_2. In a similar way it can be shown, that no input vector exists, so that a single stuck-at fault at x_1 is simultaneously observable at both outputs y_1 and y_2.

If the method based on the common gates ratio is used to analyze the structure of this circuit and if $|C'|$ is the number of gates included in C', then the value $r_{1,2}$ assigned to the pair of circuit outputs (y_1, y_2) is:

$$r_{1,2} = \frac{|F_{in,y_1} \cap F_{in,y_2}|}{|F_{in,y_1} \cup F_{in,y_2}|} = \frac{|C'|}{|C'| + 2} = 1 - \frac{2}{|C'| + 2}.$$

It follows, that if the size of the circuit part C' is increased, then $r_{1,2}$ converges at the upper bound 1, although there exists no input vector resulting in simultaneously erroneous outputs y_1 and y_2.

The values, which are computed by the analysis based on propagation distances, can not be determined if the circuit part C' is not specified more exactly. Therefore, now a single gate g' of C' is considered. g' is obviously included in the intersection of the transitive fanins of y_1 and y_2. The propagation distances from gate g' to the outputs y_1 and y_2 are $d_p(g', y_1)$ and $d_p(g', y_2)$. The term $2^{-(d_p(g',y_1)+d_p(g',y_2))}$ is part of $d_{1,2}$ which is the estimated loss of fault propagation if y_1 and y_2 are assigned to the same group, since $d_{1,2}$ is defined as

$$d_{1,2} = \sum_{g \in F_{in,y_1} \cap F_{in,y_2}} 2^{-(d_p(g,y_1)+d_p(g,y_2))}.$$

This implies that the sum $d_{1,2}$ can be increased by adding gates to the circuit part C'. Furthermore, no upper bound on $d_{1,2}$ can be determined.

For the analysis algorithms based on approximated observation probabilities it can analogically be shown, that there is no upper bound on $p_{1,2}$, which is the value assigned by these algorithms to the outputs y_1 and y_2.

In summary, for each analysis method the circuit C shown in Figure 2.15 can be constructed in such a way, that for the assignment of y_1 and y_2 to the same group of the linear compactor any error of the estimated reduction of fault propagation can be produced. Although any error can be generated, the analysis results can help to determine the parity groups in such a way that for the resulting linear compactor the probability of fault masking is reduced. This is true, because the type of error described above only prevents two outputs from being assigned to the same group, although this assignment would cause no loss of fault propagation.

Errors of the analysis algorithms, which underestimate the loss of fault propagation if two circuit outputs are assigned to the same parity group are much more critical. The construction of sample circuits for such errors is difficult and usually the errors are small compared to the other type of errors given in the example above. For many circuits it is not clear, whether the values of several output pairs are underestimated or whether the values of other output pairs are overestimated.

In conclusion this section shows, that the quality of the analysis methods can not be judged from the error of the estimations of the methods. Therefore, in the following sections several benchmark circuits are analyzed by the proposed analysis methods and the obtained results are compared.

2.2.6 Comparisons Between Exact Probabilities and Analysis Results

The analysis algorithms of circuit structures introduced above were developed to estimate the loss of fault propagation, which is caused by the assignment of two outputs of the processed circuit to the same parity group of the linear output compactor. In the following, the results of these algorithms are compared with exact values obtained by fault simulation.

For a given circuit the exact values are computed in the following way: A fault simulator applies to the inputs of the circuit all possible input vectors. Therefore, if the number of circuit inputs is m, then 2^m different vectors are applied.

For each pair of circuit outputs (y_i, y_j) the number of single stuck-at faults, which are simultaneously propagated to both outputs, is determined. During the simulation with the 2^m different input vectors, for each output pair these numbers are added up, so that at the end of the simulation the sum $s_{i,j}$ is obtained.

The higher the sum $s_{i,j}$ is, the more likely is it, that a single stuck-at fault is simultaneously observable at the corresponding pair of outputs (y_i, y_j). In particular, $s_{i,j}$ divided by $2^m \cdot |F|$, which is the product of the number of applied input vectors and the number of possible single stuck-at faults of the circuit, yields the probability, that after applying a random input vector the presence of a random single stuck-at fault causes simultaneously erroneous outputs y_i and y_j. This is only true, if the set of input vectors and the set of single stuck-at faults are uniformly distributed.

In the following, $s_{i,j}$ denotes the sum, which is obtained from the fault simulation and which is assigned to the pair (y_i, y_j) of circuit outputs. $r_{i,j}$, $d_{i,j}$, and $p_{i,j}$ denote the values assigned to (y_i, y_j) by the analysis algorithms based on the common gates ratio, propagation distances, and approximated observation probabilities, respectively.

In order to compare the results of the different algorithms the values are normalized. For example, if the values $s_{i,j}, 1 \leq i < j \leq n$, are obtained by the fault simulation, then the normalized values $s'_{i,j}, 1 \leq i < j \leq n$, are

$$s'_{i,j} = \frac{s_{i,j}}{\sum_{1 \leq i < j \leq n} s_{i,j}}.$$

This implies:

$$\sum_{1 \leq i < j \leq n} s'_{i,j} = 1.$$

In the same way the normalized values $r'_{i,j}$, $d'_{i,j}$, and $p'_{i,j}$ are computed.

The normalized values $s'_{i,j}$ obtained by the simulation are compared with the values obtained by the different analysis algorithms of circuit structures. The normalized values $r'_{i,j}$ of the analysis of the common gates ratio, for example, are compared as follows: The standard deviation D_r of the values $r'_{i,j}$ from the values $s'_{i,j}$ is:

$$D_r = \sqrt{\sum_{1 \leq i < j \leq n} (s'_{i,j} - r'_{i,j})^2}.$$

$\Delta_{r,max}$ denotes the maximal difference $|s'_{i,j} - r'_{i,j}|$ of all pairs of circuit outputs. Therefore, it is $\Delta_{r,max} = max\{|s'_{i,j} - r'_{i,j}|; 1 \leq i < j \leq n\}$.

The standard deviations D_d of the values $d'_{i,j}$ and D_p of the values $d'_{i,j}$ are analogically computed based on the values $s'_{i,j}$. $\Delta_{d,max}$ ($\Delta_{p,max}$) denotes the maximal difference $|s'_{i,j} - d'_{i,j}|$ ($|s'_{i,j} - p'_{i,j}|$) of all pairs of circuit outputs, respectively.

Combinational benchmark circuits of the 1991 International Workshop on Logic Synthesis were simulated to obtain the standard deviations D_r, D_d, D_p, and the maximal differences $\Delta_{r,max}$, $\Delta_{d,max}$, $\Delta_{p,max}$ for each circuit. Since several benchmark circuits contain complex gates, these circuits were mapped using the software tool `SIS` [72]. The underlying library contained only inverters, AND-, OR-, NAND-, NOR-, XOR-, and XNOR-gates. Table 2.3 lists the 22 combinational benchmark circuits, which were investigated. The columns of the table show the names of the circuits and the numbers of gates, inputs and outputs of each circuit. If the original circuit contained complex gates, then the number of gates after the mapping of the circuit is given.

The next section gives the results of the different analysis algorithms based on the common gates ratio, propagation distances, and approximated observation probabilities. Then a section follows which presents a comparison of the results of the different analysis algorithms, which use approximated probabilities.

Combinational	Number of		
Circuit	gates	inputs	outputs
alu2	334	10	6
alu4	686	14	8
b1	11	3	11
cc	51	21	20
cm138	19	6	8
cm162	39	14	5
cm163	41	16	5
cm42	24	4	10
cm82	22	5	3
cm85	31	11	3
cmb	41	16	4
cu	48	14	11
decod	22	5	16
f51m	127	8	8
i1	36	25	16
pcle	64	19	9
pm1	39	16	13
sct	93	19	15
tcon	49	17	16
vda	584	17	39
x2	42	10	7
z4ml	59	7	4
average	111.9	12.6	10.8

Table 2.3: Investigated combinational benchmark circuits

2.2.6.1 Comparison of The Basic Analysis Algorithms

Each circuit listed in Table 2.3 was investigated using an exhaustive fault simulation in order to obtain the standard deviations D_r, D_d, D_p, and the maximal differences $\Delta_{r,max}$, $\Delta_{d,max}$, $\Delta_{p,max}$ with respect to the exact values. The results are shown in Table 2.4. For each circuit the minimal standard deviation obtained for the three analysis algorithms and the smallest difference of $\Delta_{r,max}$, $\Delta_{d,max}$, and $\Delta_{p,max}$ are marked bold.

For most circuits, that means for 13 out of 22 circuits, the standard deviation of the analysis based on approximated observation probabilities is lower than the standard deviation of the two other analysis algorithms. However, for most circuits the worst error of the values assigned to pairs of circuit outputs is minimal, if the analysis of common gates ratio is used. The lowest average standard deviation of all circuits is also obtained for this analysis method.

In summary, the results indicate that for most circuits the analysis based on approximated observation probabilities computes the most accurate values compared to the values computed by the analysis of common gates ratio and the analysis based on propagation distances. However, for a considerable number of circuits the latter algorithms may achieve better results.

Combinational	Standard deviations			max. differences		
Circuit	D_r	D_d	D_p	$\Delta_{r,max}$	$\Delta_{d,max}$	$\Delta_{p,max}$
alu2	**0.155**	0.377	0.371	**0.102**	0.199	0.195
alu4	**0.112**	0.185	0.325	**0.047**	0.111	0.207
b1	0.226	0.209	**0.194**	0.189	0.152	**0.133**
cc	0.090	0.094	**0.088**	**0.057**	0.059	0.066
cm138	0.214	0.214	**0.210**	**0.051**	0.059	0.058
cm162	0.116	0.097	**0.092**	0.048	0.044	**0.039**
cm163	0.064	0.036	**0.027**	0.042	0.025	**0.020**
cm42	0.286	**0.286**	0.287	**0.130**	0.133	0.135
cm82	0.081	0.012	**0.060**	0.058	0.098	**0.049**
cm85	**0.127**	0.168	0.308	**0.104**	0.137	0.251
cmb	0.599	0.692	**0.381**	0.545	0.625	**0.332**
cu	0.376	0.402	**0.276**	0.313	0.338	**0.253**
decod	0.140	0.143	**0.140**	**0.024**	0.025	0.026
f51m	**0.110**	0.312	0.249	**0.048**	0.155	0.128
i1	0.204	0.170	**0.141**	0.096	0.102	**0.072**
pcle	0.107	0.131	**0.062**	0.044	0.057	**0.026**
pm1	0.138	**0.137**	0.197	0.070	**0.064**	0.088
sct	0.063	**0.048**	0.108	**0.026**	0.028	0.045
tcon	0.014	0.057	**0.008**	0.005	0.018	**0.003**
vda	**0.027**	0.042	0.071	**0.004**	0.006	0.020
x2	0.167	**0.140**	0.204	0.078	**0.060**	0.100
z4ml	0.128	0.253	**0.115**	**0.074**	0.202	0.096
average	**0.161**	0.191	0.178	**0.098**	0.123	0.106

Table 2.4: Comparison of the results of different analysis methods

2.2.6.2 Comparison of Analysis Algorithms Based on Probabilities

The comparison of the previous section included the basic analysis algorithm based on approximated observation probabilities. This algorithm uses simplifications, which are based on reasonable assumptions. Above two modifications of this algorithm are proposed which might result in more accurate results: An analysis, which is based on the assumption of independent propagation probabilities of reconvergent paths, and a modification, which includes approximated signal probabilities to analyze the circuit structure. The results of these two methods and of the basic algorithm are compared in the following.

The standard deviations of the values, which estimate the loss of fault propagation caused by the assignment of two outputs to the same parity group of the linear output compactor, are computed as described above and then compared. To distinguish these values, D_{basic}, D_{indep}, and D_{signal} denote the standard deviations of the results of the basic algorithm, of the algorithm assuming independent propagation probabilities of reconvergent paths, and of the algorithm including approximated signal probabilities, respectively. The corresponding maximal differences for all pairs of circuit outputs between the exact values obtained from the simulation and the results computed by the analysis algorithms

Combinational Circuit	Standard deviations			max. differences		
	D_{basic}	D_{indep}	D_{signal}	$\Delta_{basic,max}$	$\Delta_{indep,max}$	$\Delta_{signal,max}$
alu2	0.371	0.385	**0.191**	0.195	0.200	**0.103**
alu4	0.326	0.337	**0.092**	0.207	0.216	**0.059**
b1	0.194	0.203	**0.082**	0.133	0.150	**0.065**
cc	0.088	0.088	**0.029**	0.067	0.067	**0.024**
cm138	**0.210**	**0.210**	0.214	**0.058**	**0.058**	0.63
cm162	0.092	0.092	**0.014**	0.039	0.039	**0.008**
cm163	0.027	0.027	**0.027**	**0.020**	0.020	0.022
cm42	**0.287**	0.287	0.290	0.135	0.135	**0.133**
cm82	0.060	0.070	**0.052**	0.049	0.057	**0.042**
cm85	0.308	0.309	**0.006**	0.251	0.252	**0.005**
cmb	0.381	0.380	**0.025**	0.332	0.331	**0.018**
cu	0.276	0.276	**0.200**	0.253	0.252	**0.161**
decod	0.140	0.140	**0.139**	**0.026**	**0.026**	0.027
f51m	0.249	0.258	**0.081**	0.128	0.136	**0.035**
i1	0.141	0.143	**0.069**	0.072	0.072	**0.046**
pcle	0.062	0.064	**0.035**	0.026	0.027	**0.012**
pm1	0.197	0.198	**0.150**	**0.088**	0.089	0.094
sct	0.108	0.108	**0.057**	0.045	0.045	**0.027**
tcon	0.008	**0.002**	0.024	0.002	**0.001**	0.008
vda	0.071	0.071	**0.009**	0.020	0.020	**0.002**
x2	0.204	0.206	**0.121**	0.100	0.100	**0.056**
z4ml	0.115	0.114	**0.067**	0.096	0.095	**0.059**
average	0.178	0.180	**0.090**	0.106	0.109	**0.049**

Table 2.5: Comparison of the results of different analysis methods

are $\Delta_{basic,max}$, $\Delta_{indep,max}$, and $\Delta_{signal,max}$, respectively.

Table 2.5 lists the results obtained from 22 combinational benchmark circuits of the 1991 International Workshop on Logic Synthesis. For each circuit the minimal standard deviation and the smallest value of the maximal differences of the three analysis algorithms are marked bold.

For almost all circuits, which means for 19 out of 22 circuits, the standard deviation of the analysis including approximated signal probabilities is the lowest of the standard deviations of the three investigated analysis algorithms. And even for two of the remaining three circuits, i. e. cm138 and cm42, D_{signal} is close to the minimal standard deviation. Furthermore, the average value of D_{signal} is nearly half of the average standard deviations obtained for the other two analysis methods. And for most circuits, which means for 17 out of 22 circuits, the worst error of the values assigned to a pair of circuit outputs is minimal, if the values are computed by the analysis based on approximated signal and observation probabilities.

The results of the basic analysis and the analysis assuming independent propagation probabilities of reconvergent paths differ only slightly. If the resulting standard deviations are different, then in most cases the standard derivations of the basic algorithm based on approximated observation probabilities is smaller. Therefore, the analysis assuming independent propagation probabilities of reconvergent paths is not considered in the following sections.

A comparison between Table 2.4 and Table 2.5 shows that for 16 out of 22 circuits the standard deviation of the analysis based on approximated signal and observation probabilities is the lowest of all results, which were obtained by the five investigated analysis algorithms.

2.2.7 Comparison with Testability Measurements

The analysis algorithms, which are introduced above for the approximation of observation probabilities are based on concepts, which are similar to these known from testability measurements. Testability measurements are used to estimate whether faults of a given circuit can easily be tested or not. This is done without generating a test set or running a fault simulator with random input patterns.

If a test set for a given circuit should be generated then there are testability measures that can give a relative comparison of the costs for pattern generation based on fault simulation and of the costs required for deterministic test generation. For both, deterministic and random pattern testing, such measures can help to identify circuit areas where design for testability is required to improve the testability. Testability measurements are only useful if the required effort is less than actual test set generation or fault simulation. This implies that in general such measurements can only produce approximate results.

Many testability measurements are based on the determination of the controllability, which is a measure, how difficult it is to set a value of a circuit line to a certain value, and the observability, which describes the probability, that a fault can be propagated from the faulty line to a circuit output [14, 35, 45, 60, 63, 74, 82, 84].

The main difference between any testability measurement and the analysis based on approximated observation probabilities is, that the former estimates how difficult a fault can be excited and propagated to at least one circuit output, but the latter approximates the probability, that an already excited fault can be propagated to separately considered circuit outputs. Therefore, the analysis introduced above uses tuples instead of a single value to describe the approximated observation probabilities for each circuit line. Each component of the tuples is related to one certain circuit output.

The approximated observation probabilities can easily be used to derive a testability measurement. A simple way to obtain a measurement of the observability is to add up the components of the tuple. Other more complex methods, which may result in more accurate estimations can be developed. But so far the capability of the approximated observation probabilities to be useful as a base of a testability measurement has not been investigated.

Since there is a large number of testability measurements proposing different methods to produce an accurate approximation of the fault observabilities, such methods can be adapted to the analysis described above in order to improve the results. In the next section three simple concepts are discussed, which require no adaption, because the analysis is extended in a straightforward way.

2.2.8 Extensions for the Analysis Based on Approximated Probabilities

This section presents three extensions of the approximated probabilities analysis. The goal of the first extension is to improve the accuracy of the approximated observation probabilities, the second extension introduces a more complex algorithm to approximate signal probabilities, and the third one shows, that a few modifications of the analysis algorithm are sufficient to obtain approximated probabilities of fault propagation, that take into account the direction of the resulting error at the circuit outputs.

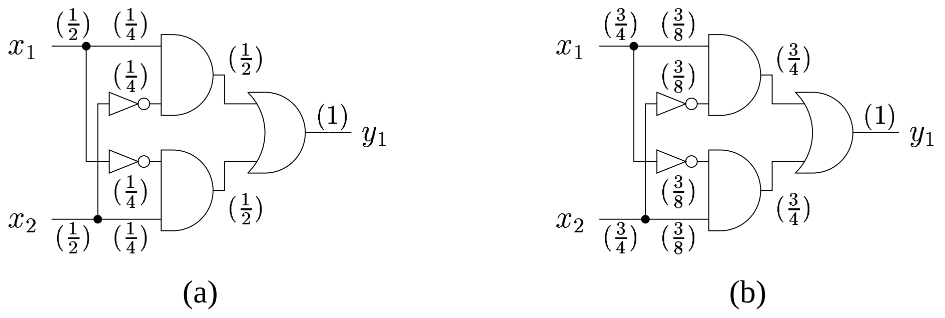


Figure 2.16: Approximated observation probabilities of the XOR-function

2.2.8.1 Complex Gates Replacing Small Circuit Parts Containing Reconvergent Paths

In most cases, reconvergent paths are one of the reasons why the observation probabilities approximated by the analysis differ from the exact values. As described in the section, which introduces the analysis algorithms based on approximated probabilities, the differences can occur if the propagation of erroneous values is prevented at the point of reconvergence. Or the approximation is not accurate, because often only a single path is sensitized so that the observation probability is larger than the approximated value.

For example, if a circuit consists of a single two-input XOR-gate g, then the analysis based on approximated observation probabilities assigns the value 1 to both circuit inputs since the propagation probability through a XOR-gate is $p(g) = 1$. This value is exact, because any erroneous value at one of the inputs is propagated to the output of the gate.

In Figure 2.16 the XOR-gate is replaced by a small circuit consisting of two inverters, two AND-gates and one OR-gate. Part (a) of Figure 2.16 shows the values which are obtained by the algorithm `Observabilities`. The values assigned to the input lines are not equal to the exact probability 1 in spite of the fact, that the function implemented by the circuit is logically the same as the function of the XOR-gate.

The values obtained by the algorithm `SignalObservabilities` are given in Figure 2.16 (b). Again, the values assigned to the input lines differ from the exact values. The reason is, that for both inputs reconverging paths exist from the inputs to the output y_1.

The following procedure reduces the effect of reconvergent paths: Before starting the analysis, disjoint parts of the circuits containing reconvergent paths are identified in such a way, that the size of each circuit part is below a certain upper bound. These parts are replaced by complex gates implementing the corresponding logical function. For each complex gate g and for each input line s_i the exact probability $p(g_i)$ of propagation from the input line s_i to the output of the gate g is computed. This can be done, for example, using fault simulation or Binary Decision Diagrams (BDD) [15]. Then for the modified circuit the approximation of the observation probabilities follows utilizing the computed propagation probabilities of the complex gates.

In the next step the complex gates are replaced by the corresponding original parts of the circuit. So far there are no approximated probabilities assigned to the internal lines of the replaced circuit parts. In a second run of the analysis algorithm only the missing values are computed based on the already assigned approximated observation probabilities. The missing values are determined without modifying any already existing approximation of probabilities.

The complexity of the procedure increases with the upper bound on the size of the replaced circuit parts. If the upper bound is raised, then it can be expected that the approximated probabilities are more accurate but that at the same time the runtime increases. The measurement of the size of the circuit parts can be defined in several ways; for example, it can be related to the number of inputs, gates or lines included in the part of the circuit.

If an upper bound on the size of replaced circuit parts is given, then the determination of the disjoint circuit parts, which are replaced by complex gates is ambiguous in many cases. Therefore, a convenient choice of replaced circuit parts may result in a more accurate approximation of observation probabilities or in a decreased runtime of the procedure.

The approximated probabilities, that in the presence of a fault a random input vector produces two simultaneously erroneous outputs, are computed as described for the basic algorithm in the section above.

2.2.8.2 Improved Approximation of Signal Probabilities

Krishnamurthy and Tollis introduce in [52] an improved algorithm for approximating the signal probabilities of a circuit. It takes into account first-order effects of reconvergent paths depending on the different circuit inputs. The complexity of the algorithm is linear in the product of the number of lines and the number of inputs of the circuit.

The basic idea of this method is, that for each circuit line in the first step m approximated signal probabilities are computed corresponding to the circuit inputs $x_1, \ldots, x_m$ of the circuit. An algorithm with polynomial runtime is used to determine in parallel for each circuit input x_i the approximated signal probability, so that no error arises from multiple dependencies on the corresponding input. However, a signal probability determined for a certain input can still be erroneous because of multiple dependencies on the other circuit inputs. Finally, for each line a weighted averaging algorithm is used to derive from m values one single approximated signal probability which is assigned to the line. In [52] the algorithm is described more in detail and experimental results are given.

Experimental results show that this algorithm achieves better approximations of the signal probabilities compared to the results obtained from the straightforward algorithm, which is part of the algorithm `SignalObservabilities` described above and which was proposed by Brglez, Pownall, and Hum [14]. In Figure 2.16, for example, a circuit is given, which implements the XOR-function: The algorithm of `SignalObservabilities` assigns to the output of the circuit the approximated signal probability $\frac{7}{16}$, but the algorithm considering first-order effects of reconvergent paths computes the exact signal probability $\frac{1}{2}$.

In general, if the more complex algorithm proposed in [52] is used for the approximation of signal probabilities of `SignalObservabilities`, then also the approximated observation probabilities may be more accurate.

2.2.8.3 Direction-Dependent Fault Propagation

The algorithms introduced above, `Observabilities` and `SignalObservabilities`, were developed for analyzing the structure of a circuit in order to derive a linear space compactor with minimized fault masking. But the analysis results might not be useful for other design approaches, which are based on a different concept.

For example, several approaches for concurrent checking [8, 27, 46, 56, 68, 69] are based on Berger codes [4]. Berger codes detect all unidirectional errors, but not all non-unidirectional errors. If the number of non-unidirectional errors is even, then these errors are masked. Errors are unidirectional, if all errors change the good values in the same way, that means all erroneous lines have the same value. Therefore, for the encoding of the circuit output based on Berger code it must be taken into account, whether one or more bits can be erroneous as well as which values are assigned to the erroneous circuit outputs. To handle this issue, a few modifications of the analysis algorithm are sufficient to obtain approximated observation probabilities distinguishing between non-inverted and inverted fault propagation to the circuit outputs. In the following, this new algorithm is called analysis of direction-dependent fault propagation.

Das, Touba, Seuring, and Gössel have already used the analysis of direction-dependent fault propagation for concurrent checking based on weight-based codes [24]. Weight-based codes were introduced by Das and Touba in [23].

For the description of the required modifications a circuit C with n circuit outputs $y_1, \ldots, y_n$ is considered. To distinguish between non-inverted and inverted fault propagation the modified algorithm assigns to each line a matrix of approximated observation probabilities. The matrix consists of two rows and n columns. The first row is related to non-inverted propagation, the second row is related to inverted propagation. Each column of the matrix corresponds to the approximated probabilities, that a fault at the corresponding line is non-inverted or inverted observable at one certain circuit output.

The modified algorithm is similar to `Observabilities`. In an initial step the following matrices are assigned to the circuit outputs $y_1, y_2, \ldots, y_n$:

$$\begin{pmatrix} 1\,0\,0 \ldots 0\,0 \\ 0\,0\,0 \ldots 0\,0 \end{pmatrix} \begin{pmatrix} 0\,1\,0 \ldots 0\,0 \\ 0\,0\,0 \ldots 0\,0 \end{pmatrix} \ldots \begin{pmatrix} 0\,0\,0 \ldots 0\,1 \\ 0\,0\,0 \ldots 0\,0 \end{pmatrix}.$$

The algorithm computes for each line s_i the corresponding $2{\times}n$ matrix of probabilities proceeding in a reverse topological order from the circuit outputs to the circuit inputs. This is done almost analogically to the algorithm `Observabilities` including the following three differences:

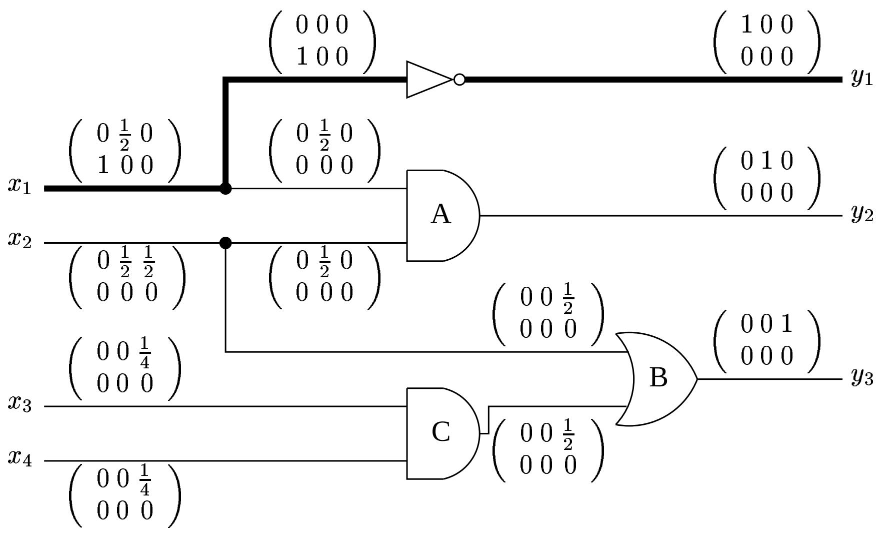
1. A matrix of two rows and n columns instead of one single n-tuple is assigned to a single line. The two rows of a matrix, which are assigned to a line are separately computed in the same way as the single n-tuple assigned to a line by the algorithm `Observabilities`.

2. If the current gate is an XOR-gate or XNOR-gate and the matrix assigned to the gate output consists of the two rows r_1 and r_2, then the following matrix is assigned to each input line:

$$\begin{pmatrix} \frac{1}{2} \cdot (r_1 + r_2) \\ \frac{1}{2} \cdot (r_1 + r_2) \end{pmatrix}.$$

3. If the current gate is an inverter, a NAND-gate, or a NOR-gate, then before assigning the matrices computed for the input lines, the rows of each input matrix are exchanged.

Because of the similarities, the runtime of the modified algorithm can be estimated as two times the runtime of the basic algorithm `Observabilities`. Furthermore, the complexity of the algorithm is equal to the complexity of the basic algorithm. Hence it is linear in the product of the number of lines of C and n, which is the number of circuit outputs of C.



$\begin{pmatrix} 0\,0\,0 \\ 1\,0\,0 \end{pmatrix}$
$\begin{pmatrix} 1\,0\,0 \\ 0\,0\,0 \end{pmatrix}$
y_1
$\begin{pmatrix} 0\,\frac{1}{2}\,0 \\ 1\,0\,0 \end{pmatrix}$
$\begin{pmatrix} 0\,\frac{1}{2}\,0 \\ 0\,0\,0 \end{pmatrix}$
$\begin{pmatrix} 0\,1\,0 \\ 0\,0\,0 \end{pmatrix}$
x_1
A
y_2
x_2
$\begin{pmatrix} 0\,\frac{1}{2}\,\frac{1}{2} \\ 0\,0\,0 \end{pmatrix}$
$\begin{pmatrix} 0\,\frac{1}{2}\,0 \\ 0\,0\,0 \end{pmatrix}$
$\begin{pmatrix} 0\,0\,\frac{1}{2} \\ 0\,0\,0 \end{pmatrix}$
$\begin{pmatrix} 0\,0\,1 \\ 0\,0\,0 \end{pmatrix}$
$\begin{pmatrix} 0\,0\,\frac{1}{4} \\ 0\,0\,0 \end{pmatrix}$
B
y_3
x_3
C
$\begin{pmatrix} 0\,0\,\frac{1}{2} \\ 0\,0\,0 \end{pmatrix}$
x_4
$\begin{pmatrix} 0\,0\,\frac{1}{4} \\ 0\,0\,0 \end{pmatrix}$

Figure 2.17: Example circuit with approximated observation probabilities including the direction of fault propagation

Figure 2.17 shows the results after using the proposed algorithm to analyze the structure of the example circuit which was already used in the previous sections. For example, the bold path starting at the circuit output y_1 is considered: First the matrix

$$\begin{pmatrix} 1\,0\,0 \\ 0\,0\,0 \end{pmatrix}$$

is assigned to the circuit output y_1 since any error at line y_1 is certainly detected without inversion at y_1, but is neither non-inverted nor inverted observable at any other circuit output.

The next gate along the bold path is an inverter. The matrix assigned to the input line of the inverter is derived from the matrix assigned to the output of the inverter by exchanging the rows. The resulting matrix can be explained as follows: Any fault at the input line of the inverter is not detectable at any circuit output except output y_1. Since a fault at the line is only observable at the output y_1, when the faulty value is inverted, the “1” is now in the second row. The matrix assigned to x_1 is the sum of the matrices of the branches of the line.

Since Berger codes detect all unidirectional errors, it is not convenient to group outputs based on the approximated probabilities, that the values of two circuit outputs are simultaneously erroneous. Only non-unidirectional errors are interesting, because unidirectional errors are always detected if the encoding is based on Berger codes. Now the description of the computation of $p'_{i,j}$ follows, which approximates the probability, that the values of two given circuit outputs y_i and y_j are simultaneously erroneous and these values are not equal.

For simplification it is assumed, that for all $s \in S$ the probabilities, that an error $e(s)$ is propagated non-inverted or inverted to the functional outputs y_i and y_j, are independent. The approximated probabilities are given by the matrix m_s assigned to the line s. Let the components of m_s be given by $m_{s,k,l}$ with $1 \leq k \leq 2$ and $1 \leq l \leq n$. Then the product of the two values $m_{s,1,i}$ and $m_{s,2,j}$ is the approximated probability, that an error $e(s)$ is observable as a non-inverted one at the output y_i and is simultaneously observable as an inverted one at the output y_j. Analogically, $m_{s,2,i} \cdot m_{s,1,j}$ approximates the probability, that an error $e(s)$ is observable as an inverted one at the output y_i and is simultaneously observable as a non-inverted one at the output y_j. Thus, the sum of these two products is the approximated probability, that in the presence of a fault the circuit outputs y_i and y_j are simultaneously non-unidirectional erroneous.

Therefore, the values $p'_{i,j}$ can easily be computed using:

$$p'_{i,j} = \frac{1}{|S|} \sum_{s \in S} (m_{s,1,i} \cdot m_{s,2,j} + m_{s,1,j} \cdot m_{s,2,i}).$$

For example, the values $p'_{i,j}$ of the circuit shown in Figure 2.17 are computed as follows: There is only one matrix with both rows unequal 0. This matrix is assigned to the circuit input x_1. This corresponds to the fact, that x_1 is the only line, which is connected to more than one output and the numbers of inverting gates (like inverters, NAND-, NOR-, XOR-, or XNOR-gates) included in the paths to the circuit outputs are different modulo 2. For all other lines and for any pair (y_i, y_j) of circuit outputs the sum $(m_{s,1,i} \cdot m_{s,2,j} + m_{s,1,j} \cdot m_{s,2,i})$ is zero. x_1 is connected to the circuit outputs y_1 and y_2 so that the following results are obtained:

$$\begin{aligned} p'_{1,2} &= \frac{1}{12} \cdot (m_{x_1,1,1} \cdot m_{x_1,2,2} + m_{x_1,1,2} \cdot m_{x_1,2,1}) = \frac{1}{12} \cdot (0 + \frac{1}{2} \cdot 1) = \frac{1}{24}; \\ p'_{2,3} &= 0; \\ p'_{1,3} &= 0. \end{aligned}$$

For any circuit with n circuit outputs $O(n^2)$ different pairs of outputs exist. In the worst case, for each pair (y_i, y_j) and for each line s the sum $(m_{s,1,i} \cdot m_{s,2,j} + m_{s,1,j} \cdot m_{s,2,i})$ must be computed to obtain $p'_{i,j}$. It follows, that the complexity of the analysis is linear in the product of the circuit size and the square of the number of outputs.

2.3 Generation of Groups of Outputs

The previous sections described techniques for analyzing the structure of a given circuit C. In particular, the analysis algorithms estimate the reduction of fault propagation for each pair of circuit outputs of C, if the output pair is included in the same group of the linear space compactor. In this section, methods are described, which use the results of the analysis to form disjoint groups of circuit outputs. These parity groups determine the function of the linear space compactor since each parity of a group equals a compacted output.

2.3.1 Basic Algorithm Determining a Fixed Number of Output Groups

Below an algorithm is described, which determines the parity groups of the linear space compactor. The input parameters of the algorithm are the results of the circuit analysis and the number of groups, which should be generated. In order to obtain an algorithm with polynomially bounded runtime, the set of circuit outputs is heuristically grouped into disjoint groups.

In the following, the input parameters are k, which is the number of groups, and $v_{i,j}$, $1 \leq i < j \leq n$. $v_{i,j}$ denotes the result of the analysis algorithm assigned to the pair (y_i, y_j) of the set of circuit outputs $Y = \{y_1, \ldots, y_n\}$. $v_{i,j}$ is the estimated reduction of fault propagation caused by the assignment of two outputs y_i and y_j to the same parity group of the linear space compactor. Based on the approximations

$v_{i,j}$, $1 \leq i < j \leq n$, the heuristic algorithm determines a partition $\mathcal{G}_k$ of the set Y into k disjoint groups. The groups are $G_1, \ldots, G_k$ with $G_1 \cup \ldots \cup G_k = Y$ and $G_i \cap G_j = \emptyset$ for $i \neq j$.

If only two circuit outputs y_i and y_j are simultaneously erroneous and the values of all other circuit outputs are fault-free, then these errors can only be detected at the compacted outputs, if the erroneous outputs are in different groups. Therefore, if the approximated value $v_{i,j}$ is high, then also the probability is high that the two outputs y_i and y_j are simultaneously erroneous in the presence of a fault. For this reason these outputs should be assigned to different parity groups.

Thus, the basic idea of the grouping algorithm is, that for the given input parameter k the partition $\mathcal{G}_k$ is determined in such a way that the sum

$$P(\mathcal{G}_k) = \sum_{G_l \in \mathcal{G}_k} \left(\sum_{y_i, y_j \in G_l \, \wedge \, i<j} v_{i,j} \right) \tag{2.4}$$

is minimized. If the probability $v_{i,j}$ is small for each pair of circuit outputs y_i and y_j included in the same parity group G_l, then the value $P(\mathcal{G}_k)$ of the partition $\mathcal{G}_k$ is also small.

The heuristic algorithm, which groups the circuit outputs works as follows: To minimize $P(\mathcal{G}_k)$, first the circuit outputs are arranged according to a connection weight determined for each circuit output. The weight estimates how difficult it is to assign the corresponding output to a group without imposing a considerable increase of $P(\mathcal{G}_k)$. Then the outputs are assigned to the groups starting with the outputs whose connection weights are the highest.

The connection weight $W(y_i)$ assigned to each output $y_j \in Y$ is:

$$W(y_i) = \sum_{y_j \in Y \setminus \{y_i\}} v_{i,j}.$$

As described above, the functional outputs are ordered with respect to their connection weights so that the first output y_{i_1} has the maximum weight. The first k outputs $y_{i_1}, \ldots, y_{i_k}$ are assigned to the k different groups $G_1, \ldots, G_k$ resulting in $G_j = \{y_{i_j}\}$, $1 \leq j \leq k$.

The remaining $n - k$ outputs are assigned to the different groups in such a way, that the increase of the sum $P(\mathcal{G}_k)$ is minimized. This is done according to the following rule: Let l, $k \leq l < n$, outputs be already assigned to the groups $G_1, \ldots, G_k$. Then the group $G \in \{G_1, \ldots, G_k\}$ of the next output y_i is chosen so that the sum $w_{G,i}$

$$w_{G,i} = \sum_{y_j \in G} v_{i,j}$$

is minimal. If $w_{G,i}$ is minimal for more than one group, then y_i is assigned to the group with the smallest index. After each circuit output is assigned to one of the k disjoint groups, the heuristic algorithm stops.

The complexity of the heuristic algorithm can be computed as follows: The complexity of determining the connection weights of each circuit output is $O(n^2)$ since for each output $y_i \in \{y_1, \ldots, y_n\}$ $n - 1$ values $v_{i,j}$ are added up. The runtime of sorting the circuit outputs is $O(n \cdot log\ n)$. Finally, the last step of the heuristic algorithm determines the groups of the circuit outputs y_i out of the remaining $n - k$ outputs. Each value $v_{i,j}$, $j \neq i$ and $1 \leq j \leq n$, is at most once added to one of the sums $w_{G,i}$, $G \in \{G_1, \ldots, G_k\}$, which are computed according to the procedure. Thus, the runtime of the last part of the heuristic algorithm is bounded by $O(n^2)$. In summary, the complexity of the heuristic which partitions the set of circuit outputs is quadratic in the number of circuit outputs of C.

2.3.2 Modifications

In the following, two modifications of the heuristic algorithm introduced in the previous section are described. The first modification may produce groups of circuit outputs whose sizes are more equally distributed. After the first k arranged outputs form $G_j = \{y_{i_j}\}$, $1 \leq j \leq k$, the modification changes the way how the remaining $n-k$ arranged outputs are assigned to the different groups: Let l, $k \leq l < n$, outputs be already assigned to the groups $G_1, \ldots, G_k$. Again, the group $G \in \{G_1, \ldots, G_k\}$ of the next output y_i is chosen in such a way, that the sum $w_{G,i}$ is minimal. If $w_{G,i}$ is minimal for more than one group then a modified procedure is used: y_i is assigned to the group with the smallest number of outputs. If this group is not uniquely determined, then out of these groups the one with the smallest index is chosen.

It may happen, for example, that the two outputs of any pair, which consists of an output already assigned to a group and y_i, which is next assigned to a group, are independent. Then it is $w_{G,i} = 0$ for all $G \in \{G_1, \ldots, G_k\}$. In this case the output y_i is assigned to the group containing the smallest number of outputs.

If the sizes of the groups are nearly equal, then the depths of the XOR-trees, which implement the linear compactor are also nearly equal. Since the delay caused by the linear compactor depends on the depth of the largest tree of XOR gates, the modified heuristic may reduce the delay caused by the linear compactor. Furthermore, in most cases the slack of paths through circuit and space compactor is only slightly changed.

The second modification is based on an algorithm for the refinement of the assignments. The algorithm is an additional step after the groups are already determined. It works straightforwardly and tries to improve the results of the heuristic algorithm, so that the value of $P(\mathcal{G}_k)$ is further minimized.

For each circuit output y_i the algorithm computes $P(\mathcal{G}_k)$ for the case that this output is reassigned to another group. If there is a group, so that the new $P(\mathcal{G}_k)$ is smaller compared to the partition given at the start of the algorithm, then the output y_i, the corresponding group, and the achieved reduction of $P(\mathcal{G}_k)$ are stored.

If all circuit outputs have been processed, only that circuit output is removed from the current group and included in the corresponding new group, whose reassignment results in the maximal decrease of $P(\mathcal{G}_k)$. Then the algorithm goes back to its starting point and runs through the next loop. If no reassignment of a circuit output that results in a decrease of $P(\mathcal{G}_k)$ can be found, then the algorithm ends.

An upper bound on the runtime of this additional algorithm can not be easily derived since it is difficult to estimate, how often the algorithm will run through the loop. However, experimental results show, that the results of the heuristic algorithm, which partitions the set of circuit outputs as described in the previous section are so good, that reassignments of outputs, which decrease $P(\mathcal{G}_k)$, can seldom be found. The maximal number of executed loops was six for all experiments. Therefore, under the assumption, that the maximal number of required loops is constant, the complexity of the additional algorithm is bounded by $O(n^2)$.

Below the modification, which forms groups whose sizes are almost equal, and the additional algorithm, which minimizes the value of $P(\mathcal{G}_k)$, are always included. Hence all experimental results presented below have been obtained by the heuristic algorithm including the two modifications described in this section.

Combinational Circuit	Runtime of analysis algorithms based on common gates ratio (s)	propagation distances (s)	approx. observ. probabilities (s)	approx. signal & observ. probabilities (s)
c432	<0.01	<0.01	<0.01	<0.01
c499	<0.01	0.01	0.01	0.01
c880	<0.01	<0.01	<0.01	<0.01
c1355	0.02	0.04	0.02	0.03
c1908	0.01	0.03	0.01	0.02
c2670	0.04	0.03	0.02	0.02
c3540	0.01	0.03	0.01	0.01
c5315	0.07	0.10	0.05	0.06
c6288	0.04	0.12	0.03	0.06
c7552	0.09	0.12	0.06	0.08

Table 2.6: Runtimes required for structure analysis using different algorithms

2.4 Experimental Results

The experimental results are derived for the ten combinational circuits of the ISCAS ’85 benchmarks suite [13]. Table 2.1 on page 15 lists the circuits consisting of 36 to 233 inputs and 7 to 140 outputs.

The structure of the benchmark circuits was analyzed using the algorithms based on common gates ratio, propagation distances, approximated observation probabilities, and approximated signal and observation probabilities. The analysis algorithms were implemented in C++. For analyzing the circuit structures the CPU-times of a computer with an AMD Athlon processor running at 700 MHz under Linux are listed in Table 2.6. Each runtime was measured without the time required for reading the netlist of the circuit whose structure is analyzed. The algorithms require less than 0.2 seconds for each circuit.

Further experiments investigated the runtime required for grouping the outputs according to the results of the structure analysis. The algorithm, which was introduced above for generating disjoint group was implemented in C++. The program was used to determine $2, 3, \dots, 10$ parity groups. The groups were derived from the results of the analysis based on approximated observation probabilities. The CPU-times of the Athlon computer were at most 0.01 seconds for each circuit out of the ten ISCAS ’85 benchmarks circuits listed in Table 2.1.

The heuristic algorithm introduced above determines groups of circuit outputs so that the sum $P(\mathcal{G}_k)$ of the resulting partition $\mathcal{G}_k$ is minimized. Table 2.7 lists these sums $P(\mathcal{G}_k)$ for $k = 1, \dots, 8$ parity groups. The results shows for each of the ten combinational circuits of the ISCAS ’85 benchmarks suite the following: If the input parameter k which determines the number of compacted outputs is increased, then the heuristic algorithm efficiently utilizes the fact, that the set of outputs can be spread over a larger number of groups. Thus, the computed partitions $\mathcal{G}_k$ lead to a smaller sum $P(\mathcal{G}_k)$ if k is increased.

This can been seen more clearly in Figure 2.18, which shows graphically the results in a different way: For the circuits c432, c499, c2670, and c7552 the sum $P(\mathcal{G}_1)$ of the parity of all circuit outputs is set to 100% and for $k = 2, \dots, 8$ the corresponding percentages of $P(\mathcal{G}_k)$ are shown by the graph. The

Combinational Circuit	$P(\mathcal{G}_k)$ for partitions $\mathcal{G}_k$ consisting of k groups computed by the heuristic algorithm							
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
c432	0.65	0.23	0.11	0.05	0.03	0.01	0.00	–
c499	2424	1093	673	428	325	254	201	152
c880	6.28	1.82	0.98	0.52	0.34	0.16	0.11	0.06
c1355	33.93	13.75	8.04	3.66	2.78	2.05	1.64	1.43
c1908	12.90	5.45	3.09	2.06	1.34	0.80	0.56	0.37
c2670	81.51	24.18	10.22	5.30	3.20	2.29	1.46	0.89
c3540	21.27	8.98	5.33	3.56	2.32	1.67	1.28	0.99
c5315	274.17	112.60	61.32	37.81	22.38	15.56	11.92	9.41
c6288	75.40	25.96	12.01	7.16	4.80	3.47	2.74	2.11
c7552	165.08	63.35	35.30	22.40	15.30	11.87	9.52	7.88

Table 2.7: Results of the heuristic algorithm for grouping outputs

graphs of the other six circuits are omitted in order to obtain a clearly arranged diagram. The graphs of the missing circuits are very similar and do not exceed the minimal and maximal bounds given by the shown graphs.

The reason for the high values $P(\mathcal{G}_k)$ obtained for the circuit c499 is, that almost half of the gates of this circuit are XOR-gates. For XOR-gates the estimated probability $p(g)$, that an error at the inputs of the gate g is propagated to the outputs of the gate, is $p(g) = 1$. Many gates with $p(g) = 1$ and a large number of reconvergent paths result in drastically increased values $P(\mathcal{G}_k)$. In contrast, all other circuits except c499 contain no or only a small number of XOR-gates.

Basically, $P(\mathcal{G}_k)$ depends on propagation probabilities and on certain characteristics of the circuit: The number of lines, the number of circuit outputs, the number of gates with $p(g) = 1$ and the number of reconvergent paths. Because these characteristics usually differ considerably for varying circuits, $P(\mathcal{G}_k)$ can not be used to compare the testability between two circuits or to obtain a general measurement of testability for circuits with compacted outputs.

Table 2.8 shows the sizes of the parity groups, which the heuristic algorithm computed using the results of the analysis on approximated observation probabilities. For each of the ten combinational circuits and for each compactor with $k = 2, \ldots, 8$ outputs the size of the smallest and largest output group is listed. For all compactors the ratio of the size of the largest group to the size of the smallest group is at most 2. Hence the difference between the depths of two parity trees is at most 1 for all investigated compactors.

2.4.1 Simulation of Stuck-At Faults

Several analysis algorithms for circuits were discussed in the previous sections. The analysis results are used to partition the set of circuit outputs into k disjoint groups of outputs. These parity groups determine the linear space compactor with k compacted outputs. The reduction of the testability of the circuit caused by the output compaction is estimated by the sum $P(\mathcal{G}_k)$.

In this section, it is investigated, how exactly the reduction of testability of the circuit with compacted outputs is estimated by $P(\mathcal{G}_k)$. The underlying analysis algorithms are the analysis based on

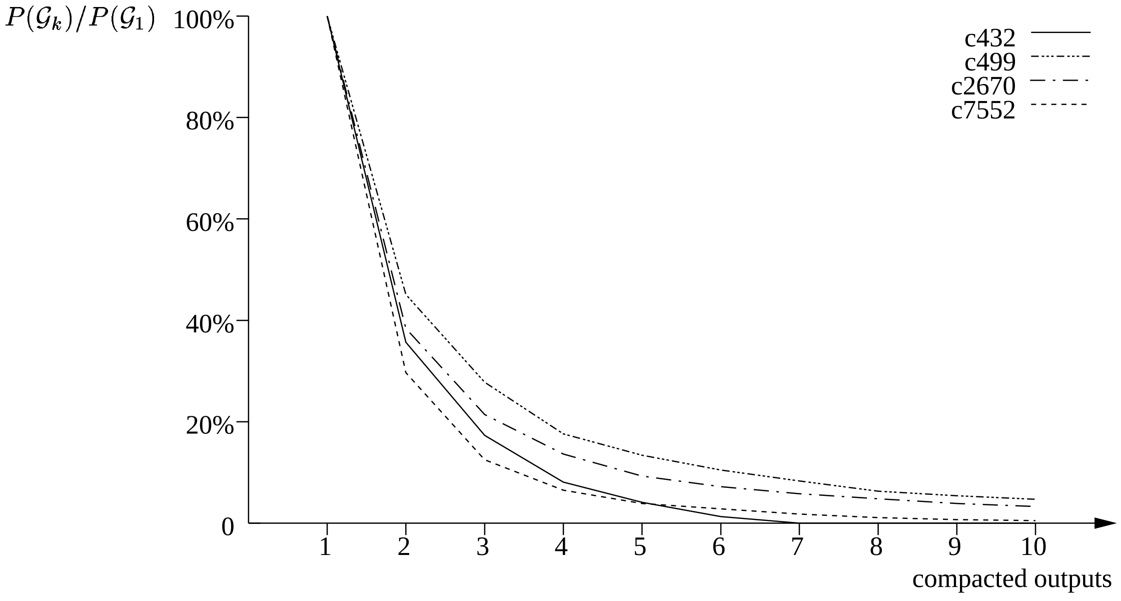


Figure 2.18: Results of the heuristic algorithm for grouping outputs

approximated observation probabilities and the analysis which includes additionally approximated signal probabilities. Experimental results given above show, that among the proposed analysis algorithms these are the most accurate analysis techniques with respect to the single stuck-at fault model. The sum $P(\mathcal{G}_k)$ obtained using these analysis algorithms are compared to results obtained from fault simulations of single stuck-at faults. The fault simulations were carried out for circuits with no output compaction and with compacted outputs generated according to the partition $\mathcal{G}_k$.

Furthermore, the results of these fault simulations are used to answer the question, how the proposed linear space compactors deal with the issue of error masking for single stuck-at faults.

The testability of circuits with the proposed linear space compactors is investigated for concurrent on-line checking, for testing with a deterministic test set, and for testing based on pseudo random input vectors. In the following, an original circuit checked or tested without output compaction is denoted by C and a circuit with k compacted outputs obtained by an output partition $\mathcal{G}_k$ is denoted by C_k.

2.4.1.1 On-Line Checking

The comparison of the on-line testability of circuits without and with the proposed linear space compactors was carried out as follows: Let $e(s)$ be a single stuck-at fault at the circuit line s and let $C(e(s))$ be the circuit C in the presence of the fault $e(s)$. A sequence of 100000 pseudo random input vectors were applied first to the circuit inputs of the faulty circuit $C(e(s))$ without output compaction and then to the circuit inputs of the faulty circuit $C_k(e(s))$ with the output compactor derived from the output partition $\mathcal{G}_k$. Let $n(e(s))$ ($n_k(e(s))$) be the number of pseudo random input vectors for which at least one of the checked outputs of $C(e(s))$ ($C_k(e(s))$) is erroneous so that the fault can be detected. If the

Combinational Circuit	Minimal/maximal group size for partitions $\mathcal{G}_k$ consisting of k groups computed by the heuristic algorithm													
	$k=2$		$k=3$		$k=4$		$k=5$		$k=6$		$k=7$		$k=8$	
c432	3	4	2	3	1	2	1	2	1	2	1	1	–	–
c499	16	16	10	11	8	8	6	7	5	6	4	5	4	4
c880	13	13	8	9	6	7	5	6	4	5	3	5	3	5
c1355	16	16	10	11	8	8	6	7	4	6	4	5	4	4
c1908	12	13	8	9	6	7	5	5	4	5	3	4	3	4
c2670	69	71	46	47	35	35	26	29	22	24	19	21	17	18
c3540	10	12	6	8	5	6	4	5	3	4	2	4	2	3
c5315	61	62	40	42	29	34	24	25	19	21	16	20	14	18
c6288	16	16	10	11	8	8	6	7	5	6	4	5	3	5
c7552	54	54	34	40	26	28	16	31	15	22	13	19	11	19

Table 2.8: Sizes of the output groups which are computed by the heuristic algorithm

outputs of C are compacted according to the partition $\mathcal{G}_k$ then

$$\mu_{k,e(s)} = \frac{n(e(s)) - n_k(e(s))}{n(e(s))}$$

estimates the probability that an error at the original functional outputs caused by the fault $e(s)$ is not observable at one of the k compacted outputs.

For E_C, which is the set of all single stuck-at faults of the circuit C,

$$\mu_k = \frac{1}{|E_C|} \cdot \sum_{e(s)\in E_C} \mu_{k,e(s)} \tag{2.5}$$

is the expected value of the probability, that after applying a pseudorandom input vector an arbitrarily chosen single stuck-at fault is propagated to the outputs of the original circuit C but not to the compacted outputs of C_k.

The results were obtained using the fault simulator `Minos` which is briefly described in the appendix of this thesis.

Analysis Based on Observation Probabilities Table 2.9 gives the experimentally determined values $\mu_k, k = 1, \ldots, 8$, for circuits with k compacted outputs derived from the results of the analysis based on approximated observation probabilities. The case $k = 1$ describes the simple space compactor which consists of a single parity tree of all functional outputs.

For each benchmark circuit a linear space compactor with $k = 5$ outputs is sufficient to obtain $\mu_k < 2\%$. On average $k = 3.6$ compacted outputs are necessary to meet this requirement.

A comparison of μ_1 and μ_2 yields the following: If two parity groups are selected instead of using a single parity tree, then the probability, that a fault is observable at the functional outputs but not at the compacted outputs, is considerably reduced. The reduction of the masking probability is at least 48% and reaches an average 65.4%.

Figure 2.19 presents graphically the experimental results: For the circuits c432, c499, c2670, and c7552 the value μ_1 obtained for a single parity tree of all circuit outputs is set to 100%. For $k = 2, \ldots, 8$

Combinational Circuit	Reduction of on-line testability for SSFs (%)							
	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
c432	34.28	17.54	7.25	3.56	1.28	1.11	0	–
c499	9.52	3.83	2.26	0.36	0.72	0.57	0.30	0.15
c880	8.09	3.17	1.42	0.87	0.47	0.29	0.12	0.08
c1355	5.44	2.38	0.86	0.13	0.10	0.08	0.37	0.06
c1908	14.55	5.57	3.18	1.72	0.24	0.09	1.35	1.33
c2670	30.00	3.90	1.07	0.27	0.04	0.18	0.05	0.06
c3540	17.17	7.45	2.92	2.06	1.25	0.55	0.29	0.12
c5315	27.75	11.48	1.54	4.42	1.64	2.28	0.97	0.13
c6288	31.43	6.63	1.27	0.28	0.07	0.01	<0.01	<0.01
c7552	9.06	2.89	1.16	0.66	0.37	0.29	0.17	0.15
average	18.73	6.48	2.29	1.43	0.62	0.55	0.36	0.21

Table 2.9: On-line testability of circuits with output compactors derived from approximated observation probabilities

the corresponding percentages of μ_k/μ_1 are shown by the graph. A comparison with Figure 2.18 obtained for the values $P(\mathcal{G}_k)$ in the same way yields the following: The graph of μ_k is less even and includes parts with positive gradients. For $k = 2, \dots, 8$ the ratio μ_k/μ_1 obtained for the circuit c2670 is less than the same ratio computed for c7552. But the corresponding approximations $P(\mathcal{G}_k)/P(\mathcal{G}_1)$ of c2670 are greater than those of c7552. This indicates, that the improvement of testability can not be exactly determined from the reduction of $P(\mathcal{G}_k)$.

Figure 2.20 shows the dependence between μ_k and $P(\mathcal{G}_k)$ for the benchmark circuits c880, c2670, and c3540. μ_k and $P(\mathcal{G}_k)$ are nearly linearly dependent. Similar graphs are obtained for all other circuits which have been investigated. These results show, that the minimization of $P(\mathcal{G}_k)$ is in general a good criterion for the optimization of μ_k.

There is a small number of exceptions concerning the linear dependency: The results for compactors of the circuits c499, c1355, c1908, c2670, and c5315 show, that the on-line testability of circuits with output compaction, which is generated based on approximated observation probabilities can get worse although the number of compacted outputs is increased. For example, for c5315 it is $\mu_3 = 1.54$ for 3 compacted outputs but $\mu_4 = 4.42$ for one more compacted output. In contrast, the corresponding values of the sum computed by the grouping algorithm are $P(\mathcal{G}_3) = 61.32$ and $P(\mathcal{G}_4) = 37.81$. The results, i. e. $\mu_3 < \mu_4$ and $P(\mathcal{G}_3) > P(\mathcal{G}_4)$, indicate, that the underlying analysis of the circuit structure computes only approximated values.

Since in general the approximated values of the analysis are sufficiently accurate, there are only a few cases as described above. These cases are caused by inaccurate approximations computed for output pair, if for a small number of output pairs (y_i, y_j) the analysis algorithm computes a low value $p_{i,j}$ even if it is very likely, that the outputs y_i and y_j are simultaneously erroneous. For example, such incorrect values are produced by reconvergent paths, which are part of the transitive fanins of both outputs, y_i and y_j. If $p_{i,j}$ is low, then the heuristic grouping the outputs is not forced to assign the corresponding outputs y_i and y_j to different groups. In this case it may happen, that for k compacted outputs y_i and y_j are assigned to different groups and for $k+1$ compacted outputs the algorithm assigns y_i and y_j to the same parity group resulting in $P(\mathcal{G}_k) > P(\mathcal{G}_{k+1})$ and $\mu_k < \mu_{k+1}$.

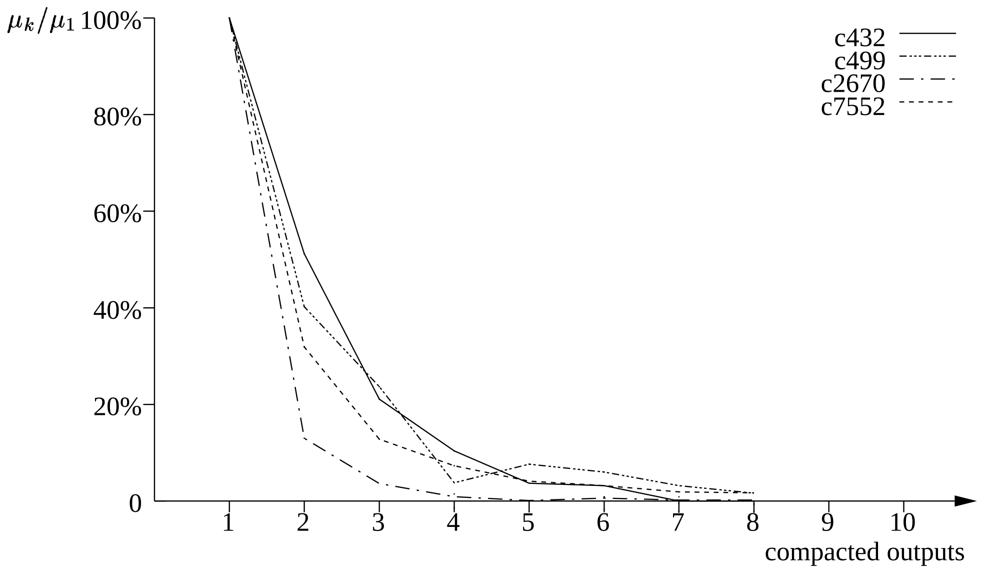


Figure 2.19: On-line testability of circuits with output compactors derived from approximated observation probabilities

Analysis Based on Approximated Signal and Observation Probabilities The compactors were derived from the results of the analysis based on approximated signal and observation probabilities. Again, the case $k = 1$ denotes linear space compaction using a single parity tree of all functional outputs. Table 2.10 gives the values of μ_k experimentally obtained by simulating circuits with $k = 1, \ldots, 8$ compacted outputs.

A linear space compactor with $k = 5$ outputs is sufficient for each benchmark circuit to obtain $\mu_k < 2\%$. On average $k = 3.7$ compacted outputs are necessary to achieve $\mu_k < 2\%$ if the generation of the compactors is based on approximated signal and observation probabilities. The average number of compacted outputs is only $k = 3.6$, if the underlying analysis does not utilize approximated signal probabilities as in the previous section described.

Similar to the the previous section, the results for compactors of the circuits c1908, c2670, c6288, and c7552 show, that the on-line testability of circuits with a compactor, which is generated based on approximated signal and observation probabilities, can get worse although the number of compacted outputs is increased.

Graphs showing μ_k in dependence on the number of compacted outputs k or in dependence on $P(\mathcal{G}_k)$ are omitted. They look similar to the corresponding graphs of Figure 2.19 and Figure 2.20, that are given in the previous section for compactors derived without approximation of signal probabilities.

Comparison of the Results Obtained from Different Analysis Algorithms The comparison of the experimental results for compactors derived from the two different analysis algorithms is based on the Tables 2.9 and 2.10. The results of Table 2.9 correspond to the analysis based on approximated observation probabilities and the results of the Table 2.10 are related to the analysis including approximated signal probabilities.

For each $k, k = 2, \ldots, 8$, lower values μ_k are obtained for the circuit c6288, if the analysis algo-

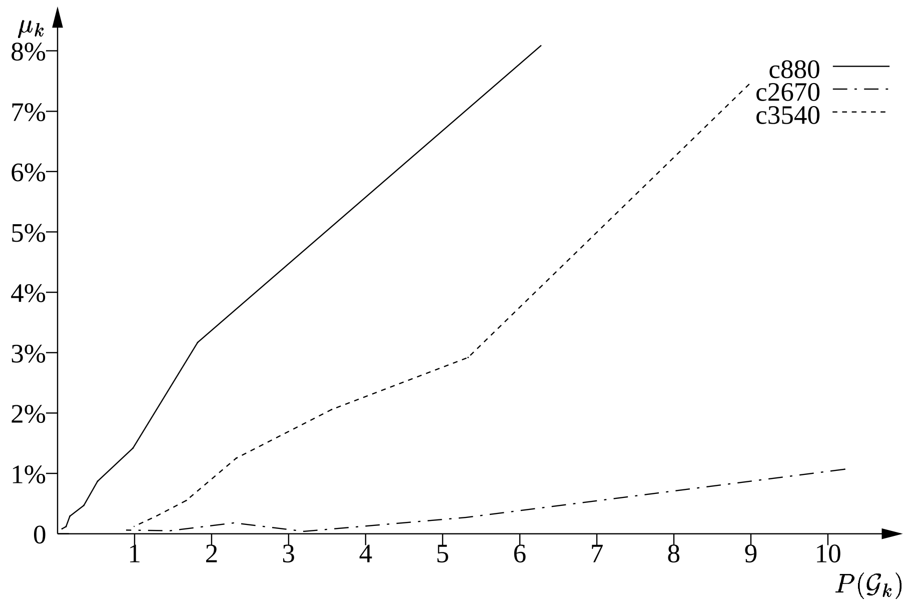


Figure 2.20: Comparison between values computed by the analysis algorithm and results of fault simulation

rithm without approximated signal probabilities is used. The on-line testability of the circuits c432, c499, c880, and c1908 combined with a compactor is better for $2, \ldots, 8$ outputs, if the generation of the compactors is based on the analysis algorithm including approximated signal probabilities. For the other circuits it depends on the number k of compacted outputs, which analysis results in a better testability.

The comparison shows, that none of the two analysis algorithms is generally the best generating always the compactor with maximal on-line testability for all circuits and for any number of compacted outputs. But in many cases the analysis, which includes approximated signal probabilities results in better compactors compared to the analysis based only on approximated observation probabilities. This result corresponds to the comparison between the standard deviations given in Section 2.2.6 for the results of the analysis algorithms.

2.4.1.2 Deterministic Test Sets

This section describes the results obtained for deterministic tests of circuits without and with the proposed linear space compactors. The untestable single stuck-at faults for the circuits C (without compactor) and C_k (with compactor) were determined using the automatic test pattern generator of `SIS` [72]. For each resulting test set the number of test vectors were counted.

Let t be the number of non-redundant single stuck-at faults, which are testable at the functional outputs of the original circuit C, and t_k the number of the faults, which are observable at the compacted outputs of the circuit C_k, if the computed test set is applied. Then

$$r_k = \frac{t - t_k}{t}$$

Combinational Circuit	Reduction of on-line testability for SSFs (%)							
	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
c432	34.28	17.39	6.96	2.48	1.15	0.08	0	–
c499	9.52	3.83	1.85	0.36	0.28	0.22	0.18	0.15
c880	8.09	2.40	1.09	0.45	0.33	0.15	0.10	0.07
c1355	5.44	2.38	1.04	0.13	0.10	0.08	0.07	0.06
c1908	14.55	5.45	1.67	1.68	0.23	0.05	1.32	1.32
c2670	30.00	3.89	0.94	0.31	0.32	0.03	0.01	<0.01
c3540	17.17	6.20	4.11	2.15	0.62	0.23	0.14	0.05
c5315	27.75	13.05	4.75	4.37	1.13	0.21	0.13	0.02
c6288	31.43	8.35	2.64	0.70	0.72	0.48	1.09	1.63
c7552	9.06	3.00	1.06	0.60	0.28	0.02	<0.01	0.04
average	18.73	6.59	2.61	1.32	0.52	0.16	0.30	0.33

Table 2.10: On-line testability of circuits with output compactors derived from approximated signal and observation probabilities

is the ratio of the originally testable faults of the circuit C, which are not detected at the outputs of C_k because of the output compaction.

Analysis Based on Observation Probabilities Experiments were carried out to determine the set of faults, which are detected by deterministic test of the circuits C_k with $k = 1, \ldots, 4$ compacted outputs. The compactors were determined using the results of the analysis based on approximated observation probabilities.

Starting from the left, each row of Table 2.11 gives the name of the investigated circuit, the number of detectable single stuck-at faults of $C, C_1, \ldots, C_4$, and the value r_k for $k = 1, \ldots, 4$, respectively. All results are given with respect to collapsed stuck-at faults.

If k_0 is the minimal number of compacted outputs achieving $t_{k_0} = t$ and $r_{k_0} = 0$, then for each circuit the value for k_0 is always less than or equal to 3 except for the circuit c5315 ($k_0 = 4$). Hence for most circuits an output compaction with 3 parity groups is sufficient to avoid any reduction of the testability for deterministic tests.

This good result for the deterministic test might be surprising since the analysis algorithm was developed to estimate only fault masking during on-line checking. The good testability can be explained in the following way: If during on-line checking only for a few input patterns a small number of faults is masked by the compactor, then it is likely, that for a certain fault at least one input pattern exists, which propagates the fault to the compacted outputs. In this case, the error at the functional outputs is not masked for all input patterns and the fault remains detectable.

Table 2.12 shows the length $|T|$ of the compacted test for all testable single stuck-at faults of the original circuit C. It lists also $|T_k|, k = 1, \ldots, 4$, which is the length of a compacted test for C_k. "–" means, that a test set could not be generated due to an error of the automatic test pattern generator. The two rightmost columns give $|T_{k_0}|$ and the ratio $|T_{k_0}|/|T|$. This ratio measures the increase of the size of the test set, if all non-redundant faults of the original circuit must be detected and under this condition the minimal number of compacted outputs is chosen.

Combinational	Number of untested SSFs					Ratio $r_k = (t - t_k)/t$ (%)			
Circuit	C	C_1	C_2	C_3	C_4	$k=1$	$k=2$	$k=3$	$k=4$
c432	4	4	4	4	4	0	0	0	0
c499	8	32	16	8	8	2.61	0.87	0	0
c880	0	6	0	0	0	0.78	0	0	0
c1355	8	32	16	8	8	1.95	0.65	0	0
c1908	7	19	7	7	7	0.68	0	0	0
c2670	112	722	118	112	112	26.03	0.26	0	0
c3540	135	168	136	135	135	1.05	0.03	0	0
c5315	59	69	64	61	59	0.21	0.10	0.04	0
c6288	34	37	34	34	34	0.05	0	0	0
c7552	133	303	141	133	135	2.51	0.12	0	0.03

Table 2.11: Deterministic test of circuits with output compactors derived from approximated observation probabilities

Combinational	Length of generated tests						
Circuit	$\|T\|$	$\|T_1\|$	$\|T_2\|$	$\|T_3\|$	$\|T_4\|$	$\|T_{k_0}\|$	$\|T_{k_0}\|/\|T\|$
c432	60	79	63	63	59	79	1.32
c499	56	–	–	57	56	57	1.02
c880	91	–	86	88	84	86	0.95
c1355	88	–	–	96	93	96	1.09
c1908	136	–	154	145	156	154	1.13
c2670	153	–	–	143	154	143	0.93
c3540	211	–	–	215	226	215	1.02
c5315	167	185	170	160	168	168	1.01
c6288	39	61	43	42	42	43	1.10
c7552	319	–	325	313	–	313	0.98

Table 2.12: Lengths of deterministic tests for circuits without and with output compaction

The required test length increases at most by a factor 1.32 and on average only by a factor 1.06. For some circuits the number of the test patterns is reduced. In this case a smaller test set is found by the heuristic algorithm of the test pattern generator, which compacts the set of test patterns.

The experiments also show, that almost all faults of the XOR-gates of the compactors are testable, too. The only exception is a single fault in the compactors of the circuit c2670.

Analysis Based on Approximated Signal and Observation Probabilities Now circuits C_1, C_2, and C_3 with 1, 2, and 3 compacted outputs derived from the analysis based on approximated signal and observation probabilities are investigated. Again, the automatic test pattern generator of `SIS` [72] was used to compute a compact test set and to determine the number of untestable faults of these circuits.

Table 2.13 lists the names of the circuits, the numbers of undetected collapsed stuck-at faults of $C, C_1, \ldots, C_3$, and the values r_1, r_2, and r_3. The minimal number of compacted outputs k_0 required

Combinational	Number of untested SSFs				Ratio $r_k = (t - t_k)/t$ (%)				
Circuit	C	C_1	C_2	C_3	$k = 1$	$k = 2$	$k = 3$	$\lvert T_{k_0}\rvert$	$\lvert T_{k_0}\rvert/\lvert T\rvert$
c432	4	4	4	4	0	0	0	79	1.32
c499	8	32	16	8	2.61	0.87	0	55	0.98
c880	0	6	0	0	0.78	0	0	87	0.96
c1355	8	32	12	8	1.95	0.32	0	89	1.01
c1908	7	19	9	7	0.68	0.11	0	145	1.07
c2670	112	722	112	112	26.03	0	0	158	1.03
c3540	135	168	135	135	1.05	0	0	225	1.07
c5315	59	69	61	59	0.21	0.04	0	166	0.99
c6288	34	37	34	34	0.05	0	0	44	1.13
c7552	133	303	133	133	2.51	0	0	304	0.95

Table 2.13: Deterministic test of circuits with output compactors derived from approximated signal and observation probabilities

for $r_{k_0} = 0$ is less than or equal to 3 for all benchmark circuits. The two rightmost columns of Table 2.13 give for each circuit the size of the test set $|T_{k_0}|$ for C_{k_0} and the ratio $|T_{k_0}|/|T|$.

The required test length increases at most by a factor 1.32, and on average the test length increases only by a factor 1.05. For some circuits the number of the test patterns is reduced.

The experiments show, that also almost all faults of the gates of the XOR-trees are testable. Again, the only exception is a single fault located in the space compactors which are designed for the circuit c2670.

Comparison of the Results Obtained from Different Analysis Algorithms If the analysis includes approximated signal probabilities, then in many cases better results can be expected. The minimal number of compacted outputs k_0, which is required to avoid any reduction of testability, is smaller for the circuits c2670, c3540, c5315, and c7552; only for the circuit c1908 it is larger in comparison to the results obtained for the analysis, which approximates only observation probabilities. Furthermore, the average size of the computed test sets is slightly smaller if the analysis includes approximated signal probabilities.

The results of the two analysis algorithms can be combined in such a way, that for each circuit and for each number of compacted outputs the analysis is chosen which results in the compactor with the better results. Then for seven out of ten benchmark circuits two compacted outputs are sufficient to avoid any reduction of testability. For the remaining circuits three compacted outputs are necessary.

2.4.1.3 Pseudo Random Testing

In this section, the effect of the proposed compactor designs on pseudo random testing is investigated. This is done by fault simulation as follows: 100000 pseudo random input vectors are applied to the original circuit C and to the circuit C_k which is related to the output partition $\mathcal{G}_k$. The fault simulator `Minos` is used to determine d and d_k, which are the numbers of the detected stuck-at faults of C and C_k. For the simulations of C_k only the stuck-at faults of the original circuit C are considered. Thus, additional faults within the XOR-trees of the compactors are not taken into account.

Combinational	Number of undetected SSFs					Ratio $q_k = (d - d_k)/d$ (%)			
Circuit	C	C_1	C_2	C_3	C_4	$k = 1$	$k = 2$	$k = 3$	$k = 4$
c432	10	10	10	10	10	0	0	0	0
c499	8	56	24	8	8	4.85	1.62	0	0
c880	0	10	0	0	0	0.57	0	0	0
c1355	8	56	24	8	8	1.78	0.59	0	0
c1908	11	23	11	11	11	0.32	0	0	0
c2670	620	1897	628	620	620	27.06	0.17	0	0
c3540	256	336	256	256	256	1.17	0	0	0
c5315	62	78	66	62	62	0.15	0.04	0	0
c6288	68	75	68	68	68	0.06	0	0	0
c7552	602	910	624	602	608	2.12	0.15	0	0.04

Table 2.14: Pseudo random test of circuits with output compactors derived from approximated observation probabilities

The quotient

$$q_k = \frac{d - d_k}{d}$$

determines the fraction of faults, which are detected at the outputs of the original circuit, but not at the compacted outputs of the circuit C_k.

Analysis Based on Observation Probabilities The analysis based on approximated observation probabilities were used to design compactors for the ten benchmarks circuits. The experimental results obtained after applying 100000 pseudo random test vectors to these circuits with and without additional output compactors are presented in Table 2.14. Each row gives the name of the investigated circuit, the number of undetected single stuck-at faults of $C, C_1, \ldots, C_4$, and the value of $q_k, k = 1, \ldots, 4$, respectively.

The results show, that a single parity tree compacting all functional outputs to one compacted output causes fault masking for nine out of ten circuits during the pseudo random test. If compactors with two outputs are used, then fault masking occurs only for half of the circuits. Finally, for each circuit three compacted outputs are sufficient to detect at the compacted outputs all faults, which are observable at the functional outputs.

Analysis Based on Approximated Signal and Observation Probabilities Again, the fault simulator `Minos` is used to apply 100000 pseudo random test vectors and to determine how many single stuck-at faults are not detected. Table 2.15 lists these numbers for the original circuit C and for the circuits $C_1, \ldots, C_4$. For each circuit $C_1, \ldots, C_4$ with $k = 1, \ldots, 4$ compacted outputs the loss of fault coverage q_k, which was computed as described above, is also given in the table.

If compactors with two outputs are added, then the fault coverage is reduced only for four circuits. No fault masking occurs for compactors with 3 outputs.

Combinational	Number of undetected SSFs					Ratio $q_k = (d - d_k)/d$ (%)			
Circuit	C	C_1	C_2	C_3	C_4	$k = 1$	$k = 2$	$k = 3$	$k = 4$
c432	10	10	10	10	10	0	0	0	0
c499	8	56	24	8	8	4.85	1.62	0	0
c880	0	10	0	0	0	0.57	0	0	0
c1355	8	56	16	8	8	1.78	0.30	0	0
c1908	11	23	13	11	11	0.32	0.05	0	0
c2670	620	1897	620	620	620	27.06	0	0	0
c3540	256	336	256	256	256	1.17	0	0	0
c5315	62	78	66	62	62	0.15	0.04	0	0
c6288	68	75	68	68	68	0.06	0	0	0
c7552	602	910	602	602	621	2.12	0	0	0.13

Table 2.15: Pseudo random test of circuits with output compactors derived from approximated signal and observation probabilities

Comparison of the Results Obtained from Different Analysis Algorithms The two compactor designs yield very similar results for pseudo random test. The analysis algorithm including approximated signal probabilities leads to slightly better values for c1355, c2670, and c7552. The other analysis only produces for the circuit c1908 a compactor with two outputs, which causes no reduction of the fault coverage.

2.4.2 Simulation of Transition Faults

Although output space compactors are designed to propagate faults of a certain model, compactors should also propagate faults of other models or unmodelled faults which can occur in practice. In most cases the occurrence of unmodelled faults depends on the physical implementation of the circuit and the characteristics of the production process. Here the effect of the compactors on the propagation of another fault model is investigated. The considered faults are transition faults, while the design of space compactors introduced above targets at the propagation of single stuck-at faults.

Again, the underlying analysis methods considered in the following are the analysis based on approximated observation probabilities and the analysis, which additionally includes approximated signal probabilities. The testability of circuits with space compactors, which are designed according to the results of the two analysis algorithms, is investigated for concurrent on-line checking and for testing based on pseudo random input vectors.

The experimental results were obtained using the fault simulator `Minos`, which supports the simulation of transition faults.

2.4.2.1 On-Line Checking

The experiments investigating the on-line testability with respect to transition faults were carried out similarly to the experiments for the model of single stuck-at faults. A sequence of 100000 pseudo random input vectors were applied to circuits without and with linear space compactors in order to determine μ_k as defined in Equation (2.5) on page 50 for a circuit C_k with k compacted outputs: If E_C

Combinational Circuit	Reduction of on-line testability for transition faults (%)							
	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
c432	34.33	17.50	7.39	3.66	1.28	1.11	0	–
c499	9.52	3.83	2.25	0.35	0.72	0.58	0.30	0.15
c880	8.11	3.14	1.42	0.85	0.46	0.26	0.11	0.08
c1355	5.43	2.37	0.86	0.13	0.10	0.08	0.37	0.06
c1908	14.56	5.57	3.18	1.73	0.24	0.08	1.35	1.31
c2670	27.54	3.88	1.07	0.27	0.04	0.18	0.05	0.06
c3540	17.17	7.45	2.94	2.08	1.26	0.56	0.30	0.12
c5315	27.78	11.50	1.54	4.44	1.66	2.29	0.96	0.14
c6288	31.39	6.60	1.26	0.28	0.07	0.01	<0.01	<0.01
c7552	9.06	2.89	1.16	0.66	0.37	0.29	0.17	0.15
average	18.49	6.47	2.31	1.45	0.62	0.54	0.36	0.21

Table 2.16: On-line testability of circuits with output compactors derived from approximated observation probabilities

is the set of all transition faults of C and $n(e(s))$ ($n_k(e(s))$) is the number of pseudo random inputs, for which at least one of the outputs of C (C_k) is erroneous under the presence of the fault $e(s) \in E_C$, then it is

$$\mu_k = \frac{1}{|E_C|} \cdot \sum_{e(s)\in E_C} \frac{n(e(s)) - n_k(e(s))}{n(e(s))}.$$

Analysis Based on Observation Probabilities Table 2.16 lists the experimental results obtained for circuits with $k = 1, \ldots, 8$ compacted outputs. The values μ_k are given with respect to compactors derived from the results of the analysis based on approximated observation probabilities.

Only a few results differ from the values obtained for the simulation of single stuck-at faults given in Table 2.9. The differences are small. Again, for each investigated benchmark circuit a compactor with $k = 5$ outputs is sufficient to obtain $\mu_k < 2\%$. On average, $k = 3.5$ compacted outputs are necessary to meet this requirement.

Analysis Based on Approximated Signal and Observation Probabilities For circuits with k compacted outputs the values $\mu_k, k = 1, \ldots, 8$, were experimentally determined for transition faults. The results are very similar to the results given in Table 2.10 with respect to single stuck-at faults. Therefore, a table showing the results for the simulation of transition faults is omitted.

Comparison of the Results Obtained from Different Analysis Algorithms The values μ_k for transition faults, which are obtained from simulations of circuits with compacted outputs, are almost equal to the results for single stuck-at faults. Therefore, a comparison of the values μ_k derived for the two different analysis algorithms yields the same outcome for transition faults as for single stuck-at faults: For each $k, k = 2, \ldots, 8$, lower values μ_k are obtained for the circuit c6288, if the compactors are derived from the analysis based on approximated observation probabilities only. The

Combinational	# undetected transition faults					Ratio $q_k = (d - d_k)/d$ (%)			
Circuit	C	C_1	C_2	C_3	C_4	$k = 1$	$k = 2$	$k = 3$	$k = 4$
c432	10	10	10	10	10	0	0	0	0
c499	8	56	24	8	8	4.85	1.62	0	0
c880	5	15	5	5	5	0.57	0	0	0
c1355	35	83	51	35	35	1.79	0.60	0	0
c1908	11	23	11	11	11	0.32	0	0	0
c2670	772	1918	780	772	772	25.09	0.18	0	0
c3540	260	345	260	260	260	1.25	0	0	0
c5315	63	79	67	63	63	0.15	0.04	0	0
c6288	85	92	85	85	85	0.06	0	0	0
c7552	742	1055	764	742	748	2.18	0.15	0	0.04

Table 2.17: Pseudo random test of circuits with output compactors derived from approximated observation probabilities

on-line testability of the circuits c432, c499, c880, and c1908 combined with a compactor improves, if the generation of compactors additionally includes the approximation of signal probabilities.

2.4.2.2 Pseudo Random Testing

The effect of the proposed compactor design on pseudo random testing of transition faults is investigated similarly to the experiments for the model of single stuck-at faults. A sequence of 100000 pseudo random input vectors were applied to circuits without and with linear space compactors. All possible transition faults are simulated in order to determine d, which is the number of detected faults at the functional outputs of the original circuit C. d_k, which is the number of detected faults at the k compacted outputs of the circuit C_k, is determined in the same way. For the simulation of C_k only the transition faults of the original circuit C are considered. Additional faults within the XOR-trees of the compactors are not taken into account.

The quotient $q_k = \frac{d-d_k}{d}$ determines the fraction of faults which are detectable at the outputs of the original circuit, but which are not detected at the compacted functional outputs of the circuit C_k.

Analysis Based on Observation Probabilities The analysis based on approximated observation probabilities was used to design space compactors for the investigated circuits. The experimental results obtained after simulating the circuits with these compactors are presented in Table 2.17. Each row gives the name of the investigated circuit, the corresponding number of undetected transition faults of $C, C_1, \ldots, C_4$, and the corresponding value of $q_k, k = 1, \ldots, 4$, respectively.

The results show, that a single parity tree, which compacts the functional outputs to one output, results in fault masking for nine out of ten circuits during the pseudo random test. If compactors with two outputs are used, then fault masking occurs only for half of the circuits. Finally, if the number of compacted outputs of the investigated circuits is three, then any fault, which is observable at the functional outputs, can be detected at the compacted outputs.

The fault coverages achieved for transition faults are almost as high as the fault coverages obtained from the simulations of single stuck-at faults.

Analysis Based on Approximated Signal and Observation Probabilities For transition faults the values $q_k = \frac{d-d_k}{d}$ are almost equal to the results given in Table 2.15 for single stuck-at faults. Therefore, a table showing the results for transition faults is omitted.

Comparison of the Results Obtained from Different Analysis Algorithms The values q_k, which were obtained by simulating transition faults of the circuits with k compacted outputs, are very similar to the results for the simulation of single stuck-at faults. Therefore comparing these values yields the same outcome as given above for single stuck-at faults: The use of compactors derived from the results of the analysis algorithm including approximated signal probabilities and the compactors generated without approximation of signal probabilities leads to very similar results for pseudo random test. The former algorithm results in slightly better values q_k for c1355, c2670, and c7552. Only the latter algorithm produces no reduction of the fault coverage for c1908 if the number of compacted outputs is two.

2.4.3 Comparison with Other Approaches

In this section, the experimental results of space compactors proposed above are compared with the results of other space compactor designs given in the literature.

As already mentioned in the introduction of this chapter, Chakrabarty and Hayes investigate two different approaches in [18]. In both approaches first a set of test inputs T is computed by an automatic test pattern generator or such a test set is already given. Then for all circuits one compacted output is always obtained by a single parity tree, which compacts all outputs of the circuit under test. For the test set T the single stuck-at faults, which are not observable at the parity of all outputs, are determined.

In the first approach additional functional outputs are successively selected so that the faults, which are not observable so far become also detectable for the test set T. The selected single outputs are added to the set of the outputs of the compactor until 100% fault coverage is achieved. The added outputs are not compacted. Column 2 of Table 2.18 lists for this method the number of outputs, which are necessary to obtain 100% fault coverage.

The second approach described in [18] is based on a modification of the original circuit C. Selected internal lines are transformed into new external circuit outputs in addition to the parity of all functional outputs. These lines are successively added to the outputs of the compactor until 100% fault coverage is achieved. Column 3 of Table 2.18 lists the number of outputs, which are necessary for this method.

In the introduction of this chapter a zero-aliasing space compactor design is described, which is proposed for testing by Pouya and Touba [61]. Since the suggested space compactors consist of AND-, OR-, NAND-, or NOR-gates, the compaction is non-linear. The gates are added step by step as follows: After choosing a pair of outputs consisting of functional outputs or already compacted outputs, the type of the gate, which combines these two outputs is chosen. The gate is added to the space compactor, if this results in no additional redundancy. The check, if any redundancy is introduced by the additional gate, involves 5-valued logic and automatic test pattern generation. If all possibilities of output pairs and gate types are checked, but no gate with the demanded property can be found, then the algorithm ends. The experimental results of [61] show for the ten ISCAS '85 benchmarks circuits, that the computations of space compactors for the investigated benchmark circuits require runtimes from less than one minute up to ten hours depending on the size of the processed circuits. Column 4 of Table 2.18 presents the minimal number of compacted outputs, which can be achieved using this approach.

Circuit	[18]		[61]	new approach	
1	2	3	4	5	6
c432	3	4	2	1	1
c499	2	7	1	3	3
c880	1	1	1	2	2
c1355	6	2	1	3	3
c1908	4	2	3	2	3
c2670	9	2	2	3	2
c3540	9	2	2	3	2
c5315	2	35	4	4	3
c6288	5	2	2	2	2
c7552	9	2	5	3	2
average	5.0	5.9	2.3	2.6	2.3

Table 2.18: Results of various compactors for deterministic off-line testing

The compactors proposed in the previous sections can also achieve 100% fault coverage for single stuck-at faults. If the number of compacted outputs is not too small, then for the circuit with an output compactor an automatic test pattern generator can generate a test set, which detects all non-redundant faults of the circuit. For compactors derived from the results of the analysis based on approximated observation probabilities, the minimal numbers of outputs which are required to achieve 100% fault coverage are given in column 5. Column 6 lists for each investigated circuit the minimal number of outputs necessary to detect all testable single stuck-at faults, if the compactors are designed according to the analysis results, which include approximated signal probabilities.

In comparison to the proposed compactor design based on the analysis of the circuit structure, the methods described in [18] on average require more compacted outputs to achieve 100% fault coverage. The methods, which are based on adding functional outputs or selecting internal lines, require on average 5.0 or 5.9 compacted outputs, respectively; while on average 2.6 or 2.3 outputs are sufficient for compactors, which are designed based on approximated observation probabilities or which are derived from the results of the analysis including approximated signal probabilities, respectively.

Although the algorithms proposed in this chapter generate space compactors within less than one second runtime, the compaction ratios of these algorithms are almost as good as the results achieved by the zero-aliasing space compactor design proposed in [61].

In [9] Böhlau introduces a method which generates linear space compactors for functional outputs based on an analysis of the circuit structure, which is different from the algorithms suggested in this work. The basic concept of the analysis of [9] is based on the determination of path lengths in the circuit. Table 2.19 lists the results given in this paper for pseudo random testing with 100000 test vectors. In particular, column 2 lists the numbers of compacted outputs, which are sufficient to detect all single stuck-at faults, that are observable at the functional outputs.

In the previous sections the effect of the proposed compactors on pseudo random testing is investigated, too. The compactors were derived from the analysis based on approximated observation probabilities and from the algorithm including approximated signal probabilities. For comparison, for these compactors columns 4 and 5 of Table 2.19 give the numbers of compacted outputs, which are sufficient to avoid fault masking for the test with 100000 pseudo random vectors. These numbers

Circuit	[9]	new approach	
1	2	3	4
c432	3	1	1
c499	4	3	3
c880	13	2	2
c1355	16	3	3
c1908	6	2	3
c2670	6	3	2
c3540	4	2	2
c5315	5	3	3
c6288	4	2	2
c7552	45	3	2
average	10.8	2.4	2.3

Table 2.19: Results of various compactors for pseudo random testing

of compacted outputs are considerably smaller than the numbers given in column 2. Therefore, the methods introduced above result in a higher compaction ratio compared to the approach of [9], if any reduction of the fault coverage for pseudo random test is not permitted.

2.5 Summary

In this chapter, an approach for the design of linear output space compactors for combinational circuits is discussed. The approach is structural, since it is based on an analysis of the circuit netlist. Several analysis algorithms with polynomial runtime are presented: the analysis algorithms based on common gates ratio or based on propagation distances, the algorithm based on approximations of observation probabilities, a modification of the probability based analysis and, finally, the analysis algorithm, which additionally includes approximated signal probabilities.

The error of the approximations of the proposed analysis algorithms is investigated in this chapter. For each analysis algorithm an example circuit can be constructed in such a way, that the estimation of the reduction of fault propagation caused by two functional outputs assigned to the same parity group is as bad as possible. But it is very unlikely, that such circuits are used in practice.

This is corroborated by comparisons between the exact probabilities and the analysis results for several benchmark circuits. These comparisons are also used to evaluate the accuracy of the various analysis algorithms. The results show, that the analysis algorithms based on approximated probabilities achieve higher accuracies. Therefore, only for these algorithms experimental results are presented. In most cases the best results are obtained for the algorithm based on the approximation of signal and observation probabilities.

Three extensions for the analysis algorithms based on approximated probabilities are introduced: The insertion of complex gates, which replace small circuit parts containing reconvergent paths, the more accurate estimation of signal probabilities, and a direction-dependent fault propagation.

The heuristic algorithm, which computes the parity groups of the compaction function is described in detail. Two modifications of the heuristics are discussed and it is shown, that the total complexity of the proposed design procedure for linear space compactors is at most linear in the product of the circuit size and the square of the number of outputs.

Experimental results are presented for the ten combinational circuits of the ISCAS '85 benchmarks suite [13]. Most of the tests were carried out for the two analysis algorithms, which achieve higher accuracies, i. e. the algorithm based on approximated observation probabilities and the algorithm, which additionally takes into account signal probabilities.

The results for concurrent checking show, that on average less than four compacted outputs are required to achieve an error detection probability of 98%. This is obtained for both fault models, stuck-at and transition faults. For deterministic test generation at most three groups of compacted outputs are necessary for a 100% fault coverage with respect to single stuck-at faults. The size of the test sets produced by an automatic test pattern generator increases at most by one third. Experimental results for pseudo random test are also presented. For each benchmark circuit three compacted outputs are sufficient to detect at the compacted outputs all stuck-at or transition faults, which are observable at the functional outputs.

The experiments indicate, that in many cases better results are obtained by a space compactor design based on approximated signal and observation probabilities than by a compactor derived from the analysis including approximated observation probabilities only. However, very often the results are almost equal.

A comparison between the experimental results of these two analysis algorithms and of a similar approach, which was introduced by Böhlau [9], shows, that the methods described in this chapter produce compactors with a higher compaction ratio which do not reduce the fault coverage for pseudo random test.

3 Structural Output Space Compaction for Sequential Circuits

So far, only output space compaction for combinational circuits has been considered. Now the approach described in the previous chapter is extended to the generation of linear space compactors for sequential circuits.

The next section contains a brief introduction of the application of output space compactors for sequential circuits, since it is very similar to the usage for combinational circuits. As in the previous chapter, the compactor design for sequential circuits is based on parity trees of disjoint groups of outputs. Again, these parity groups are derived from the results of an analysis of the circuit structure. The analysis algorithm is modified, so that fault propagation via the flipflops is additionally taken into consideration. This modification is explained in the second section of this chapter. Then the heuristic algorithm is described, which computes disjoint parity groups based on the results of the analysis. Finally, the last two sections present experimental results and a summary.

3.1 Introduction

The main difference between a combinational and a sequential circuit is, that the sequential circuit contains flipflops, which store the state of the circuit. In the following, techniques for accessing the values stored in the flipflops are not considered. For example, if the flipflops are fully accessible, then only the combinational part of the sequential circuit must be taken into consideration. In this case output space compactors for combinational circuits can be used for testing or concurrent checking as described in the previous chapter.

The design of output space compactors for sequential circuits is optimized for the propagation of single stuck-at faults, and the design procedure is limited to polynomial runtime in terms of the number of inputs, outputs, and gates.

If the outputs of additional observation points of a circuit are treated as functional outputs, then the approach described in the following can be easily extended, and a compactor for the test responses from observation points and functional outputs is derived. The experimental results consider the simulation of single stuck-at faults and transition faults. These fault models are described in the introduction of the previous chapter.

Parts of the work described in this chapter have been published in [77].

State of the Art In the literature, most methods consider output space compaction only for combinational circuits. While for combinational circuits the event, that a certain path is sensitized, depends only on the values applied to the circuit inputs, for sequential circuits it also depends on the state values. For example, for sequential circuits the fault coverage obtained for a given test set depends on the order, which is used for the application of the test patterns, since this order determines the sequence of

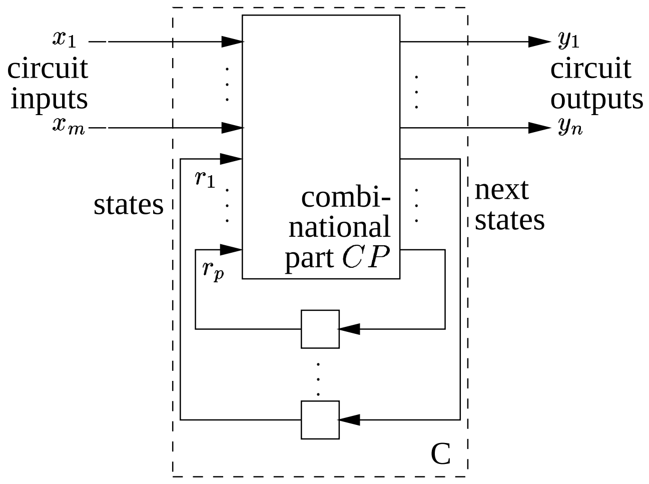


Figure 3.1: Sequential circuit

state values during the test. Therefore, most of the structural approaches, which are proposed for the design of space compactors for combinational circuits, are not suitable for sequential circuits.

In the introduction of the previous chapter a large number of approaches for output space compaction for combinational circuits are mentioned. The approaches, which do not take into consideration the circuit structure can also be used to design space compactors for sequential circuits.

This work investigates for the first time output space compaction for sequential circuits which takes into account fault propagation from faulty states to the outputs of the circuit.

3.2 Analysis of the Circuit Structure

The analysis of the structure of a given circuit requires the knowledge of the netlist of the circuit. The results of the analysis are utilized to determine disjoint parity groups of the linear space compactor. In particular, the analysis results are used to estimate the loss of fault propagation caused by including two functional outputs in the same parity group.

For sequential circuits it must be taken into consideration, that a fault, which is not immediately detectable at the outputs, can produce erroneous states, while in the following clock cycles the error of the states can be propagated to the functional outputs, so that the fault is detected with some delay.

Again, polynomial runtime of the analysis algorithm is demanded, because the design procedure for the linear space compactors should also be applicable to very large circuits.

Since among the analysis algorithms discussed in the previous chapter, the best results were obtained for most combinational circuits using the algorithms based on approximated probabilities, only extensions of these algorithms are described in this chapter.

The same basic notations as in the previous chapter are used. Additionally, the circuit C contains p flipflops $r_1, \dots, r_p$ storing the states of the sequential circuit. The functions implemented by the outputs $y_1, \dots, y_n$ depend on the values of the circuit inputs $x_1, \dots, x_m$ and on the state values stored in $r_1, \dots, r_p$ as shown in Figure 3.1.

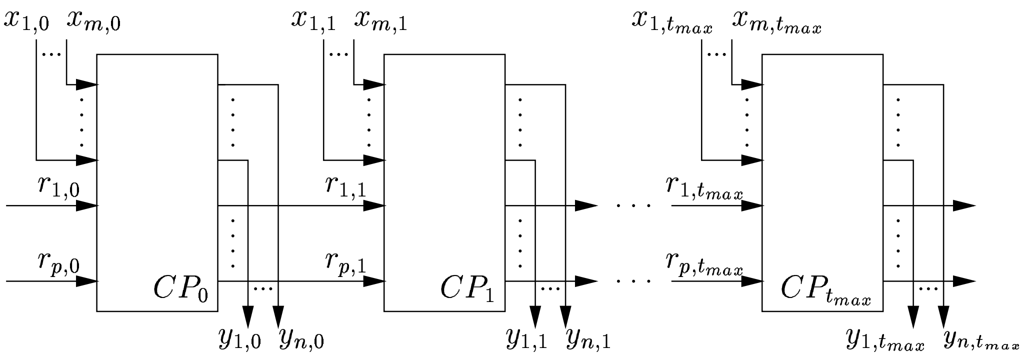


Figure 3.2: Iterative array of time frames of a sequential circuit

3.2.1 Model consisting of Time Frames

A detectable fault $e(s)$ at a line s can be propagated to the circuit outputs directly during the same clock cycle or with a delay of t, $t \geq 1$, clock cycles passing the flipflop elements of the sequential circuit t times. As the delay t increases, the lengths of the paths, which propagate the error are enlarged. Thus, the probability, that one of these paths is sensitized, decreases.

The approximated probability that a path is sensitized decreases exponentially with the number of AND-, NAND-, OR-, or NOR-gates included in this path. Hence the approximated propagation probabilities, that the paths of large delays t are sensitized, are very small compared to the approximated probabilities, which are obtained for direct propagation to the circuit outputs or for small delays t. In order to simplify the analysis, the analysis of the structure of sequential circuits does not take into account paths with delay larger than a convenient upper bound. This bound is denoted by t_{max}.

If the considered delay is limited to t_{max}, then it is sufficient, that the analysis takes into consideration the functional behaviour of the sequential circuit during $t_{max} + 1$ clock cycles. $t_{max} + 1$ clock cycles of a sequential circuit can be modelled as an iterative array $A_C(t_{max}+1)$ of t_{max}+1 time frames $CP_0, CP_1, \ldots, CP_{t_{max}+1}$. This technique is described in [1].

The iterative array $A_C(t_{max} + 1)$ is shown in Figure 3.2. Each time frame CP_t, $0 \leq t \leq t_{max}$, is identical to the combinational part of the sequential circuit C. The inputs and the outputs of the time frame CP_t are denoted by $x_{1,t}, \ldots, x_{m,t}$ and $y_{1,t}, \ldots, y_{n,t}$, respectively. A line s of the original sequential circuit C corresponds to t_{max}+1 lines $s_0, s_1, \ldots, s_{t_{max}}$ of the time frames $CP_0, CP_1, \ldots, CP_{t_{max}}$, respectively. Since the flipflops are removed, $A_C(t_{max} + 1)$ is a combinational circuit.

An error $e(s)$ at the circuit line s, which is propagated with a delay of t clock cycles through the sequential circuit C to the circuit output y_j, is modelled as follows: It corresponds to an error $e(s_0)$ at the corresponding circuit line s_0 of CP_0, which is propagated to the output $y_{j,t}$ of CP_t, $1 \leq t \leq t_{max}$. Hence the probability, that an error $e(s)$ is propagated to the circuit output y_j with a delay of t clock cycles, can be approximated by the value $p(s_0 \rightsquigarrow y_{j,t})$ computed for the iterative array $A_C(t_{max} + 1)$.

3.2.2 Computation of Approximated Probabilities

In the following, the analysis algorithm based on approximated observation probabilities and the analysis, which also includes approximated signal probabilities, are used to compute $p(s_0 \rightsquigarrow y_{j,t})$. In general, this approximated probability can be determined by any analysis algorithm, which is proposed in the previous chapter for combinational circuits and which is based on approximated probabilities. For example, the analysis algorithm based on independent propagation probabilities of reconvergent paths can also be used.

The computation of $p(s_0 \leadsto y_{j,t})$ corresponds to sensitized paths from an erroneous line s_0 of CP_0 through the iterative array $A_C(t_{max}+1)$ to the output $y_{j,t}$ of CP_t. Such a path among a variety of other paths from s_0 to $y_{j,t}$ can go through the erroneous line s_k of CP_k, $1 \leq k \leq t \leq t_{max}$. In this case, the corresponding erroneous line s of the sequential circuit is included twice in the path.

If the probabilities $p(s_0 \leadsto y_{j,t})$ are computed using the analysis algorithm introduced in the previous chapter, then the fact is neglected, that the lines s_k of CP_k, $1 \leq k \leq t$, are erroneous. If a sensitized path from s_0 to $y_{j,t}$ passes at least one of the lines s_k, $1 \leq k \leq t$, that correspond to passing a second time (or even several times) the erroneous line s, then the propagation of the fault of s_0 is masked. But in this case there is also a shorter sensitized path from s_0 to $y_{j,t'}$ of CP'_t, $t' < t$, which does not pass the erroneous lines s_k, $1 \leq k \leq t'$. Because of the shorter length of the path ($t' < t$) the probability $p(s_0 \leadsto y_{j,t'})$ is considerably larger than the probability $p(s_0 \leadsto y_{j,t})$. Therefore, the error of the approximation $p(s_0 \leadsto y_{j,t})$ is neglected below.

The objective of the design of the output compactors, which is based on the analysis of the circuit structure, is to avoid fault masking during on-line checking. For simplification, the possible influence of the initial state of a sequential circuit can be neglected, since after operating for several clock cycles the number of various states, which could be stored in the flipflops, is large for most circuits.

Therefore, the analysis of the structure of a sequential circuit can be carried out as follows: A convenient value t_{max} is chosen and the sequential circuit is modelled as an iterative array yielding $A_C(t_{max}+1)$. The analysis based on approximated observation probabilities or the analysis, which additionally includes approximated signal probabilities are used to compute for each output y_i of the sequential circuit C

$$p_{t_{max}}(s \leadsto y_i) = \sum_{t=0}^{t_{max}} p(s_0 \leadsto y_{i,t}).$$

s_0 corresponds to the line s of the sequential circuit C and is located in the first time frame CP_0 of the array $A_C(t_{max}+1)$. The experimental results show, that for all investigated circuits $t_{max} = 8$ is sufficient, since the changes of the values $p_{t_{max}}(s \leadsto y_i)$ are very small for higher values of t_{max}.

If $|S|$ is the number of lines of $A_C(t_{max}+1)$ and n is the number of outputs of the sequential circuit, then the computation of the values $p_{t_{max}}(s \leadsto y_i)$ can be implemented so that the corresponding complexity is $O(|S| \cdot n^2)$. This bound is valid for both analysis algorithms, i. e. the analysis based on approximated observation probabilities and the analysis, which additionally includes approximated signal probabilities.

The estimation $p_{i,j}$ of the loss of fault propagation produced by the assignment of two circuit outputs y_i and y_j to the same parity group is derived similarly to the methods described in the previous chapter:

$$p_{i,j} = \frac{1}{|S|} \sum_{s \in S} p_{t_{max}}(s \leadsto y_i) \cdot p_{t_{max}}(s \leadsto y_j).$$

The complexity of the algorithms can be determined as follows: The number $|S|$ of lines of the array $A_C(t_{max}+1)$ is at most $(t_{max}+1) \cdot |S_C|$, since the iterative array consists of $t_{max}+1$ time frames $CP_0, \ldots, CP_{t_{max}+1}$ of the combinational part of the underlying sequential circuit C. $|S_C|$ denotes the number of lines of C. The number of the output pairs (y_i, y_j) and thus the number of the corresponding values $p_{i,j}$, which are computed by the analysis, is $(n \cdot (n+1))/2$. Hence the complexity of the analysis algorithms for sequential circuits is $O(|S_C| \cdot (t_{max} + 1) \cdot n^2)$.

Since for $t_{max} > 8$ the experimental results show only small changes for all investigated circuits, a constant value can be chosen for t_{max}. Then the complexity of the proposed analysis algorithms can be described independent of t_{max}, yielding $O(|S_C| \cdot n^2)$.

3.2.3 Modifications and Extensions

Any modification or extension proposed in the previous chapter for the analysis of combinational circuits can also be used for the analysis of the sequential circuits: Small circuit parts of $A_C(t_{max} + 1)$, which contain reconvergent paths can be replaced by complex gates. Furthermore, the analysis can also be based on independent propagation probabilities of reconvergent paths.

The signal probabilities of a sequential circuit can be easily approximated: First $\frac{1}{2}$ is assigned to each circuit input and also to each flipflop. This means that for each circuit input and each flipflop the assignment of the value “0” or the value “1” is equally likely. If the set of reachable states is not uniformly distributed and for each flipflop the independent probability is given, that the flipflop stores “1”, then these probabilities are assigned to the flipflops. Furthermore, the method introduced by Krishnamurthy and Tollis [52] can be used to improve the approximation of signal probabilities.

An extended analysis considering the direction of fault propagation as proposed in the previous chapter can also be applied.

3.3 Generation of Groups of Outputs

The algorithms, which analyze the structure of a sequential circuit as described above, compute for each pair of circuit outputs of C an estimated reduction of fault propagation caused by the assignment of two outputs to the same parity group of the linear space compactor. Therefore, the analysis algorithms of sequential circuits compute the same type of results as the algorithms given in the previous chapter for combinational circuits. Thus, the grouping algorithm and its modifications, which are described for combinational circuits, can also be used for the design of output compactors for sequential circuits.

Again, for a given number of compacted outputs k the heuristic algorithms compute disjoint groups of outputs from the results of the analysis algorithms, so that the sum $P(\mathcal{G}_k)$ of the resulting partition $\mathcal{G}_k$ is minimized.

3.4 Experimental Results

The experimental results are derived for 20 sequential circuits of the ISCAS ’89 benchmarks suite [12]. The circuits are shown in Table 3.1. They contain 3 to 78 inputs, 5 to 152 outputs, and 5 to 638 flipflops.

The structure of each sequential benchmark circuit was modelled as an iterative array $A_C(9)$. This implies, that t_{max} was set to 8. As mentioned above, experimental results show, that $t_{max} = 8$ is sufficient, since higher values of t_{max} change the approximated probabilities $p_{t_{max}}(s \rightsquigarrow y)$ only slightly.

Each iterative array derived from a sequential benchmark circuit was analyzed using the algorithm based on approximated observation probabilities and the analysis including approximated signal prob-

Sequential	Number of			
Circuit	gates	inputs	outputs	flipflops
s298	119	3	6	14
s344	160	9	11	15
s349	161	9	11	15
s382	158	3	6	21
s386	159	7	7	6
s400	162	3	6	21
s444	181	3	6	21
s510	211	19	7	6
s526	193	3	6	21
s641	379	35	23	19
s713	393	35	23	19
s820	289	18	19	5
s832	287	18	19	5
s1196	529	14	14	18
s1423	657	17	5	74
s1494	647	8	19	6
s5378	2779	35	49	164
s9234.1	5597	36	39	211
s13207.1	8020	67	152	638
s15850.1	9785	78	150	534
average	1543	21	29	92

Table 3.1: ISCAS '89 benchmark circuits

abilities. The analysis algorithms were implemented in C++. It took a computer with an AMD Athlon processor running at 700 MHz under Linux less than 5 seconds to analyze any iterative array. The runtime of the analysis was measured without the time required for reading the netlist of the processed circuit. The experimental results are given in Table 3.2.

The heuristic algorithm, which was introduced for the design of output compactors for combinational circuits, was used to determine disjoint groups of outputs. Again, the algorithm computes a partition $\mathcal{G}_k$ so that the corresponding sum $P(\mathcal{G}_k)$ is minimal. The parities of these groups are the compacted outputs of the sequential circuit.

The experimental results involving fault simulation of stuck-at and transition faults were obtained using the simulator `Minos`.

3.4.1 Simulation of Stuck-At Faults

In order to investigate how good the proposed design of linear space compactors for sequential circuits deals with the issue of error masking, sequential circuits with no output compaction and with compacted outputs were simulated. The output space compactors for the sequential circuits were derived from the analysis based on approximated observation probabilities and the analysis, which includes additionally the signal probabilities. The experiments included simulation of single stuck-at faults.

Sequential Circuit	Runtime of analysis algorithms based on approx. observ. probabilities (s)	approx. signal & observ. probabilities (s)
s298	0.01	<0.01
s344	<0.01	0.01
s349	<0.01	<0.01
s382	<0.01	0.01
s386	0.01	<0.01
s400	0.01	0.01
s444	<0.01	<0.01
s510	<0.01	0.01
s526	<0.01	<0.01
s641	0.02	0.03
s713	0.02	0.02
s820	0.01	0.03
s832	0.01	0.04
s1196	0.01	0.01
s1423	0.01	0.01
s1494	0.03	0.05
s5378	0.48	0.63
s9234.1	0.53	0.60
s13207.1	3.56	4.30
s15850.1	3.52	3.98

Table 3.2: Runtimes required for structure analysis using different algorithms

The testability of sequential circuits with the proposed linear space compactors is investigated for concurrent on-line checking and for testing based on pseudo random input vectors. In the following, an original circuit checked or tested without output compaction is denoted by C and circuits with k compacted outputs, which are derived from an output partition $\mathcal{G}_k$, are denoted by C_k.

3.4.1.1 On-Line Checking

The comparison of the on-line testabilities of circuits without and with the proposed linear space compactors was carried out as in the previous chapter: First a sequence of 100000 pseudo random input vectors were applied to the circuit inputs of the original circuit $C(e(s))$, which contained a single stuck-at fault $e(s)$ at the circuit line s. Then the same vectors were applied in the presence of $e(s)$ to the faulty circuit $C_k(e(s))$ with k compacted outputs. If $n(e(s))$ and $n_k(e(s))$ are the numbers of pseudo random input vectors, for which at least one of the checked outputs of $C(e(s))$ and $C_k(e(s))$ is erroneous, then

$$\mu_{k,e(s)} = \frac{n(e(s)) - n_k(e(s))}{n(e(s))}$$

Sequential Circuit	Reduction of on-line testability for SSFs (%)							
	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
s298	45.95	12.27	14.12	4.52	<0.01	0	–	–
s344	25.81	8.33	2.77	1.00	0.47	0.07	0.64	0.21
s349	25.82	8.39	2.81	1.00	0.48	0.07	0.64	0.21
s382	7.87	2.85	2.85	<0.01	<0.01	0	–	–
s386	11.64	6.93	0.04	<0.01	<0.01	<0.01	0	–
s400	7.89	7.43	3.24	1.04	<0.01	0	–	–
s444	5.65	5.22	3.13	0.68	<0.01	0	–	–
s510	34.29	17.43	5.86	3.44	5.01	1.94	0	–
s526	4.09	0.24	2.75	<0.01	<0.01	0	–	–
s641	11.04	2.61	1.14	0.82	0.29	0.14	0.13	0.10
s713	10.53	2.59	1.07	0.62	0.18	0.15	0.31	0.10
s820	18.37	5.04	0.72	1.29	0.78	0.30	0.54	0.36
s832	18.16	4.93	0.71	1.27	0.77	0.28	0.53	0.36
s1196	15.56	6.46	2.98	1.42	1.03	0.53	0.25	0.21
s1423	3.13	0.17	0.01	<0.01	0	–	–	–
s1494	14.32	7.16	2.13	0.68	0.57	0.48	0.95	0.08
s5378	11.42	2.09	0.79	0.33	0.16	0.07	0.04	0.03
s9234.1	10.81	3.30	2.71	1.76	2.13	0.09	<0.01	1.42
s13207.1	18.19	1.63	1.15	0.51	0.26	0.35	0.05	0.01
s15850.1	12.25	5.22	3.58	0.41	0.20	0.10	0.05	0.03
average	15.64	5.51	2.73	1.04	0.62	0.23	0.21	0.16

Table 3.3: On-line testability of sequential circuits with output compactors derived from approximated observation probabilities

estimates the probability, that an error at the functional outputs caused by the fault $e(s)$ is not observable at the k compacted outputs.

Again, the Equation (2.5) given on page 50 is used: If E_C is the set of all single stuck-at faults of the circuit C, then

$$\mu_k = \frac{1}{|E_C|} \cdot \sum_{e(s)\in E_C} \mu_{k,e(s)}$$

can be used to compute the expected value of the probability that a pseudorandom input vector makes an arbitrarily chosen single stuck-at fault detectable at the functional outputs, but not at the compacted outputs of C_k.

Analysis Based on Observation Probabilities Table 3.3 gives the experimentally determined values μ_k, $k = 1, \ldots, 8$, for circuits with k compacted outputs derived from the analysis based on approximated observation probabilities.

For each benchmark circuit a compactor with $k = 6$ outputs is sufficient to achieve $\mu_k < 2\%$. On average, $k = 3.6$ compacted outputs are necessary to meet this requirement.

Sequential Circuit	Reduction of on-line testability for SSFs (%)							
	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
s298	45.95	35.41	8.21	0.56	<0.01	0	–	–
s344	25.81	8.44	5.88	0.85	2.46	2.02	0.18	<0.01
s349	25.82	8.50	5.86	0.86	2.43	2.00	0.17	<0.01
s382	7.87	7.19	1.09	3.25	<0.01	0	–	–
s386	11.64	2.14	1.25	0.33	0.01	<0.01	0	–
s400	7.89	6.39	<0.01	<0.01	<0.01	0	–	–
s444	5.65	4.54	<0.01	<0.01	<0.01	0	–	–
s510	34.29	15.10	4.60	3.06	0.02	<0.01	0	–
s526	4.09	3.84	0.24	<0.01	<0.01	0	–	–
s641	11.04	4.72	3.68	0.35	0.13	0.09	0.07	0.05
s713	10.53	3.43	2.47	0.29	0.17	0.23	0.10	0.07
s820	18.37	5.09	5.10	1.44	2.53	<0.01	<0.01	0.07
s832	18.16	4.97	1.08	1.81	2.47	<0.01	<0.01	0.07
s1196	15.56	6.31	3.63	1.07	0.96	0.67	0.02	0.01
s1423	3.13	0.03	0.08	<0.01	0	–	–	–
s1494	14.32	8.33	3.62	2.68	0.67	1.32	0.18	0.03
s5378	11.42	1.42	1.19	0.69	0.32	0.16	0.10	0.06
s9234.1	10.81	3.23	0.63	2.08	1.10	0.26	0.24	0.09
s13207.1	18.19	1.48	1.79	0.59	0.38	0.33	0.10	0.02
s15850.1	12.25	2.38	1.33	0.16	0.18	0.12	0.07	0.35
average	15.64	6.65	2.59	1.00	0.69	0.36	0.06	0.04

Table 3.4: On-line testability of sequential circuits with output compactors derived from approximated signal and observation probabilities

A comparison between μ_1 and μ_2 yields the following: If two parity groups instead of a single parity tree are selected, then the probability, that a fault is observable at the functional outputs, but not at the compacted outputs, is considerably reduced. The reduction of the masking probability is on average 65% for the twenty circuits. For sixteen out of the twenty circuits μ_2 is equal or less than μ_1 divided by 2.

The results obtained for the compactors, which are generated based on approximated observation probabilities show for most circuits, that the on-line testability can get worse, although the number of compacted outputs is increased. For example, for s526 it is $\mu_2 = 0.24$ for 2 compacted outputs, but $\mu_3 = 2.75$ for one more compacted output. This observation corresponds to similar results derived for combinational circuits and can be explained in the same way as in the previous chapter.

Analysis Based on Approximated Signal and Observation Probabilities Table 3.4 gives the values of μ_k, which are obtained by simulating the original circuit C and the circuits C_k with $k = 1, \ldots, 8$ compacted outputs. The compactors were generated according to the results of the analysis based on approximated signal and observation probabilities.

Compactors with $k \leq 5$ outputs are sufficient to achieve $\mu_k < 2\%$ for all benchmark circuits.

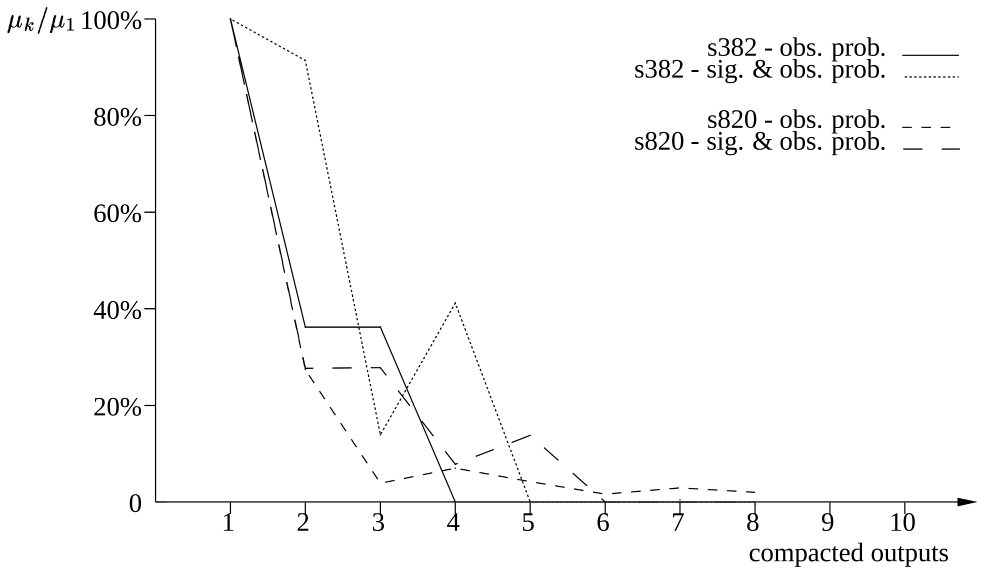


Figure 3.3: Comparison of results for on-line checking

On average $k = 3.4$ compacted outputs are necessary to fulfill this condition, if the generation of the compactors is based on approximated signal and observation probabilities. If the underlying analysis does not approximate signal probabilities, then the average number of compacted outputs is slightly higher ($k = 3.6$).

Similar to the values listed in the Table 3.3, the results for compactors which are generated based on approximated signal and observation probabilities show for many circuits, that the on-line testability of circuits with compacted outputs can decrease, although the number of compacted outputs is increased.

Comparison of the Results Obtained from Different Analysis Algorithms The experimental results of compactors derived from the two different analysis algorithms are given in Table 3.3 and Table 3.4: The results of Table 3.3 correspond to the analysis based on approximated observation probabilities and the results of the Table 3.4 are related to the analysis including approximated signal probabilities. A comparison of the two tables shows, that for many circuits the results obtained from the two analysis algorithms are varying considerably. This is shown more clearly in Figure 3.3.

Figure 3.3 shows the reduction of μ_k, $k = 1, \ldots, 8$, for two circuits s382 and s820 with compactors based on the two different analysis algorithms. The graphs are computed in the following way: μ_1, which is the value obtained for a single parity tree of all circuit outputs, is set to 100%. For $k = 2, \ldots, 8$ the corresponding percentages of μ_k/μ_1 are determined and shown in Figure 3.3 for the four combinations: The solid line and the line of long dashes show the results obtained for the sequential circuits s382 and s820 with compacted outputs, respectively. The compactors were derived from the analysis based on approximated observation probabilities. The lines consisting of dots and short dashes show also results obtained for s382 and s820 with compacted outputs, but the compactors were generated according to the analysis results, which include approximated signal probabilities.

The two graphs of s382, which both include parts with positive gradients, differ considerably for less than 5 compacted outputs. For the sequential circuit s820 the analysis based on approximated observation probabilities achieves lower μ_k for $k = 3, 4, 5$ compacted outputs. In contrast, the results of the other analysis algorithm are better for $k = 6, 7, 8$.

The varying results indicate, that for several sequential circuits and for various k both analysis algorithms are not accurate enough, so that the resulting compactors mask too many faults. In contrast, for almost all combinational circuits the two analysis algorithms result in compactors, which minimize fault masking for any k with only small differences.

The distinction between combinational and sequential circuits is, that sequential circuits contain flipflops, which store the states. Often, sequential circuits reach only a limited number of states or some states are less likely. In this case, the propagation probability is 0 or very low for paths, which become only sensitized under such states. The two analysis algorithms proposed above neglect this effect.

To further improve the analysis the approximation of signal probabilities can take into account the set of reachable states as described in Section 3.2.3. Also the other modifications, which are proposed for the analysis algorithms in Section 3.2.3, might be useful.

3.4.1.2 Pseudo Random Testing

In this section, the effect of the proposed compactors on pseudo random testing is investigated. This is done by fault simulation: 100000 pseudo random input vectors are applied to the original circuit C and to the circuit C_k corresponding to the output partition $\mathcal{G}_k$. The fault simulator determines d and d_k which are the numbers of detected stuck-at faults of C and C_k, respectively. For the simulation of C_k only the stuck-at faults of the original circuit C are considered. Additional faults within the XOR-trees of the compactors are omitted.

The quotient

$$q_k = \frac{d - d_k}{d}$$

determines the fraction of faults, which are observable at the functional outputs of the sequential circuit, but which are not detected at the compacted circuit outputs of C_k.

Analysis Based on Observation Probabilities The analysis based on approximated observation probabilities were used to determine compactors for the benchmarks circuits. The experimental results, which are obtained after applying 100000 pseudo random test vectors to the circuits with compacted outputs, are presented in Table 3.5. Each row gives the name of the investigated circuit, the number of undetected single stuck-at faults of $C, C_1, \ldots, C_4$, and the value of q_k, $k = 1, \ldots, 4$.

For several sequential circuits the pseudo random test achieves a poor fault coverage so that the pseudo random test is not sufficient for these circuits. The circuits s382, s400, s444, s526, and s9234.1 are not listed in Table 3.5, because for each of these circuits the fault coverage obtained by fault simulation is less than 40%.

The results show, that for pseudo random test a single parity tree compacting all functional outputs to one compacted output causes fault masking for most circuits. If compactors with two outputs are added to the circuits, then fault masking is avoided for nine of the fifteen circuits. Finally, the space compactors of 13 circuits require at most four outputs, so that all faults which are observable at the functional outputs are detected at the compacted outputs. To fulfill also this condition for the circuits s5378 and s13207.1, five compacted outputs are required.

Sequential	Number of undetected SSFs					Ratio $q_k = (d - d_k)/d$ (%)			
Circuit	C	C_1	C_2	C_3	C_4	$k = 1$	$k = 2$	$k = 3$	$k = 4$
s298	64	265	66	64	65	49.81	0.38	0	0.19
s344	18	18	18	18	18	0	0	0	0
s349	22	22	22	22	22	0	0	0	0
s386	192	193	193	192	192	0.17	0.17	0	0
s510	0	0	0	0	0	0	0	0	0
s641	152	152	152	152	152	0	0	0	0
s713	230	230	230	230	230	0	0	0	0
s820	829	849	829	829	829	2.47	0	0	0
s832	852	873	852	852	852	2.59	0	0	0
s1196	18	21	18	18	18	0.13	0	0	0
s1423	977	977	977	977	977	0	0	0	0
s1494	1049	1152	1112	1060	1049	5.31	3.25	0.57	0
s5378	3020	3447	3069	3024	3027	5.70	0.65	0.05	0.09
s13207.1	15573	18529	15576	15595	15576	27.04	0.03	0.20	0.03
s15850.1	16917	17711	16935	16950	16917	5.36	0.12	0.22	0

Table 3.5: Pseudo random test of sequential circuits with output compactors derived from approximated observation probabilities

Analysis Based on Approximated Signal and Observation Probabilities In the following, output compactors for sequential circuits are considered, which are obtained from the results of the analysis based on approximated signal and observation probabilities. Again, 100000 pseudo random test vectors were applied to the sequential circuits and fault simulation was used to determine, how many single stuck-at faults remain undetected. Table 3.6 lists the numbers of undetected faults for the original circuit C and for the circuits $C_1, \ldots, C_4$ with $k = 1, \ldots, 4$ compacted outputs. For each circuit $C_1, \ldots, C_4$ the loss of fault coverage q_k, which was computed as described above, is given in the table. Again, if for any original circuit the fault coverage, which was obtained by fault simulation, is less than 40%, then this circuit is not listed in Table 3.6.

If compactors with two outputs are used, then the fault masking occurs only for a fifth of the listed circuits. For almost all circuits the fault coverage is not reduced for compactors with 4 or less outputs. The only exception is s1494, which requires 5 compacted outputs.

Comparison of the Results Obtained from Different Analysis Algorithms The usage of the compactors, which are derived from the two different analysis algorithms, yields varying results for a large number of the investigated sequential circuits.

For the analysis, which is only based on approximated observation probabilities, on average 2.4 compacted outputs are sufficient to avoid fault masking. In comparison, the compactors derived from the analysis algorithm including approximated signal probabilities require on average a slightly lower number of compacted outputs (2.1). This result indicates again, that for most circuits the approximation of signal probabilities increases the accuracy of the approximated probabilities.

Sequential Circuit	Number of undetected SSFs					Ratio $q_k = (d - d_k)/d$ (%)			
	C	C_1	C_2	C_3	C_4	$k=1$	$k=2$	$k=3$	$k=4$
s298	64	265	71	66	65	49.81	1.32	0.19	0
s344	18	18	18	18	18	0	0	0	0
s349	22	22	22	22	22	0	0	0	0
s386	192	193	192	192	192	0.17	0	0	0
s510	0	0	0	0	0	0	0	0	0
s641	152	152	152	152	152	0	0	0	0
s713	230	230	230	230	230	0	0	0	0
s820	829	849	829	829	829	2.47	0	0	0
s832	852	873	852	852	852	2.59	0	0	0
s1196	18	21	18	18	18	0.13	0	0	0
s1423	977	977	977	977	977	0	0	0	0
s1494	1049	1152	1098	1075	1056	5.31	2.53	1.34	0.36
s5378	3020	3447	3020	3027	3031	5.70	0	0.09	0.15
s13207.1	15573	18529	15573	15626	15573	27.04	0	0.48	0
s15850.1	16917	17711	16984	16932	16917	5.36	0.45	0.10	0

Table 3.6: Pseudo random test of sequential circuits with output compactors derived from approximated signal and observation probabilities

3.4.2 Simulation of Transition Faults

In this section, the effect of the proposed compactors on the propagation of transition faults is investigated. Although the design of the compactors was optimized for the propagation of single stuck-at faults, the compactors should also propagate output errors caused by defects, which are not modelled by single stuck-at faults.

The underlying analysis algorithms considered in the following are the analysis based on approximated observation probabilities and the analysis, which additionally includes signal probabilities. The testability of circuits with compactors, which are generated according to the results of the two analysis algorithms, is investigated for concurrent on-line checking and for testing based on pseudo random input vectors.

3.4.2.1 On-Line Checking

The experiments investigating the on-line testability with respect to transition faults were carried out similar to the experiments for the single stuck-at fault model. A sequence of 100000 pseudo random input vectors was applied to circuits without and with the proposed linear space compaction, in order to determine μ_k as given in Equation (2.5) on page 50 for a circuit C_k with k compacted outputs: If E_C is the set of all transition faults of C, and $n(e(s))$ $(n_k(e(s)))$ is the number of pseudo random inputs for which under the presence of the fault $e(s) \in E_C$ at least one of the outputs of C (C_k) is erroneous, then it is

$$\mu_k = \frac{1}{|E_C|} \cdot \sum_{e(s) \in E_C} \frac{n(e(s)) - n_k(e(s))}{n(e(s))}.$$

Sequential Circuit	Reduction of on-line testability for transition faults (%)							
	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
s298	37.79	9.16	12.77	2.88	<0.01	0	0	–
s344	18.72	5.76	2.13	0.70	0.43	0.05	0.52	0.18
s349	18.81	5.86	2.16	0.71	0.44	0.05	0.52	0.18
s382	0.52	<0.01	<0.01	<0.01	<0.01	0	–	–
s386	9.17	5.13	0.01	<0.01	<0.01	<0.01	0	–
s400	0.50	0.50	<0.01	<0.01	<0.01	0	–	–
s444	0.34	0.34	<0.01	<0.01	<0.01	0	–	–
s510	30.23	15.38	4.89	3.25	4.25	1.90	0	–
s526	0.29	<0.01	<0.01	<0.01	<0.01	0	–	–
s641	8.95	1.71	0.74	0.59	0.23	0.10	0.10	0.06
s713	8.35	1.59	0.72	0.55	0.13	0.11	0.28	0.08
s820	14.28	4.17	0.64	1.12	0.75	0.26	0.42	0.29
s832	14.16	4.08	0.63	1.10	0.73	0.26	0.41	0.28
s1196	15.23	6.30	2.91	1.33	1.16	0.54	0.26	0.22
s1423	2.73	0.13	0.01	<0.01	0	–	–	–
s1494	11.48	5.81	1.81	0.53	0.46	0.40	0.78	0.07
s5378	8.84	1.44	0.64	0.24	0.15	0.12	0.04	0.03
s9234.1	5.48	1.61	1.48	1.20	1.36	<0.01	<0.01	1.15
s13207.1	9.80	0.90	0.77	0.30	0.09	0.09	0.01	<0.01
s15850.1	7.46	3.67	2.65	0.29	0.19	0.06	0.01	0.01
average	11.16	3.68	1.75	0.74	0.52	0.20	0.17	0.13

Table 3.7: On-line testability of sequential circuits with output compactors derived from approximated observation probabilities

Analysis Based on Observation Probabilities Table 3.7 lists the experimental results obtained for circuits with $k = 1, \dots, 8$ compacted outputs.

The fault coverages achieved for transition faults are considerably smaller than the fault coverages obtained from the experiments, which simulated single stuck-at faults. For the circuits s382, s400, s444, and s526 the fault coverages are below 10%.

The low fault coverages can be explained due to the fact, that a test of a single delay fault consists of two successive vectors. The first vector initializes the faulty site to a given value and the second vector launches the transition at the faulty site, so that the delay fault is propagated to the flipflops or to the circuit outputs. If the number of different pairs of two vectors, which are applied successively to the circuit is low, then usually a low fault coverage is obtained [70].

For sequential circuits the test vectors applied to the combinational part consists of the values assigned to the inputs and the state. For several sequential circuits the sequence of states produces only a small number of different pairs of two successive states yielding a low fault coverage for transition faults. In order to achieve higher fault coverages, an access mechanism to the state values can be added so that the states can be controlled during the test.

The low fault coverage implies, that mostly faults that are easily detectable are observed during the simulation. For these faults the approximated probabilities of propagation to the outputs are very

Sequential Circuit	Reduction of on-line testability for transition faults (%)							
	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
s298	37.79	29.37	6.82	0.47	<0.01	0	–	–
s344	18.72	5.55	3.56	0.62	1.94	1.08	0.08	<0.01
s349	18.81	5.64	3.57	0.63	1.92	1.07	0.08	<0.01
s382	0.52	0.52	<0.01	0.52	<0.01	0	–	–
s386	9.17	1.48	0.99	0.26	0.01	<0.01	0	–
s400	0.50	0.50	<0.01	<0.01	<0.01	0	–	–
s444	0.34	0.34	<0.01	<0.01	<0.01	0	–	–
s510	30.23	13.34	4.17	2.06	<0.01	<0.01	0	–
s526	0.29	0.29	<0.01	<0.01	<0.01	0	–	–
s641	8.95	3.87	3.27	0.26	0.10	0.06	0.05	0.03
s713	8.35	2.63	2.17	0.21	0.13	0.21	0.08	0.07
s820	14.28	4.05	4.48	1.24	2.21	<0.01	<0.01	0.07
s832	14.16	3.96	0.98	1.59	2.16	<0.01	<0.01	0.06
s1196	15.23	6.07	3.57	1.00	0.94	0.63	0.02	0.01
s1423	2.73	0.02	0.07	<0.01	0	–	–	–
s1494	11.48	6.50	2.98	2.38	0.60	1.09	0.10	0.05
s5378	8.84	1.17	0.97	0.44	0.29	0.14	0.09	0.05
s9234.1	5.48	1.61	0.18	1.35	0.35	0.06	0.07	<0.01
s13207.1	9.80	0.78	1.28	0.41	0.31	0.14	0.05	<0.01
s15850.1	7.46	1.45	0.84	0.09	0.14	0.07	0.02	0.30
average	11.16	4.41	2.00	0.68	0.56	0.23	0.03	0.04

Table 3.8: On-line testability of sequential circuits with output compactors derived from approximated signal and observation probabilities

high. Thus, the proposed algorithms form parity groups in a way, that makes it very unlikely, that one of these faults is propagated to two outputs of the same group. So for each circuit and almost every k, the values μ_k listed in Table 3.7 are smaller in comparison to the results given in Table 3.3 for single stuck-at faults.

For each investigated benchmark circuit a compactor with $k = 6$ outputs is sufficient to achieve $\mu_k < 2\%$, and on average $k = 2.9$ compacted outputs are necessary, to fulfill this condition.

Analysis Based on Approximated Signal and Observation Probabilities For circuits with k compacted outputs the values μ_k, $k = 1, \dots, 8$, were experimentally determined for transition faults. The values are listed in Table 3.8. The compactors were derived from the results of the analysis based on approximated signal and observation probabilities.

The comparison between the simulation of transition faults and single stuck-at faults yields the same result as above: The fault coverages achieved for transition faults are considerably smaller than the fault coverages obtained from the simulation of single stuck-at faults. Again, for the circuits s382, s400, s444, and s526 the fault coverages are below 10%. And for each circuit and almost every k, the values μ_k listed in Table 3.8 are smaller in comparison to the results given in Table 3.4 for single

Sequential	Number of undet. transition faults					Ratio $q_k = (d - d_k)/d$ (%)			
Circuit	C	C_1	C_2	C_3	C_4	$k = 1$	$k = 2$	$k = 3$	$k = 4$
s298	167	360	167	167	167	51.28	0	0	0
s344	64	66	64	64	64	0.33	0	0	0
s349	69	71	69	69	69	0.33	0	0	0
s386	336	341	343	336	336	1.15	0.69	0	0
s510	129	132	129	129	129	0.34	0	0	0
s641	244	244	244	244	244	0	0	0	0
s713	353	353	353	353	353	0	0	0	0
s1196	56	63	56	56	56	0.30	0	0	0
s1423	1421	1421	1421	1421	1421	0	0	0	0
s1494	1364	1454	1413	1374	1364	5.54	3.02	0.62	0
s5378	3920	4245	3944	3928	3920	4.93	0.36	0.12	0

Table 3.9: Pseudo random test of sequential circuits with output compactors derived from approximated observation probabilities

stuck-at faults.

On average, $k = 2.9$ compacted outputs are sufficient to achieve $\mu_k < 2\%$.

Comparison of the Results Obtained from Different Analysis Algorithms The comparison of the two Tables 3.7 and 3.8 shows, that for most circuits the results obtained from the two analysis algorithms based on approximated probabilities are considerably varying. The reasons for the different results are the same as those given above for the differences obtained for stuck-at faults.

3.4.2.2 Pseudo Random Testing

In this section, the effect of the proposed compactors on pseudo random testing of transition faults is investigated. 100000 pseudo random test input vectors are applied to the original circuit C and to the circuit C_k, in order to determine d and d_k, which are the numbers of detected faults of C and C_k, respectively. Then the quotient $q_k = \frac{d-d_k}{d}$ is computed.

Analysis Based on Observation Probabilities The analysis based on approximated observation probabilities was used to determine compactors for the investigated circuits. The experimental results, which were obtained from the simulation of the circuits with these compactors, are presented in Table 3.9.

For several sequential circuits the pseudo random test achieves poor fault coverages. The explanation is the same as given in the previous section given for the low fault coverages obtained for on-line checking. Sequential circuits with fault coverage less than 40% are not listed in Table 3.9.

While one compacted output consisting of a single parity tree reduces the fault coverage for eight out of eleven circuits, two compacted outputs cause fault masking only for three circuits. Finally, a compactor with four outputs is sufficient to avoid fault masking for each listed circuit.

Sequential	Number of undet. transition faults					Ratio $q_k = (d - d_k)/d$ (%)			
Circuit	C	C_1	C_2	C_3	C_4	$k = 1$	$k = 2$	$k = 3$	$k = 4$
s298	167	387	180	167	167	51.28	3.03	0	0
s344	64	66	64	64	64	0.33	0	0	0
s349	69	71	69	69	69	0.33	0	0	0
s386	336	341	338	338	336	1.15	0.46	0.46	0
s510	129	132	130	130	131	0.34	0.11	0.11	0.22
s641	244	244	244	244	244	0	0	0	0
s713	353	353	353	353	353	0	0	0	0
s1196	56	63	56	56	56	0.30	0	0	0
s1423	1421	1421	1421	1421	1421	0	0	0	0
s1494	1364	1454	1401	1385	1374	5.54	2.28	1.29	0.62
s5378	3920	4245	3927	3925	3924	4.93	0.11	0.08	0.09

Table 3.10: Pseudo random test of sequential circuits with output compactors derived from approximated signal and observation probabilities

Analysis Based on Approximated Signal and Observation Probabilities In the following, output compactors for sequential circuits are considered, which are obtained from the results of the analysis based on approximated signal and observation probabilities. Again, 100000 pseudo random test vectors were applied to the circuits with these compactors, in order to simulate the propagation of transition faults.

Table 3.10 lists the numbers of undetected faults for the original circuit C and for the circuits $C_1, \ldots, C_4$ and also the loss of fault coverage q_k with $k = 1, \ldots, 4$.

Two compacted outputs are required to achieve no fault masking for six out of eleven circuits. A compactor with five outputs is sufficient to avoid fault masking for each listed circuit.

Comparison of the Results Obtained from Different Analysis Algorithms Compactors derived from the analysis based on approximated observation probabilities only achieve better results than compactors, which were generated including approximated signal probabilities. On average 2.2 compacted outputs are sufficient to avoid fault masking, if the former analysis is used. A compactor generated according to the results of the latter analysis algorithm requires on average 2.8 compacted outputs to fulfill this condition.

3.5 Summary

In this chapter the design of linear output space compaction, which is proposed for combinational circuits in the previous chapter, is extended to the application to sequential circuits. Again, the approach is based on an analysis of the circuit netlist. Only the extensions of two analysis algorithms based on approximated probabilities are described in this chapter, since for the most combinational circuits the compactors derived from these two algorithms achieve the best results.

The basic idea of the structural analysis of a sequential circuit is to model the sequential circuit as an iterative array of time frames, which consist of the combinational part of the sequential circuit. Then the analysis algorithms proposed for combinational circuits can be used to process the iterative array.

The groups of outputs, which determine the function of the linear compactor, are obtained in the same way as described in the previous chapter for combinational circuits. The proposed approach is applicable to large sequential circuits, since the complexity of the design procedure is linear in the product of the circuit size and the square of the number of outputs.

Experimental results are given for twenty sequential circuits of the ISCAS '89 benchmarks suite [12]. The results for concurrent checking show, that on average four compacted outputs are enough to achieve an error detection probability of 98% for stuck-at and transition faults. For pseudo random test, compactors with five outputs at most are sufficient to detect all errors at the compacted outputs, which are produced by stuck-at or transition faults and which are observable at the functional outputs.

If two different compactors are designed for the same sequential circuit using the two different analysis algorithms based on approximated probabilities, then in many cases the reduction of fault propagation of these two compactors differs considerably. This indicates, that the approximations of the probabilities, which are computed by the analysis algorithms are inaccurate, so that the resulting compactors may mask too many faults. In contrast, for combinational circuits the experimental results differ only slightly for the two analysis algorithms.

Obviously, the reason for the varying accuracy of the analysis algorithms for combinational and sequential circuits is, that a sequential circuit contains flipflops storing the state. For example, the analysis algorithms neglect the effect, that for many sequential circuits several states are not reachable and then certain paths can never be sensitized. To handle these issues, some modifications of the analysis algorithms are proposed in this chapter. These modifications can improve the design of linear output space compactors for sequential circuits.

4 Output Space Compaction for Circuits with Unknown Structures

In the previous chapters, the design of output space compactors for a given circuit required the knowledge of the circuit netlist. The linear space compactors are designed so that the propagation of errors at the functional outputs is optimized for stuck-at faults. In this chapter it is assumed, that the structure of the circuit is unknown, but a precomputed test set and the corresponding fault-free test responses of the circuit are given. Under this condition, non-linear space compactors are designed without knowledge of the underlying fault model of the test set.

The following section introduces the basic concept of space compactors for circuits, whose structure is not known, and discusses the usage of these compactors. In the following sections the basic design of such compactors and an extension, which improves the compaction ratio, are described. Afterwards experimental results are presented. Finally, the last section gives a summary.

4.1 Introduction

Circuits or modules, whose structures are unknown, are often used for large chip designs. The design of a chip starts with the design of the architecture, which is developed according to the specifications of the chip. The chip is divided into several modules implementing various functionalities. These modules are designed separately or some pre-designed embedded cores are used. The advantage of including pre-designed modules is, that the demanded functionality can be added to the chip without expensive and time consuming development. Therefore, design reuse allows greater on-chip functionality and shorter product development cycles.

In many cases, such pre-designed functional blocks are intellectual property (IP) cores. Normally, the core vendor gives no structural information about the IP core, so that the intellectual property is protected. In this case, the methods proposed in the previous chapters can not be used to design an output space compactor. However, if the vendor wants to support the test of the IP core and provides a set of test patterns with the corresponding fault-free test responses of the IP core, then the space compactor design introduced in this chapter can be applied.

Testing a core embedded in a chip poses the problem of propagating the test responses of the core to observable chip input/output pins. If the test responses of the core are inputs of other circuits, then errors at the outputs of the core may be masked, so that they can not be observed. As shown in Figure 4.1, wrapper cells can be placed on the functional outputs [86]. Since wrapper cells serially shift the test responses to observable chip input/output pins, the test application time increases considerably. Furthermore, this approach causes delays on functional paths between cores.

In order to reduce the delays on functional paths, space compactors can be used to access the test responses of the core. Since the access is in parallel, the test time increases only slightly and at-speed testing is facilitated. This approach is shown in Figure 4.2 for a sequential circuit in full-scan design.

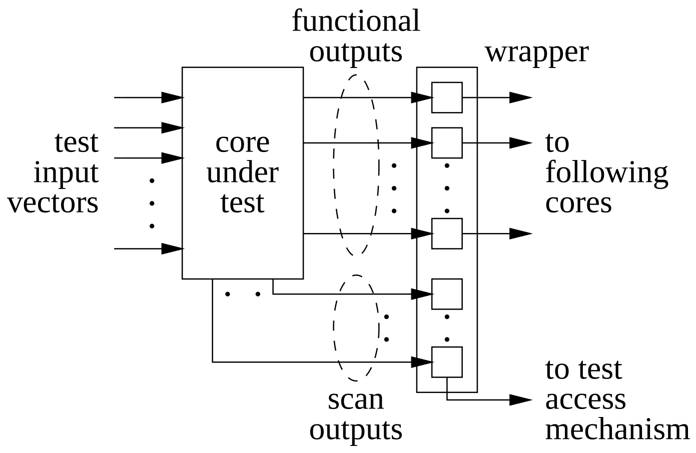


Figure 4.1: Core with test access based on a wrapper

In the previous chapters, output space compactors were designed in such a way, that minimized or eliminated aliasing is achieved for the stuck-at fault model. In order to reduce aliasing, most approaches for output compaction suggested in the literature are based on an underlying fault model. Since the method proposed in this chapter should be applicable to IP cores and usually the type of faults tested by the corresponding test set is unknown, it is important to design the space compactor so that they propagate the maximum amount of error information, irrespective of an underlying fault model. Furthermore, the methods presented in the previous chapters and also most techniques proposed in the literature are inherently unsuitable for compacting the test responses of IP cores, since they require structural knowledge about the circuit.

The approach presented in this chapter requires no information about the internal structure of the circuit under test. The design of the space compactors is based on the knowledge of a precomputed compact test set and the corresponding fault-free test responses. It guarantees, that all output errors produced by the given test set are propagated to the outputs of the compactor. Therefore, the compactors are zero-aliasing with respect to the test set.

For the purpose of simplification, below the design of zero-aliasing space compactors is described only for functional outputs, but these space compactors can also be used to compact the outputs of observation points. Such compactors can be easily designed, if the outputs of the observation points are treated as functional outputs. These compactors require no knowledge about where the observation points have been placed in the circuit and zero-aliasing for any underlying fault model is guaranteed for the given test set.

The proposed method is a general one and can be applied to any circuit with a precomputed test set. However, it yields the best results concerning compaction ratio and low area overhead, if the number of test patterns is small. In contrast, a large test set usually results in a large zero-aliasing space compactor which causes high area overhead.

Often a long test sequence includes a smaller, essential set of test patterns. For example, a pseudo random test pattern generator may generate the long test sequence. Then special logic is designed and added, so that the essential test patterns are part of the sequence. In this case, the method proposed in this chapter can be used to design a zero-aliasing compactor with respect to the essential test set.

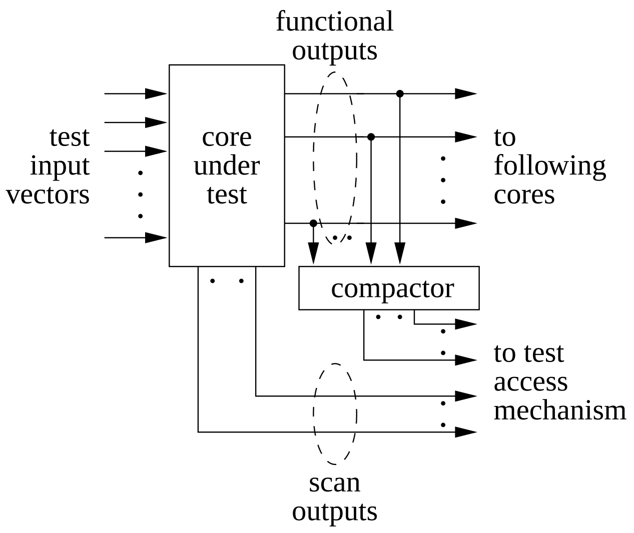


Figure 4.2: Core with test access based on a space compactor

This compactor can be combined with a second compactor, whose function improves the propagation of errors produced by the other test patterns. The second compaction function should ideally propagate the maximum amount of error information independent of the test set and the underlying fault model.

In the absence of specific test and fault information, it is reasonable to assume, that all errors are equally likely. Since it can be shown that a parity tree has the highest probability of propagating arbitrary errors, which are equally likely [58], the second compaction function can be implemented by a parity tree.

The second compactor requires an additional time step for applying the long test sequence, so that the testing time may increase. The benefit of this additional step is, that faults may be detected, which are not tested by the patterns of the smaller, essential test set.

The proposed output compaction can be applied to combinational circuits. Furthermore, the approach is also applicable to sequential circuits in full-scan design. In order to reduce the testing time required for the scan-in and scan-out operation of test patterns and responses, most current scan designs utilize the concept of multiple scan chains. Therefore, in many cases the number of functional outputs is greater than the length of a single scan chain. In order to access the test responses at the functional outputs of a circuit without further increasing the test application time, an output space compactor can be used to compact the test responses at the functional outputs and to propagate the errors to a test access mechanism as shown in Figure 4.2.

In contrast to the compactors introduced in the previous chapters, the number of outputs of the compactors described below can not be arbitrarily chosen. There is a lower bound on the number of compacted outputs, which are necessary to propagate all errors at the functional outputs during the test. The bound depends on the number of distinct test responses of the fault-free circuit.

If the space compactors work in two time steps, then the number of compacted outputs can be further reduced. In the following this concept is described more in detail. The corresponding design is called two-step compaction. The straightforward method, which requires only one compaction step per test pattern is denoted as one-step compaction. This method is presented first. Both compactor designs have been published in [76].

4.1.1 State of the Art

Various compaction designs and test access architectures for cores have been suggested [5, 6, 7, 21, 34, 37, 42, 59, 85]. Several approaches propose designs of zero-aliasing compactors, which require no knowledge of the structure of the given circuit. In the following two approaches are briefly described.

Bhattacharya, Dmitriev and Gössel propose in [5] the following design: The outputs and additionally all the inputs of the circuit under test are connected to the compactor. The compactor compresses test responses from multiple outputs to a single periodic stream observable at a single compacted output. This method achieves zero-aliasing space compaction without requiring any information about the circuit structure. The additional interconnect area required to connect the inputs of the circuit to the compactor may be high in several cases. Such a compactor is strictly speaking an "input and output" compactor. In contrast, in the following compaction including the circuit inputs is left out of consideration.

Another compactor design which is structure independent is introduced in [37] by Gössel, Sogomonyan and Morosov. The compactor propagates all errors at its inputs to the outputs. Hence it is totally error propagating. Since the compactor operates in different modes, several time steps are required for the compaction. Because the number of time steps depends on the compaction ratio, the test application time may increase considerably.

4.1.2 Basic Notations

In the following, a circuit C with m inputs $x_1, \ldots, x_m$ and n outputs $y_1, \ldots, y_n$ is considered. The space compactor SC maps n functional outputs $y_1, \ldots, y_n$ to k, $k < n$, compacted outputs $z_1, \ldots, z_k$. SC implements the boolean function f with $f = (f_1, \ldots, f_k)$ and $z_i = f_i(y_1, \ldots, y_n)$, $i = 1, \ldots, k$. The precomputed test set T of C consists of p different test input patterns $t_1, \ldots, t_p$. The corresponding fault-free test responses are $r_1, \ldots, r_p$, which form the set of fault-free responses R. The number of distinct fault-free responses of R is l. Obviously, it is $l \leq p$.

4.2 One-Step Compaction

For any given test set T a lower bound k_{min} exists on the number of compacted outputs, which are necessary to guarantee the propagation of all errors produced by T. k_{min} depends on the size l of the set of fault-free responses R as follows:

Theorem 4.1 *Let T be a test for a circuit C and let l be the number of distinct fault-free responses of C to T. Let SC be a space compactor which is connected to the functional outputs of C and which propagates any error produced by T. Then a lower bound on the number of compacted outputs k is given by*

$$k \geq \log_2(l + 1).$$

Proof:
The space compactor propagates any fault that causes the test response to change from a fault-free response r_i to a different, erroneous response r_i'. Hence $f(r_i')$ must be distinct from $f(r_i)$. Furthermore,

the compactor must map two different fault-free test responses $r_i, r_j \in R$ to distinct output vectors $f(r_i) \neq f(r_j)$. Otherwise, there may exist a fault which produces instead of the fault-free response r_i the erroneous response r'_i with $r'_i = r_j$. In this case, the vector at the outputs of the compactor is $f(r'_i) = f(r_j) = f(r_i)$, which means that the fault is not propagated.

Therefore, the varying fault-free test responses must be mapped to unique output vectors which are also distinct from the output vectors of all other erroneous test responses. Since the number of different fault-free test responses is l, the compactor must produce at least $l+1$ distinct output vectors and at least $\lceil \log_2(l+1) \rceil$ compacted outputs are necessary to generate all output vectors. $\square$

The proof of Theorem 4.1 states, that a compactor, which propagates any erroneous test response, must map the fault-free test responses to l unique output vectors. Now the set of all other test responses is denoted by R^- with $R^- = \{0,1\}^n - R$. The set R^- can be mapped to an additional unique output vector f_e. Then any erroneous test response $f(r'_i) \neq f(r_i)$ is either mapped to $f(r_j)$ with $r_i \neq r_j$ or to f_e. In both cases, the error is propagated since it is $f(r_i) \neq f(r_j)$ and $f(r_i) \neq f_e$. The $l+1$ different output vectors can be produced by $\log_2(l+1)$ compacted outputs, so that a zero-aliasing space compactor with the minimal number of outputs is obtained.

Therefore, a zero-aliasing compactor can be designed as follows:

1. Unique symbolic values are assigned to each of the fault-free responses and an additional distinct symbolic value is assigned to the set of all other erroneous responses R^-.

2. Then a logic synthesis tool can be used for mapping the symbolic values to binary values, so that the function of the resulting compactor can be simply implemented, which means that the expected area of the compactor is minimal.

3. After that the output vectors of the space compactor are fully specified for the fault-free test responses and for the set R^-. Thus, the compactor can be synthesized. Again, this can be done using a logic synthesis tool.

The design described above produces an one-step compactor, which is zero-aliasing for faults of the circuit detected by the given test set. This can only be guaranteed, if the compactor works properly. Therefore, concerning the set of faults which are located in the compactor, the number of detectable faults should be maximized. If the compactor is implemented as a two-level non-redundant AND-OR circuit and the underlying fault model is that of stuck-at faults, then it is sufficient to consider faults occurring at the inputs of the AND gates [49]. Usually, test sets include patterns that detect stuck-at faults at the functional outputs of the circuit, so that also stuck-at faults at the inputs of the space compactor are tested.

If the compactor is synthesized in such a way that the overhead is minimized, then a multi-level circuit is generated in most cases. If the multi-level implementation is obtained using algebraic factorization from the two-level functional description, then the test is preserved [2, 62]. Even if the synthesis is not restricted to special transformations, the test set designed for a two-level design can achieve high fault coverages for stuck-at faults [26].

In contrast to the compactors described in the previous chapters, the compactors generated according to the procedure given above are non-linear. Linear space compactors are not suitable, since the number of outputs of a zero-aliasing linear space compactor is never less than the number of functional outputs of the circuit:

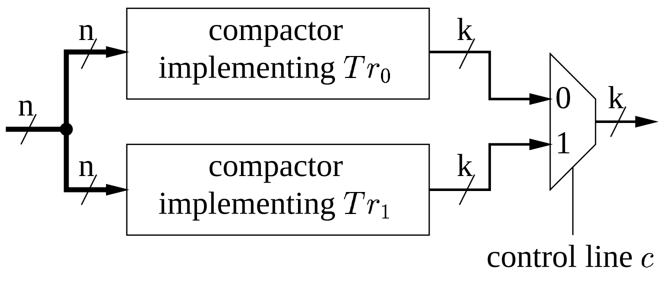


Figure 4.3: Two-step compactor based on two transmission function

Theorem 4.2 *Let T be a test for a circuit C with n outputs and let l be the number of distinct fault-free responses of C to T. If SC is a linear space compactor, which propagates any error produced by T from the outputs of C to its outputs in one time step, then the number of compacted outputs k is: $k \geq n$.*

Proof:
As mentioned above, a space compactor which propagates any erroneous test response must map the fault-free test responses to l unique output vectors. The linear compactor with k compacted outputs implements k linear functions. Each of the k linear functions is 1 for exactly 2^{n-1} input combinations dividing the input space of the compactor into two equal sets.

In the following, it is assumed, that the linear functions vary. Then each successive linear function subdivides the sets of the input space into equal halves. Therefore, $k \leq n$ different linear functions will divide the input space of the compactor into a total of 2^k sets and each set consists of 2^{n-k} input vectors. Since the compactor must map each single fault-free test response to a unique output vector, there must exist input sets containing exactly one input vector. This implies $k = n$.

Obviously, if several linear functions of the compactor are identical, then more than n compacted outputs are required. □

4.3 Two-Step Compaction

Since Theorem 4.1 provides a tight lower bound on the number of compacted outputs k, an alternative compactor design is required, if the value of k should be reduced further. The two-step compaction design increases the compaction ratio. The basic idea is, that for each single test vector t_i the corresponding response is compacted twice using two different compaction functions. The compactor includes a $2k$-input multiplexer with an additional control line, which is used to switch between the two compaction functions as shown in Figure 4.3.

The benefit of a higher compaction ratio of two-step compaction is accompanied by an increase in the test application time. If two-step compaction is used for sequential circuits in full-scan design, then loading the scan chain twice with the same vector can be avoided as follows: After serially loading the scan cells and running the functional clock for one cycle, the values at the functional output must be held stable until both compaction steps have been carried out. The two time steps of the two-step compactor can be interleaved with the scan-in and scan-out operation. Thus, in comparison to the one-step compaction approximately the same testing time is required.

The compactor implements the two compaction functions so that all errors that are not propagated in the first time step are guaranteed to be propagated in the second step. In order to determine such functions, the generation of the compactor is based on the use of orthogonal transmission functions. The notion of orthogonal transmission functions is introduced by Murray in [58].

4.3.1 Transparency and Orthogonal Transmission Functions

The function of the zero-aliasing compactor must be chosen in such a way that, after a fault modified a fault-free test response, this error is propagated to the outputs of the compactor. This requirement is related to the concept of transparency given in [55, 58]. A compactor is transparent, if any change at the inputs results in a change of the output vector. If only for a subset of input vectors any change causes a modified output vector, then the compactor is partially transparent.

In [58] Murray defines a special type of propagation function to analyze the transparency property of circuit modules. This type is called transmission function and it is defined for combinational modules with control inputs which are fed by a sequence of control values.

In order to simplify the following explanations, which describe the generation of the two compaction functions, a modified definition of transmission functions is given below. The transmission function for a single time step is:

Definition 4.3 (Transmission Function for the Control Value c) *Let M be a combinational circuit module with control inputs, data inputs and data outputs. If the control value c is applied to M and if $\{v_1, \ldots, v_p\}$ is the set of all vectors, which can occur at the data outputs of the module, then the transmission function of the compactor for the control value c is given by*

$$Tr(M, [c]) = \{(S_{c,v_1}; v_1), \ldots, (S_{c,v_p}; v_p)\}$$

where S_{c,v_i}, $1 \leq i \leq p$, is the set of all input vectors mapped to the output vector v_i.

The two-step compactor has only one control line, which controls the multiplexer switching between the two compaction functions. In the first compaction step the control line of the compactor SC is set to 0. Therefore, the corresponding transmission function of the compactor is denoted by $Tr(SC, [0])$. If the control line is set to 1, then the transmission function of the compactor SC is $Tr(SC, [1])$. Again, for simplification here the definition of the transmission function for two or more time steps differs slightly from the original definition given in [58]:

Definition 4.4 (Transmission Function) *Let M be a combinational circuit module with control inputs, data inputs and data outputs. Let $CV_{seq} = [c_1, \ldots, c_t]$ be a sequence of t control values. If the control sequence CV_{seq} is applied to M in t time steps and if $\{V_{seq,1}, \ldots, V_{seq,p}\}$ is the set of all sequences of vectors, which can occur at the data outputs of the module, then the transmission function of the compactor for control sequence CV_{seq} is given by*

$$Tr(M, CV_{seq}) = \{(S_1; V_{seq,1}), \ldots, (S_p; V_{seq,p})\}$$

where S_i, $1 \leq i \leq p$, is the set of all input vectors, which are mapped to the sequence of output vectors $V_{seq,i}$.

Since Definition 4.4 includes the case, that the sequence CV_{seq} consists only of one control value, it comprises Definition 4.3. Furthermore, a trivial definition of transmission functions of modules without control inputs can be simply derived from these definitions.

The definition given by Murray in [58] includes an equation, which describes the computation of the sets $S_1, \ldots, S_q$ of $Tr(M, CV_{seq})$ from the input sets given in Definition 4.3. For a fixed control value c and for a single output vector v the input sets of the corresponding transmission function $Tr(M, [c])$ is $S_{c,v}$. If CV_{seq} consists of the control values c_j, $1 \leq j \leq t$, then the input set S_i mapped to a sequence of output vectors $V_{seq,i} = [v_{i,1}, \ldots, v_{i,t}]$ can be determined as follows:

$$S_i = S_{c1,v_{i,1}} \cap \ldots \cap S_{ct,v_{i,t}}. \tag{4.1}$$

Thus, the transmission function $Tr(M, CV_{seq})$ for a control sequence can be easily derived from the corresponding transmission functions $Tr(M, [c_1]), \ldots, Tr(M, [c_t])$ for one time step.

Since the intersection operation is commutative, Equation (4.1) implies, that the input sets of a transmission function $Tr(M, CV_{seq})$ are independent from the order of the control values of the sequence CV_{seq}, which is applied to M in t time steps. Furthermore, if a control value is added to the sequence, which is already included in CV_{seq}, then the input sets remain unmodified.

Because here transmission functions are only used for a two-step compactor SC, now the transmission functions $Tr(SC, [0])$ and $Tr(SC, [1])$ are denoted by Tr_0 and Tr_1, respectively. Furthermore, in the following it is assumed, that the input lines of the compactor are ordered. Then the vectors assigned to the input lines can be described by the binary value according to this order. Therefore, below the sets of input vectors are written as a list of binary values which corresponds to the notation of [58]. Analogically, a single output vector is represented by the corresponding binary value.

For example, if a compactor with two inputs and one output is given and for the control input $c = 0$ ($c = 1$) the first (second) function of the compactor equals the OR-function (XOR-function), then it is:

$$\begin{aligned}
Tr_0 &= \{(0;0), (1,2,3;1)\}, \\
Tr_1 &= \{(0,3;0), (1,2;1)\}, \\
Tr(SC, [0,1]) &= \{(0;00), (3;10), (1,2;11)\}, \\
Tr(SC, [1,0]) &= \{(0;00), (3;01), (1,2;11)\}.
\end{aligned}$$

As mentioned above, the input sets of $Tr(SC, [0,1])$ and $Tr(SC, [1,0])$ are identical, since the control sequences differ only in the order of the control values.

The amount of information, that can be propagated by the compactor SC, depends on the structure of the input sets in SC's transmission function. If each set consists only of one element, then SC is transparent. If not all but several sets consist of only one input vector, then the compactor is called partially transparent with respect to these input vectors.

The basic idea of the generation of a two-step compactor is to choose transmission functions Tr_0 and Tr_1, so that the combination of these functions in two time steps results in a compactor with improved transparency. The proposed method determines orthogonal transmission functions to achieve this. According to [58] the definition of orthogonal transmission functions is:

Definition 4.5 (Orthogonal Transmission Functions) *Let M be a combinational circuit module with control inputs, data inputs and data outputs. For the two sequences of control values $C_{seq,0}$ and $C_{seq,1}$ let $Tr(M, C_{seq,0})$ and $Tr(M, C_{seq,1})$ be the corresponding transmission functions of M. If no two elements which are included in the same input set of $Tr(M, C_{seq,0})$ appear in a common input set of $Tr(M, C_{seq,1})$, then $Tr(M, C_{seq,0})$ and $Tr(M, C_{seq,1})$ are orthogonal transmission functions.*

In some cases only for a subset of input vectors the condition is fulfilled, that no two elements which are included in the same input set of $Tr(M, C_{seq,0})$ appear in a common set of input vectors of $Tr(M, C_{seq,1})$. In this case $Tr(M, C_{seq,0})$ and $Tr(M, C_{seq,1})$ are called orthogonal transmission functions with respect to this subset.

The two transmission functions Tr_0 and Tr_1 of the example given above are not orthogonal, since the inputs 1 and 2 are included in the same input set for both functions. The corresponding compactor SC is only partially transparent with respect to $\{0, 3\}$.

If the example above is modified such that the first function of the compactor is equal to the XNOR-function, then the corresponding transmission function is $Tr_0 = \{(1, 2; 0), (0, 3; 1)$. In this case, the Tr_0 and Tr_1 are not orthogonal either, since the inputs 1 and 2 are included in the same input set for both functions, even when different output values are assigned to the input sets. The corresponding compactor SC is not partially transparent.

Finally, if in the first compaction step the first input line of the compactor is fed through, then the resulting transmission function is $Tr_0 = \{(0, 2; 0), (1, 3; 1)$. Obviously, the transmission functions Tr_0 and Tr_1 are orthogonal, since it is $Tr(SC, [0, 1]) = \{(0; 00), (2; 01), (3; 10), (1; 11)\}$ and $Tr(SC, [1, 0]) = \{(0; 00), (3; 01), (2; 10), (1; 11)\}$. Therefore, the corresponding two-step compactor is transparent.

A compactor SC, which propagates any error which is produced by the test set T at the functional outputs of the circuit, must partially transparent for a certain subset of the input vectors. This is stated in the following theorem:

Theorem 4.6 *Let T be a test set for a circuit C and let l be the number of distinct fault-free responses of C to T. Let SC be a space compactor with control inputs, which is connected to the functional outputs of C. For a given sequence CV_{seq} of control values let S_i, $1 \leq i \leq p$, be the input sets of the corresponding transmission function $Tr(SC, CV_{seq})$. Then the compactor propagates any error produced by T, if and only if each fault-free test response r is the only element in its input set S_i.*

Proof:

1. If each input set S_i, which contains a fault-free test response includes no other input vectors, then any erroneous modification of the fault-free test response is an element of a different input set. This leads to at least one different output vector during the application of the sequence of control values. Thus, such an error is propagated through the compactor.

2. If a fault-free test response r and any other vector r' are included in the same input set S_i, then a fault may be not propagated. For example, if the test t which corresponds to r is applied and a fault modifies r in such a way that the erroneous response is r', then the fault is masked. The reason is, that r_i and r' are elements of the same input set and hence the compactor produces the same sequence of output vectors. □

Obviously, this theorem is also applicable to space compactors without control inputs. The result given in this theorem, that for a zero-aliasing compactor any input set, which includes a fault-free test response, consists of exactly this single element, is used in the proof of the following theorem. The theorem gives a lower bound on the number of outputs which are required for a zero-aliasing two-step compactor.

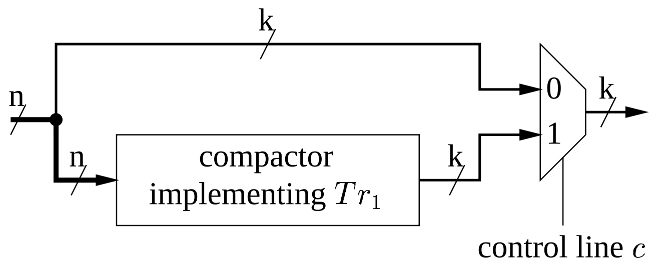


Figure 4.4: Proposed design of the two-step compactor

Theorem 4.7 *Let T be a test set for a circuit C and let l be the number of distinct fault-free responses of C to T. Let SC be a two-step space compactor, which is connected to the functional outputs of C and which propagates any error produced by T for a given sequence CV_{seq} of control values. Then a lower bound on the number of compacted outputs k is given by*

$$k \geq \frac{\log_2(l+1)}{2}.$$

Proof:
Firstly, a lower bound on the number of different input sets of the corresponding transmission function is derived: According to Theorem 4.6, each input set, which contains a fault-free test response, does not include any other input vector, since SC propagates any error produced by T. Additionally, at least one input set contains the remaining, erroneous test responses of R^-, so that at least $l+1$ different input sets of the transmission function exist.

If the number of compacted outputs is k then in both compaction steps at most 2^k different output vectors are possible. Furthermore, the number of input sets of the transmission functions of the first and second step is at most 2^k. Therefore, one of the input sets of the transmission function Tr_0 contains at least $(l+1)/2^k$ elements, which must be assigned to different input sets of the transmission function of SC.

These $(l+1)/2^k$ elements must be assigned to different input sets of the transmission function Tr_1 of the second step. As mentioned above, the number of input sets is at most 2^k. Thus, it is:

$$\begin{aligned} \frac{l+1}{2^k} &\leq 2^k \\ l+1 &\leq 2^{2\cdot k} \\ 2\cdot k &\geq \log_2(l+1) \\ k &\geq \frac{\log_2(l+1)}{2} \end{aligned}$$

□

Obviously, the lower bound on the number of outputs for two-step compactors is the corresponding bound for one-step compactors divided by two.

4.3.2 Determination of the Compaction Functions of the Two Time Steps

This section describes in detail how the compaction functions of the two steps are determined. In order to reduce the area overhead of the resulting compactors, a design is introduced, which trivially

```
1.  responses_bound := $\lceil l/2^k \rceil$;
2.  while (no_solution_found) do
3.      #outputs_extracted := 0;
4.      search_start_output := 1;
5.      SelectOutputs(#outputs_extracted, search_start_output);
6.      responses_bound := responses_bound $+1$;
7.      if (responses_bound= $2^k$) then
8.          $k$ := $k + 1$;
9.          responses_bound := $\lceil l/2^k \rceil$;
10.     endif;
11. endwhile;
12. end.
```

Figure 4.5: The algorithm `GetTr0` which determines functional outputs implementing Tr_0

implements Tr_0 using a set of k feed-through wires. Hence the expected area required for additional gates and interconnection is smaller. The second step of the compaction is derived from the first step in such a way, that the two transmission functions Tr_0 and Tr_1 are orthogonal with respect to the fault-free test responses. Furthermore, one input set of Tr_1 equals the set R^-, which consists of all test responses, that are not generated by the fault-free circuit.

The compaction of the first compaction step is given by k carefully chosen functional outputs of the circuit C. These outputs are directly connected to the multiplexer of the two-step compactor as shown in Figure 4.4.

Determination of the First Compaction Function The k functional outputs are chosen in such a way that the fault-free test responses are distributed as uniformly as possible among the input sets of the corresponding transmission function Tr_0. This means, that the maximal number of fault-free test responses, which are included in a single input set of Tr_0, is minimized.

The algorithm `GetTr0` can be used to determine the functional outputs implementing trivially Tr_0. The pseudocode of the algorithm is given in Figure 4.5. The input parameters of the algorithm are the fault-free test responses at the functional outputs of C and k, which is the number of outputs of the compactor. k is also the number of outputs of C, which the algorithm should choose. The algorithm calls the recursive algorithm `SelectOutputs` in order to perform an exhaustive search for k outputs. If the search fails, then no k functional outputs can be chosen so that each input set of the corresponding transmission function Tr_0 contains at most $2^k - 1$ fault-free test responses. Furthermore, this means no transmission function Tr_1, which is orthogonal to Tr_0 with respect to the fault-free test responses, can be generated. In this case, no set of k feed-through outputs exists for a zero-aliasing two-step compactor with k compacted outputs. Therefore, the algorithm `GetTr0` increases k.

In particular, the transmission function Tr_0 of the first step produces 2^k different input sets for k compacted outputs. In the best case, each of the input sets contains at most $\lceil l/2^k \rceil$ fault-free test responses. This bound is determined in step 1 of the algorithm `GetTr0`.

Then the initial parameters of the recursive algorithm `SelectOutputs` are determined and in step 5 the algorithm is called. The algorithm `SelectOutputs` searches for k functional outputs so

```
1.  #outputs_extracted := #outputs_extracted $+1$;
2.  for next_output = search_start_output to $n$ do
3.          select output according to “next_output” and determine input sets of
            the transmission function implemented by the outputs selected so far
4.          max_responses_of_set := determine_max_responses_of_set();
5.          if (max_responses_of_set $\leq$ responses_bound) then
6.                return(solution_found);
7.          else if(max_responses_of_set $\leq 2^{k-\#outputs\_extracted}\cdot$ responses_bound) then
9.                SelectOutputs(#outputs_extracted, next_output$+1$);
10.               if (solution_found) then return(solution_found);
11.         endif;
12. endfor;
13. return(no_solution_found);
14. end.
```

Figure 4.6: The recursive algorithm `SelectOutputs` called by `GetTr0`

that for the generated transmission function Tr_0 the number of fault-free test responses included in the same input set is not larger than the given bound. In step 6 this bound is incremented.

If the bound equals 2^k, then no solution for k compacted outputs exists, because 2^k is the number of different input sets of the second transmission function Tr_1 and one of these sets is reserved for R^-. If the equation of step 7 is true, then in the two following steps k is incremented and a new bound is computed analogically to step 1. If k outputs could be sucessfully chosen, then the algorithm ends, otherwise the search starts again with the modified value of the bound.

The pseudocode of the algorithm `SelectOutputs` is shown in Figure 4.6. The additional parameters given by its call are the number of functional outputs, which are already chosen for Tr_0, and the number of the next functional output, which should be checked for selection. In step 1 of this algorithm the number of selected outputs is incremented. Then the next output is selected. Furthermore, the transmission funtion Tr_0 and the input sets are computed corresponding to the outputs selected so far.

In step 5 it is tested, if one of the input sets contains too many fault-free test responses, which means that the bound given by `GetTr0` is exceeded. If the bound is valid, then a solution has been found and the algorithm returns the selected functional outputs. Otherwise the inequation of step 7 is used to check, whether the selection of more functional outputs can result in a solution or not. If a solution can possibly found, a recursive call of `SelectOutputs` follows in step 9. If no solution is possible, the last selected functional output is dropped. If there are still functional outputs which can be chosen, then the next output is selected in step 2. Otherwise the algorithm returns no solution.

The algorithm `GetTr0` performs an exhaustive search. Thus, the complexity of the algorithm is $O(\binom{n}{k}\cdot l) < O(n^k \cdot l)$, if the algorithm does not increase the number of compacted outputs k. Below it is shown, that a very small value can be chosen for k. Furthermore, functional outputs, which can not be part of a solution are often skipped, because of the condition given in step 7 of the algorithm `SelectOutputs`. Therefore, in practice in many cases the algorithm `GetTr0` requires a short execution time.

For example, for the ISCAS ’85 benchmarks circuit c6288 a test set of 12 input vectors is available.

Test	Response values of functional outputs													
response	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	$\dots$
r_1	1	1	0	0	1	1	**1**	0	0	1	**1**	0	1	$\dots$
r_2	0	0	0	0	0	0	**0**	0	0	0	**0**	0	0	$\dots$
r_3	1	1	1	1	0	1	**0**	0	0	0	**1**	0	0	$\dots$
r_4	1	0	0	0	0	1	**1**	0	0	0	**0**	1	1	$\dots$
r_5	1	0	0	0	0	0	**0**	0	0	1	**0**	0	0	$\dots$
r_6	0	0	1	1	0	0	**0**	1	0	1	**1**	1	1	$\dots$
r_7	0	1	0	0	1	1	**1**	0	1	0	**0**	0	0	$\dots$
r_8	0	1	1	1	0	1	**0**	0	1	1	**1**	1	1	$\dots$
r_9	1	1	0	1	0	1	**1**	0	1	1	**0**	1	0	$\dots$
r_{10}	1	0	0	0	1	1	**0**	0	1	1	**0**	0	0	$\dots$
r_{11}	1	0	1	0	1	1	**1**	1	1	0	**1**	0	1	$\dots$
r_{12}	0	0	1	0	0	1	**1**	0	1	0	**1**	0	0	$\dots$

Table 4.1: Test responses of the c6288 benchmark circuit

This set detects all non-redundant single stuck-at faults of c6288. Table 4.1 gives for the corresponding fault-free test responses $r_1, \dots, r_{12}$ the values of 13 functional outputs of c6288.

If the number of compacted outputs is set to 2, then two output lines must be chosen, so that each input set of the generated transmission function Tr_0 contains three fault-free test responses. The algorithm `GetTr0` determines this bound in the first step. Then the algorithm `SelectOutputs` is called. The search starts with output y_1 whose values are "1" for seven test responses and "0" for five test responses. Thus, the two input sets of the corresponding transmission function of the first output contain five and seven test responses, respectively. The maximal number of fault-free test responses included in the same input set is seven. Obviously, the inequation $7 \leq 3$ of step 5 of `SelectOutputs` is not true and no solution has been found so far. The inequation $7 \leq 2 \cdot 3$ of step 7 is not true either. Hence the output y_1 is skipped.

Only outputs, whose values are "1" for six test responses and "0" for six test responses are not skipped, because only for these outputs the inequation of step 7 is true. The first output with this property is y_7. Then the algorithm recursively calls itself in order to determine the second output. The search starts with y_8 and after additionally checking four outputs the algorithm finds the solution y_7 and y_{11}.

The output values corresponding to the solution are marked in Table 4.1. The two columns of y_7 and y_{11} show each of the combinations 00, 01, 10, and 11 three times. The transmission function Tr_0 is:

$$Tr_0 = \{(r_2, r_5, r_{10}, R_0^-; 0), (r_3, r_6, r_8, R_1^-; 1), (r_4, r_7, r_9, R_2^-; 2), (r_1, r_{11}, r_{12}, R_3^-; 3)\}$$

where $R_0^-, \dots, R_3^-$ denote the parts of the set R^- of erroneous test responses, which are mapped to the output values $0, \dots, 3$, respectively.

Determination of the Second Compaction Function The second step of the compaction is derived from the first compaction function so that the two transmission functions Tr_0 and Tr_1 are

```
1.  $R_0 = R$;
2.  **for** output_value_of_$Tr_1$=1 **to** $2^k - 1$ **do**
3.      **for** output_value_of_$Tr_0$=0 **to** $2^k - 1$ **do**
4.          **for each** fault-free test response $r \in R_0$ **do**
5.              **if** ($Tr_0(r)$ = output_value_of_$Tr_0$) **then**
6.                  $Tr_1(r)$ := output_value_of_$Tr_1$;
7.                  remove $r$ in $R_0$;
8.                  **break**;   // of "For each fault-free test response $r \in R$"
9.              **endif**;
10.         **endfor**;
11.         **if** ($|R_0| \leq 2^k - 1$ - output_value_of_$Tr_1$) **then**
12.             **for each** fault-free test response $r \in R$ **do**
13.                 output_value_of_$Tr_1$ := output_value_of_$Tr_1$ $+1$;
14.                 $Tr_1(r)$ := output_value_of_$Tr_1$;
15.             **endfor**;
16.             **break**;        // of "For output_value_of_$Tr_0$=0 to $2^k - 1$"
17.         **endif**;
18.     **endfor**;
19. **endfor**;
20. **end**.
```

Figure 4.7: The algorithm `GetTr1` which derives Tr_1 from Tr_0

orthogonal with respect to the fault-free test responses. That means, that fault-free test responses, which are included in the same input set of Tr_0, are assigned to different input sets of Tr_1. Finally, one additional input set of Tr_1 equals R^-, which consists of all test responses that are not generated by the fault-free circuit.

The algorithm `GetTr1` can be used to determine the second compaction function implementing Tr_1. The pseudocode of the algorithm is given in Figure 4.7.

In step 1, the algorithm `GetTr1` copies the set R of fault-free test responses to the set R_0. R_0 is the set of fault-free test responses, which are not yet included in any input set of Tr_1. Since one input set is reserved for R^-, the algorithm assigns the elements of R_0 to the remaining 2^{k-1} input sets. In step 2, the algorithm gradually generates all output values corresponding to these 2^{k-1} input sets. Then for such an output value v the corresponding input set of Tr_1 is generated: For each input set S of Tr_0 the algorithm searches in R_0 for exactly one test response $r \in S$. If the search is successful, then in step 6 the algorithm assigns the output vector v to r, so that the test response is added to the new input set corresponding to v, and r is removed from the set R_0. Since at most one test response out of each input set S of Tr_0 is chosen for the new input set of Tr_1, the resulting transmission function Tr_1 is orthogonal to Tr_0 with respect to the fault-free test responses.

The comparison of step 11 checks, whether the number of fault-free test responses, which are not yet included in an input set of Tr_1, is equal or less than the number of output values of Tr_1, which are unused so far. If the inequation is true, then the remaining fault-free test responses of R_0 are assigned to the input sets, which are related to the output values unused so far. Because in step 14 of the algorithm these responses are assigned to distinct output values, each of the generated input sets consists of a single element. Thus, the steps 11 to 17 of the algorithm are only added to increase the likelihood of input sets consisting of a single fault-free test response.

The algorithm `GetTr1` can be implemented in a way that is different from the one given in Figure 4.7, so that the complexity of the algorithm is linear in l, which is the number of distinct fault-free test responses.

For example, the description of the compactor design for the circuit c6288 with a test set consisting of 12 test vectors is continued. Tr_0 can be generated by the algorithm `GetTr0` as described above, yielding:

$$Tr_0 = \{(r_2, r_5, r_{10}, R_0^-; 0), (r_3, r_6, r_8, R_1^-; 1), (r_4, r_7, r_9, R_2^-; 2), (r_1, r_{11}, r_{12}, R_3^-; 3)\}.$$

$R_0^-, \dots, R_3^-$ denote the parts of the set of erroneous test responses R^-, which are mapped to the output values $0, \dots, 3$, respectively.

The algorithm `GetTr1` computes Tr_1 based on Tr_0. The first output value of Tr_1 which is considered in step 2 of `GetTr1` is 1. The algorithm gradually assigns the output value 1 to the first element of each input set of Tr_0 resulting in the input set $\{r_2, r_3, r_4, r_1\}$, which is mapped to the first output value. r_2, r_3, r_4, and r_1 are removed from R_0, so that these fault-free test responses are not considered, when the input set of the next output value 2 is determined. Therefore, the input set of value 2 is $\{r_5, r_6, r_7, r_{11}\}$. Again, the elements of the new input set are removed from R_0. Finally, after the elements of the input set $\{r_{10}, r_8, r_9, r_{12}\}$ of the output value 3 have been removed from R_0, it is $R_0 = \emptyset$.

The set R^- of test responses, which can not be produced by a fault-free circuit, is mapped to the output value 0, so that finally the following transmission function Tr_1 is obtained:

$$Tr_1 = \{(R^-; 0), (r_2, r_3, r_4, r_1; 1), (r_5, r_6, r_7, r_{11}; 2), (r_{10}, r_8, r_9, r_{12}; 3)\}$$

Thus, the two steps of the compaction function are specified and a compactor SC, which implements these two transmission functions Tr_0 and Tr_1, can be synthesized. Then the resulting transmission function is:

$$\begin{aligned} Tr(SC, [0,1]) \quad = \quad & \{(R_0^-; 00), (r_2; 01), (r_5; 02), (r_{10}; 03), \\ & (R_1^-; 10), (r_3; 11), (r_6; 12), (r_8; 13), \\ & (R_2^-; 20), (r_4; 21), (r_7; 22), (r_9; 23), \\ & (R_3^-; 30), (r_1; 31), (r_{11}; 32), (r_{12}; 33)\}. \end{aligned}$$

Obviously, the transmission function of SC is partially transparent with respect to the set of fault-free responses R, since the transmission functions Tr_0 and Tr_1 were generated in such a way, that these functions are orthogonal with respect to R. It follows, that the compactor SC propagates any error produced by T if both control values 0 and 1 are applied.

4.3.3 Lower Bounds on the Number of Compacted Outputs

In this section, a lower bound k_{min} on the number of compacted outputs is given for the design of zero-aliasing two-step compactors introduced above. For any given test set T, the bound k_{min} depends on the number of distinct fault-free responses of C as follows:

Theorem 4.8 *Let T be a test set for a circuit C and let l be the number of distinct fault-free responses of C to T. Let SC be a zero-aliasing two-step space compactor generated as described above. Then a lower bound on the number of compacted outputs k is given by*

$$k_{min} \geq \log_2(\sqrt{4 \cdot l + 1} + 1) - 1.$$

Proof:
According to Theorem 4.6 each input set, which contains a fault-free test response, does not include any other input vector, since SC propagates any error produced by T. Thus, there are at least l input sets consisting of a single vector.

If the number of compacted outputs is k, then for both compaction steps at most 2^k different output vectors exist. This implies, that the number of input sets of the transmission functions of the first and second step is at most 2^k. Furthermore, one of the input sets of the transmission function Tr_0 contains at least $l/2^k$ fault-free test responses.

So at least $l/2^k$ fault-free test responses must be assigned to different input sets of the transmission function Tr_1 of the second step. As stated above, the number of input sets is at most 2^k. One of the input sets is reserved for the responses of R^-, that are not generated by the fault-free circuit under test. Thus, at least $l/2^k$ test responses must be assigned to $2^k - 1$ different input sets:

$$\begin{aligned} \frac{l}{2^k} &\leq 2^k - 1 \\ l &\leq 2^k \cdot (2^k - 1) \\ 0 &\leq K^2 - K - l \text{ for } K = 2^k \end{aligned}$$

The two solutions of the last quadratic inequation are

$$\begin{aligned} 0 &\leq K - \frac{1 + \sqrt{1 + 4 \cdot l}}{2} \quad \text{for } K - \frac{1}{2} \geq 0 \text{ and} \\ 0 &\leq K - \frac{1 - \sqrt{1 + 4 \cdot l}}{2} \quad \text{for } K - \frac{1}{2} < 0. \end{aligned} \tag{4.2}$$

The latter solution with $K < \frac{1}{2}$ is not valid. That implies:

$$\begin{aligned} K &\geq \frac{1 + \sqrt{4 \cdot l + 1}}{2} \\ k &\geq \log_2(\frac{\sqrt{4 \cdot l + 1} + 1}{2}). \\ k &\geq \log_2(\sqrt{4 \cdot l + 1} + 1) - 1. \end{aligned}$$

□

Obviously, the bound on the number of outputs of any zero-aliasing two-step compactor given in Theorem 4.7 can not be greater than the bound on the number of outputs of the special design introduced above. This can be shown as follows:

$$\log_2(\sqrt{1 + 4 \cdot l} + 1) - 1 = \log_2(\frac{\sqrt{1 + 4 \cdot l} + 1}{2})$$

$$\begin{aligned}
&= \frac{1}{2} \cdot \log_2(\frac{\sqrt{1+4\cdot l}+1}{2})^2 \\
&= \frac{1}{2} \cdot \log_2(\frac{1+4\cdot l+2\cdot\sqrt{1+4\cdot l}+1}{4}) \\
&\geq \frac{1}{2} \cdot \log_2(\frac{1+4\cdot l+2+1}{4}) \\
&= \frac{1}{2} \cdot \log_2(1+l) \\
&= \frac{\log_2(l+1)}{2}.
\end{aligned}$$

4.3.4 Generation of Zero-Aliasing Two-Step Space Compactors

For a circuit C with precomputed test set T the procedure for designing a zero-aliasing two-step compactor SC is given as follows:

1. The minimum number of compactor outputs k_{min} is computed based on Theorem 4.8.
2. Then the algorithm `GetTr0` is used to search for k_{min} functional outputs so that at most $2^{k_{min}}-1$ fault-free test responses are included in any input set of the corresponding transmission function Tr_0. If the search fails, then the algorithm increases the number of outputs. Finally, algorithm `GetTr0` returns k functional outputs ($k \geq k_{min}$).
3. The transmission function Tr_1 of the second compaction step is computed using algorithm `GetTr1` so that Tr_0 and Tr_1 are orthogonal with respect to the set of fault-free test responses.
4. Unique symbolic values are assigned to the 2^k input sets of Tr_1. Then a logic synthesis tool can be used for mapping the symbolic values to binary values in such a way that the function of the second compaction step can be simply implemented, which means that the expected area is minimal.
5. Then the output vectors of the second compaction step are fully specified for the fault-free test responses and for the set R^-. Thus, the compactor can be synthesized. Again, this can be done using a logic synthesis tool.
6. Finally, the multiplexer is connected to the functional outputs determined by the algorithm `GetTr0` and to the compactor designed for the second step.

The forth step modifies the transmission function determined by the algorithm `GetTr1`. The modified transmission function is still orthogonal to Tr_0 with respect to the fault-free test responses, since only the output values of Tr_1 are reassigned to the input sets in order to reduce the area required for the implementation.

While the two-step compactor is zero-aliasing for any fault of the circuit, which is detected by the given test set, a maximal number of faults, which affect the compactor should also be detectable. Since the function of the trivially implemented first compaction step and the multiplexer can be easily tested, in the following only the testing of the faults of the second compaction step is discussed: Testing the second compaction step is analogical to testing a one-step compactor. Hence the same conclusions as described above for one-step compactors can be derived: For a two-level non-redundant AND-OR realization usually all stuck-at faults of the second compaction step are detected. Even for a multi-level design high fault coverages for stuck-at faults can be expected.

4.3.5 Reduction of Testing Time for Two-Step Space Compactors

Although the design for zero-aliasing two-step space compactors introduced above is based on two compaction steps implementing two orthogonal transmission functions, the first compaction step can be omitted under certain conditions.

In this case, for some vectors of the test set it is sufficient to compact the test response only once using the compactor, which implements Tr_1. Thus, the testing time can be reduced from $2 \cdot |T|$ cycles to $(1 + \alpha) \cdot |T|$ cycles, where $0 < \alpha \leq 1$ depends on the number of compacted outputs k and the number of fault-free test responses l.

The following lemma characterizes the patterns of T, for which the first compaction step is not necessary to achieve zero-aliasing error propagation of the compactor:

Lemma 4.9 *Let T be a test set for a circuit C and let l be the number of distinct fault-free responses of C to T. Let SC be a zero-aliasing two-step space compactor generated as described in Section 4.3.4. It is sufficient to compact the circuit response of a test pattern $t \in T$ in one step using the compactor implementing Tr_1, if and only if the fault-free test response r corresponding to t is the only element in its input set of Tr_1.*

Proof:

1. If the input set, which contains the fault-free test response r does not include any other input vectors, then any erroneous modification of the fault-free test response is an element of a different input set of Tr_1. Therefore, a different output vector is produced and the erroneous test response to t is observable at the compacted outputs.

2. Since one input set of Tr_1 consists of all erroneous test responses in R^-, the input sets of Tr_1, which contain a fault-free test response, do not include any erroneous test response. If two fault-free test responses r and r' are included in the same input set, then a fault may not be propagated by Tr_1. If the test t, which corresponds to r is applied and a fault modifies the test response to t so that the erroneous response equals r', then the fault is masked. The reason is, that r and r' are in the same input set of Tr_1 and thus this compaction step produces the same output vector. In this case, the first compaction step must also be carried out, since it implements Tr_0, which is orthogonal to Tr_1 with respect to R. □

For the reduction of testing time the test vectors, which can be discarded in the first compaction step, can be easily derived from the transmission function Tr_1. These test vectors are mapped to fault-free test responses, which are the only elements in their input sets. In the following, the set of these test vectors is denoted by $T_1 \subset T$. In order to achieve the reduction from $2 \cdot |T|$ cycles to $(1 + \alpha) \cdot |T|$ cycles, the test responses to vectors of T_1 are compacted only once using the transmission function Tr_1. Therefore, it is $\alpha = (|T| - |T_1|)/|T|$.

The steps 11 to 17 of the algorithm `GetTr1`, which generates Tr_1 based on Tr_0, increases the likelihood of input sets consisting of a single fault-free test response. Thus, the size of T_1, which is the set of test vectors requiring only the second compaction step, is maximized.

The following lemma gives an upper bound on the number of test vectors, which are the only elements in their input sets:

Lemma 4.10 *Let T be a test set for a circuit C and let l be the number of distinct fault-free responses of C to T. Let SC be a zero-aliasing two-step space compactor generated as described in Section 4.3.4 with k outputs. Let $T_1 \subset T$ be the set of test vectors, for which one compaction step implementing Tr_1 is sufficient. An upper bound on the size of T_1 is given by*

$$|T_1| \leq 2^k - \frac{l}{2^k - 1}.$$

Proof:
If T_1 consists of q test vectors, then there are q input sets of Tr_1, which consist of a single fault-free test response. One input set of Tr_1 contains all test responses, which can be only generated by a faulty circuit. Each of the remaining $2^k - q - 1$ input sets of Tr_1 contains at most 2^k fault-free test responses. This is true since Tr_0 and Tr_1 are orthogonal transmission functions with respect to R. That means, that two fault-free test responses in the same input set of Tr_1 must belong to different input sets of Tr_0. The number of different input sets of Tr_0 is 2^k. In consequence, the number of fault-free test responses included in any input set of Tr_1 is at most 2^k.

Therefore, the number of fault-free test responses included in the input sets of Tr_1 is at most $q + (2^k - q - 1) \cdot 2^k$. The number of fault-free test responses is l, yielding:

$$\begin{aligned} l &\leq q + (2^k - q - 1) \cdot 2^k \\ l &\leq q - 2^k \cdot q + (2^k - 1) \cdot 2^k \\ 2^k \cdot q - q &\leq (2^k - 1) \cdot 2^k - l \\ q &\leq 2^k - \frac{l}{2^k - 1} \\ |T_1| &\leq 2^k - \frac{l}{2^k - 1}. \end{aligned}$$

□

For example, for the test set of circuit c6288 T_1 is empty, since it is $l = 12$ and $k = 2$ yielding $|T_1| \leq 4 - \frac{12}{3} = 0$. In general, no reduction in testing time is possible, if it is $2^k - \frac{l}{2^k - 1} < 1$.

Lemma 4.10 suggests for a given test set, that the greater the compaction ratio is, the smaller is the number of test vectors, which require only the second compaction step. Hence the test application time can be traded off with the compaction ratio by choosing the number of compacted outputs k, so that k is slightly higher than the lower bound given in Theorem 4.8. The following theorem characterizes the rate at which k must grow, so that the value $\alpha = (|T| - |T_1|)/|T|$, which describes the reduction of testing time, remains constant with increasing l.

Theorem 4.11 *Let T be a test set for a circuit C and let l be the number of distinct fault-free responses of C to T. Let SC be a zero-aliasing two-step space compactor generated as described in Section 4.3.4. Let $T_1 \subset T$ be the set of test vectors for which one compaction step implementing Tr_1 is sufficient. If $\alpha = (|T| - |T_1|)/|T|$ is to remain constant with increasing size of T, then the growing rate of the number of compacted outputs k is $k = O(\log_2 |T|)$.*

Circuit	Number of			
	gates	inputs	outputs	flipflops
c499	202	41	32	0
c880	357	60	26	0
c1355	514	41	32	0
c3540	1667	50	22	0
c6288	2416	32	32	0
s35932	17793	323	320	1728
s38417	23815	28	106	1636

Table 4.2: ISCAS benchmark circuits

Proof:
In Theorem 4.10 is a lower bound on $|T_1|$ given. It follows:

$$\begin{aligned} |T_1| &\leq 2^k - \frac{l}{2^k-1} \\ \frac{|T_1|}{|T|} &\leq \frac{2^k}{|T|} - \frac{l}{|T|\cdot(2^k-1)}. \end{aligned}$$

It is $\frac{l}{|T|} \leq 1$ since the number of distinct fault-free test responses is equal or less than the number of test vectors. Furthermore, obviously it is $1-\alpha = \frac{|T_1|}{|T|}$. Then it is obtained:

$$\begin{aligned} 1-\alpha &\leq \frac{2^k}{|T|} - \frac{1}{2^k-1} \\ 1-\alpha &\leq \frac{K}{|T|} - \frac{1}{K-1} \text{ for } K = 2^k. \end{aligned}$$

Several steps result in the following quadratic inequation:

$$0 \leq K^2 + (|T|\cdot\alpha - |T| - 1)\cdot K - |T|\cdot\alpha \tag{4.3}$$

which yields

$$K \leq \frac{|T|\cdot\alpha - |T| - 1 + \sqrt{(|T|\cdot\alpha - |T| - 1)^2 + 4\cdot|T|\cdot\alpha}}{2}. \tag{4.4}$$

If α is constant, then it is $K = O(|T|)$ and $k = O(\log_2|T|)$. □

4.4 Experimental Results

In this section, experimental results on the synthesis of one-step and two-step space compactors for several ISCAS benchmark circuits [12, 13] are presented. The outputs of the combinational benchmark circuits c499, c880, c1355, c3540, c6288 and the functional outputs of the full-scan versions of the sequential benchmark circuits s35932 and s38417 were compacted using the design procedures described above. Table 4.2 lists the number of gates, inputs, outputs, and flipflops of each circuit. The compaction functions were determined for test sets from the `Mintest` ATPG program [38] as well as for compact test sets derived from high-level functional models [39]. All test sets achieve 100% fault coverage for non-redundant single stuck-at faults.

Circuit	Test set	#Vectors $\|T\|$	l	#Outputs n	k	Literal count L_C	L_{SC}	Overhead (%)
c499	[39]	52	3	32	2	616	64	10.4
c880	[38]	16	16	26	5	703	277	39.4
c880	[39]	17	17	26	5	703	214	30.4
c1355	[39]	84	14	32	4	1032	197	19.1
c3540	[38]	84	84	22	7	2934	864	29.5
c6288	[38]	12	12	32	4	4800	266	5.5
c6288	[39]	12	12	32	4	4800	201	4.2
s35932	[38]	12	12	320	40	35181	2414	6.9
s35932	[38]	12	12	320	32	35181	2309	6.6
s35932[1]	[38]	12	12	320	4	35181	2542	7.2
s35932[2]	[38]	12	12	320	4	35181	2603	7.4
s38417	[38]	68	68	106	28	38790	3897	10.0
s38417	[38]	68	68	106	7	38790	4652	12.0

[1]Two-stage design according to Figure 4.9(a)
[2]Two-stage design according to Figure 4.9(b)

Table 4.3: Results for the synthesis of one-step compactors

4.4.1 One-Step Compaction

One-step compactors were derived as explained in the section above: $l + 1$ distinct symbolic values were assigned to the l fault-free test responses and to R^-. The logic synthesis tool NOVA, which is part of the SIS package [72], was used for mapping the symbolic values to binary values. The compactors were synthesized with SIS yielding compactors with the minimal number of outputs according to Theorem 4.1. SIS provides literal counts in order to estimate the area required by a circuit. Below, the estimation of the area overhead caused by a compactor is based on these literal counts.

Table 4.3 lists the results obtained for the investigated circuits. The first column lists the circuit names and the second column shows which test set was used. For each circuit the third and forth columns give the size of the test set $|T|$ and the number of distinct fault-free test responses l, respectively. The next two columns list the number of functional outputs n and the number of compacted outputs k. The seventh and eighth columns give L_C, which is the literal count of the investigated circuit, and L_{SC}, which is the literal count of the compactor designed according to the given test set. For sequential circuits the literal count L_C includes the area estimation of the scan flipflops. The very right column shows the estimated area overhead in percent, which is $\frac{L_{SC}}{L_C} \cdot 100$.

For the smaller circuits c499 and c880 the area overhead of the proposed method is high, since the compactor area is comparable to the area of the circuits.

Nevertheless, an one-step compactor for c499 with only 10.6% overhead can be designed, if the test set from [39] is used, which produces only three distinct fault-free test responses. This indicates, that there is a significant relationship between the number of distinct fault-free test responses and the area overhead caused by the compactor. This conclusion can also be derived from the results obtained for the circuit c1355. If the test set from [38] is used for c1355, then 64 distinct fault-free responses are obtained and the area overhead of the corresponding one-step compactor is over 130%. However, the

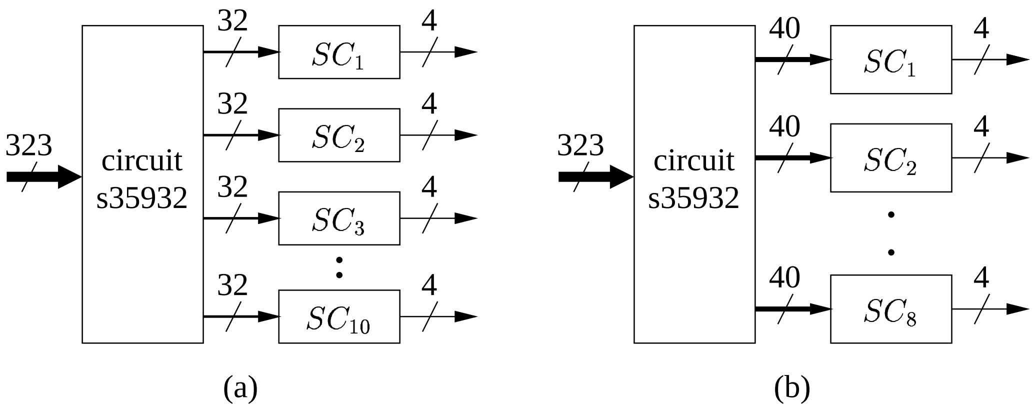


Figure 4.8: Output partitioning for s35932: (a) ten groups of 32 outputs each; (b) eight groups of 40 outputs each.

overhead reduces to 19%, when the test set from [39] with fourteen distinct fault-free test responses is used.

The high area overhead required for the compactor of the combinational circuit c3540 is a consequence of the large number of distinct fault-free test responses. In contrast, the area overhead of the compactor for c6288 is very low for test sets from both, [38] and [39]. This benchmark circuit, which is a 16-bit multiplier circuit, can be tested with small test sets consisting of twelve vectors.

Table 4.3 also lists the results for the full-scan versions of two sequential circuits s35932 and s38417. For both circuits the scan outputs were not compacted, so that only the functional outputs are connected to the compactor as shown in Figure 4.2.

The complexity of the synthesis procedure depends on the number of functional outputs n, which is equal to the number of compactor inputs. Thus, for the two sequential circuits with 106 and 320 functional outputs, the outputs were partitioned in such way, that SIS could synthesize space compactors for them in reasonable time. If at most a couple of hours of runtime is permitted, then the bound $n \leq 40$ on the number of inputs of one compactor is obtained by the experimental results.

For example, first of all the 320 functional outputs of s35932 were partitioned into ten groups of 32 outputs each. Then for each of these partitions a space compactor was synthesized with $k = 4$ compacted outputs each. As shown in Figure 4.8(a), this resulted in a total of 40 compactor outputs, yielding a compaction ratio of 8. The overhead caused by this compactor design is 6.9%. Alternatively, as shown in Figure 4.8(b), the 320 outputs can also be partitioned into eight groups of 40 outputs each. Then the total number of compacted outputs is 32, yielding a compaction ratio of 10. The overhead of the latter design is almost the same: 6.6%. The scan inputs and scan outputs of the full-scan version of s35932 are not shown in the figures.

In order to increase the compaction ratio further, a second stage of compaction can be added to the circuit s35932. In the second stage the 40 or 32 outputs from the first stage are further compacted to four outputs. This is the minimal number of outputs given by Theorem 4.1. Figure 4.9 shows the two-stage compaction designs for the circuit s35932. The scan inputs and scan outputs are omitted. Obviously, these designs are extensions of the one-stage compaction designs given in Figure 4.8. The overhead for the two-stage designs of Figure 4.9 (a) and (b) are 7.2% and 7.4%, respectively.

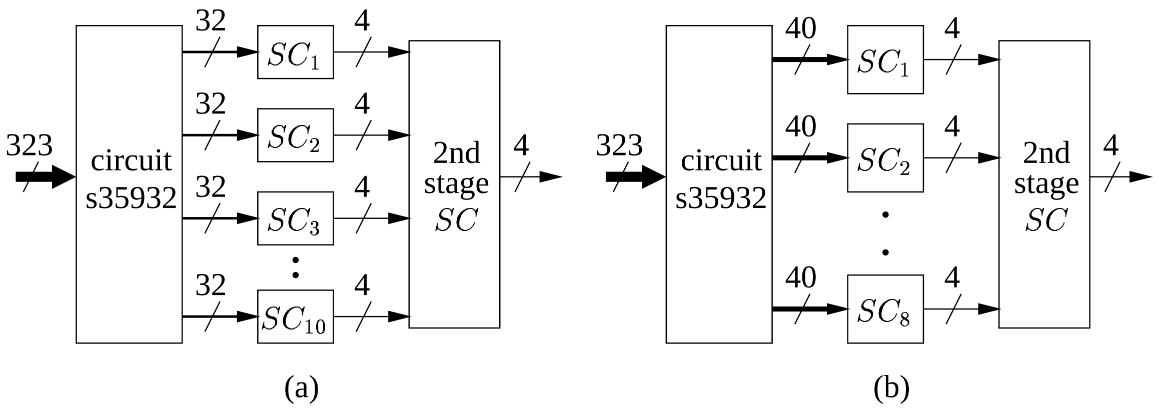


Figure 4.9: Two-stage, one-step compactor designs for s35932.

The 106 functional outputs of s38417 were partitioned in a similar way. Three groups consisting of 27 outputs each and a fourth group containing 25 outputs were formed. The outputs of each group were compacted to seven outputs each, yielding a total of 28 compacted outputs. The total area overhead of the four compactors is 10.1%. A second compaction stage was added in order to further reduce the number of compacted outputs. Then the number of compacted outputs is 7 which is the minimum according to Theorem 4.1. The area overhead increases only slightly to 12.0%.

4.4.2 Two-Step Compaction

Here experimental results are presented for two-step compactors, which were designed for the circuits listed in Table 4.2. As described above, in order to determine the first compaction step, the algorithm `GetTr0` was used to select functional outputs of the circuit. Then the transmission function Tr_1 of the second compaction step was derived from the algorithm `GetTr1`. The new output values assigned to the input sets of Tr_1 were obtained by using the logic synthesis tool `NOVA`, which is part of the `SIS` package [72]. Furthermore, `SIS` was used to synthesize the second compaction step.

Experimental results on two-step compactors based on orthogonal transmission functions are listed in Table 4.4. The first column gives the names of the investigated circuits. The second column shows, which test set was used. The following columns list for each circuit the size of the test set $|T|$, the number of distinct fault-free test responses l, the number of functional outputs n, and the number of compacted outputs k, respectively. The seventh column presents L_C, which is the literal count of the investigated circuit. For sequential circuits the literal count L_C includes the scan flipflops. The eighth column shows the literal count L_{SC} of the compactor, which was designed according to the given test set and which consists of the multiplexer, the feed-through lines, and the implementation of the second compaction function. The far right column shows the estimated area overhead in percent, which is derived from the literal counts as follows: $\frac{L_{SC}}{L_C} \cdot 100$.

The large number of functional outputs of the sequential circuits was partitioned as described above for one-step compactors. The lower bound k_{min} on the number of outputs given by Theorem 4.8 is achieved for each considered zero-aliasing two-step space compactor. For several cases the area overhead for two-step compaction is lower than that for one-step compaction, but in general the values of Table 4.3 and Table 4.4 are almost equal.

Circuit	Test set	#Vectors $\|T\|$	l	#Outputs n	k	Literal count L_C	L_{SC}	Overhead (%)
c499	[39]	52	3	32	2	616	77	10.4
c880	[38]	16	16	26	3	703	259	36.8
c880	[39]	17	17	26	3	703	204	29.0
c1355	[39]	84	14	32	3	1032	200	19.4
c3540	[38]	84	84	22	4	2934	792	27.0
c6288	[38]	12	12	32	2	4800	254	5.3
c6288	[39]	12	12	32	2	4800	201	4.2
s35932	[38]	12	12	320	20	35181	2385	6.8
s35932	[38]	12	12	320	16	35181	2290	6.5
s35932[1]	[38]	12	12	320	2	35181	2623	7.5
s35932[2]	[38]	12	12	320	2	35181	2478	7.0
s38417	[38]	68	68	106	16	38790	3666	9.5
s38417	[38]	68	68	106	4	38790	4553	11.7

[1]Two-stage design according to Figure 4.9(a)
[2]Two-stage design according to Figure 4.9(b)

Table 4.4: Results for the synthesis of two-step compactors

In Table 4.5 experimental results on the number of test patterns, that can be discarded in the first compaction step, are presented. The first and second columns give the names of the investigated circuits and the type of the underlying test set, respectively. The following columns list for each circuit the size of the test set $|T|$ and the number of distinct fault-free test responses l. The fifth column shows the upper bound on the size of T_1 given in Lemma 4.10. T_1 is the set of test vectors, which require only the second compaction step implementing the transmission function Tr_1. The sixth column gives the values of $|T_1|$ obtained by the experiments. The far right column lists the percentage of the test patterns, that can be discarded in the first compaction step. The percentage is computed by $\frac{|T_1|}{|T|} \cdot 100$.

For all circuits except c3540 the size of $|T_1|$ determined by the experiments equals the upper bound on this value. A significant fraction of input tests can be dropped for the circuits c432, c499, and s38417. If k is not as low as the lower bound k_{min} that is predicted by Theorem 4.8, then for even more test vectors only the second compaction step based on the transmission function Tr_1 is sufficient. The reason is, that a higher number of compacted outputs increases the number of input sets of Tr_1 and thus more input sets containing a single fault-free test response can be generated.

4.5 Summary

A space compaction approach for test responses of digital circuits is presented in this chapter. It does not use any knowledge about the internal structure of the circuit under test. The compactors are derived from the fault-free responses of a given, precomputed test set. For example, for intellectual property cores often only a test set and the corresponding output values are given and then such a space compactor can be used to access the test responses in parallel. Furthermore, the approach makes no assumption about an underlying error model.

Circuit	Test set	#Vectors $\|T\|$	l	Bound on $\|T_1\|$	Obtained $\|T_1\|$	Saving (%)
c499	[39]	52	3	4	4	7.7
c880	[38]	16	16	5	5	31.3
c880	[39]	17	17	5	5	29.4
c1355	[39]	84	14	6	6	7.1
c3540	[38]	84	84	10	9	10.7
c6288	[38]	12	12	0	0	0
c6288	[39]	12	12	0	0	0
s35932[1]	[38]	12	12	0	0	0
s35932[2]	[38]	12	12	0	0	0
s38417	[38]	68	68	11	11	16.2

[1]Two-stage design according to Figure 4.9(a)

[2]Two-stage design according to Figure 4.9(b)

Table 4.5: Results for the synthesis of two-step compactors

The application of the approach to a sequential circuit or to a large test sequence containing a small essential test set is described. The proposed method guarantees zero aliasing for all output errors tested by the given set of input patterns, so that any error at the functional outputs that is produced by the test set is propagated. A lower bound on the number of compacted outputs of these compactors is proven. The straightforward design approach for one-step compaction always achieves the maximum compaction ratio.

In order to further reduce the number of compacted outputs, the design of two-step compactors is introduced. Two distinct compaction functions are implemented by these compactors, and these two functions are applied in two time steps, which may double the test application time. The determination of the compaction functions based on orthogonal transmission functions is explained in detail.

In many cases the optimal compaction ratio is achieved for two-step compaction, too. If the ratio is not optimal, then for several test patterns often only one compaction step is required so that the test application time can be reduced.

Experimental results presented for several ISCAS benchmark circuits demonstrate the feasibility of the one-step and two-step compactors, even if the number of functional outputs of the circuit is large. Furthermore, in many cases the optimal compaction ratio is achieved for two-step compaction, too. If the ratio is not optimal, then for several test patterns often only one compaction step is required, so that the test application time can be reduced. The synthesis of the compactors show, that both techniques are best suited to circuits, whose test sets produce only a small number of distinct fault-free test responses, otherwise the area overhead caused by the compactor will be very large. The required area is almost the same for one-step compaction and two-step compaction.

5 Contributions and Future Work

In this chapter, the major contributions of this work are reviewed. The second section gives brief prospects for future work.

5.1 Contributions

As the high integration of circuits results in increasing numbers of inputs, outputs and gates, the size of test sets becomes larger so that bit-by-bit comparison of the test responses is not feasible. Output space compaction is a simple method, which reduces the quantity of test response data. This work presents new space compaction techniques for testing or concurrent checking of digital circuits.

If the circuits are given by a netlist of gates, then an analysis of the circuit structure can be used to obtain information about fault propagation in the circuit. Based on this information, the design of the space compactor can be suited to the circuit.

In this thesis, compactor designs, which use the concept of structural space compaction are described and investigated:

- Several analysis algorithms for the design of space compactors are described. The introduced analysis algorithms are investigated for the first time. The algorithms include simple approximations of probabilities of fault propagation and observation.
- The complexity of each algorithm is polynomial. In particular, for each algorithm the complexity is linear with respect to the number of gates and at most quadratic with the number of circuit outputs. Therefore, these algorithms are applicable to large circuits.
- For 22 combinational benchmark circuits the values computed by the analysis algorithms are compared to each other and to exact values. The best results are obtained for the analysis algorithms based on approximated probabilities.
- In order to further improve the analysis based on approximated probabilities, several modifications of the analysis algorithms are discussed.
- An algorithm for deriving the compaction function from the analysis results is described. The complexity of the heuristic algorithm is polynomial.
- For the ten combinational circuits of the ISCAS ’85 benchmark suite experimental results are given for deterministic testing, concurrent checking and pseudo random testing. The reduction of fault coverage was experimentally determined for the fault models of single stuck-at faults and transition faults.
- Comparisons with other approaches show, that for combinational circuits the proposed linear space compactor design achieves high compaction ratio and reduces fault masking.

- The approach for combinational circuits can be extended in such a way, that it can be applied to sequential circuits which provide no access to the state values. It is the first structural approach for sequential circuits, which takes into account fault propagation from faulty states to the circuit outputs.

- The design procedure for space compactors is applicable to large sequential circuits. The complexity of the design procedure is polynomial: Again, it is linear with respect to the number of gates and quadratic with the number of circuit outputs.

- In order to improve the design of space compactors, several modifications of the analysis of the sequential circuit are proposed.

- For 20 sequential circuits of the ISCAS '89 benchmark suite experimental results are given, which were obtained for concurrent checking and pseudo random testing. The reduction of fault coverage was experimentally determined for the fault models of single stuck-at faults and transition faults.

Furthermore, in this work a space compaction design is presented, which requires no information about the internal structure of the circuit under test or about the underlying fault model. The basic concept is to use the knowledge of a precomputed compact test set and the corresponding fault-free test responses to design a zero-aliasing space compactor. This means that the compactor causes no fault masking for all errors, which are produced by the precomputed test set.

The main contributions are:

- A proof for the lower bound on the number of compacted outputs is given for zero-aliasing compactors using a single compaction step.

- A procedure to design zero-aliasing space compactors with a minimal number of outputs is explained in detail.

- In order to further reduce the number of compacted outputs, the concept of two-step compactors is introduced.

- The design of two-step compactors is based on orthogonal transmission functions. Orthogonal transmission functions and the corresponding algorithms for generating the functions are described in detail.

- A lower bound on the number of compacted outputs of two-step compactors has been proven.

- It has been shown, that two-step compactors allow to trade off testing time with the compaction ratio.

- Experimental results show, that both approaches, one-step and two-step compaction, are feasible. Furthermore, the area overhead caused by these compactors is moderate if the number of distinct fault-free test responses is small.

Parts of this work were published in the proceedings of the Workshop on Test Methods and Reliability of Circuits and Systems 1998 [80], of the VLSI Test Symposium 1998 [79], of the Workshop on Implementing Automata [78], of the Symposium on Defect and Fault Tolerance in VLSI Systems 1999

[77], of the VLSI Test Symposium 2000 [76], and of the International On-line Testing Workshop 2000 [24].

Other publications of the author of this thesis are:

- M. Seuring, “Built-In Self Test mit Multi-Mode Scannable Memory Elementen,” in *Proc. Workshop on Test Methods and Reliability of Circuits and Systems,* pp. 22–25, 1999;
- A. Singh, E. Sogomonyan, M. Gössel, and M. Seuring, “Testability Evaluation of Sequential Designs Incorporating the Multi-Mode Scannable Memory Element,” in *Proc. Int. Test Conf.,* pp. 227–235, 1999;
- D. Das, N. A. Touba, M. Seuring, and M. Gössel, “Low Cost Concurrent Error Detection Based on Modulo Weight-Based Codes,” in *Proc. Int. On-line Testing Workshop,* pp. 171–176, 2000.

5.2 Future Work

In this section, extensions of the results of this work are suggested and discussed. In particular, proposals which may improve the approach of structural space compaction are given.

The structural approaches for linear space compaction are based on algorithms, which analyze the structure of the circuit under test. If the design procedure should be applicable to large circuits, then the complexity of the algorithms must be limited and the computation of exact values is prevented. Therefore, the circuit analysis can be done in various ways and it is likely, that further improvements for the analysis algorithms can be found.

Since the best results were obtained for the analysis algorithms based on approximated probabilities, future work should investigate improvements for this algorithms. In Section 2.2.8, two modifications are proposed, which target more accurate results of the circuit analysis.

The first modification concerns the approximation of signal probabilities, which can be computed using a more sophisticated algorithm introduced in [52]. Any other algorithm, which determines more accurate signal probabilities than the standard algorithm can be included in the analysis of the circuit structure. Future work may investigate, for example, whether such algorithms can be derived from several testability measurements, and furthermore, whether the experimental results improve if the algorithms are used for the design of space compactors.

In this work, the results obtained for sequential circuits are not as good as those for combinational circuits. Thus, an analysis which takes into account the set of reachable states of the sequential circuit under test as described in Section 3.2.3 may improve the analysis results. Also, the model of the iterative array of the combinational parts can be used to approximate the signal probabilities through several time frames. For example, the initial state of the sequential circuit can be taken into account, if the signal probabilities of the state of the first time frame are set corresponding to the initial state.

Finally, future investigation may focus on the first suggestion described in Section 2.2.8: Small circuit parts of the circuit under test which contain reconvergent paths can be replaced by complex gates. Then the analysis uses the corresponding exact propagation probabilities of the complex gates. The analysis algorithm could be implemented so that the maximal number of input lines of circuit parts, which are replaced by complex gates, is an input parameter of the algorithm. Then this parameter can be used to trade off the runtime of the algorithm with the accuracy of the results. However, the runtime can exponentially increase with respect to the maximal number of input lines allowed for circuit parts which are replaced.

The suggestions above should lead to improved results for the analysis of the structure of both, combinational and sequential circuits. A linear space compactor, whose design is based on an improved analysis algorithm, should provide a higher space compaction ratio combined with a lower probability of fault masking.

A Simulator MINOS

This appendix briefly describes the fault simulator `Minos` which was implemented in order to facilitate the experimental investigations of this thesis. Its first version was restricted to the simulation of single stuck-at faults and later it was extended in such a way, that also transition faults can be simulated. `Minos` which is implemented in C++ can simulate both combinational and sequential circuits. It is based on single pattern and parallel fault propagation.

In order to simulate the propagation of faults in parallel, `Minos` uses two-valued parallel simulation which was introduced by Seshu [73]. Starting with a single input pattern, which is applied to the circuit inputs, the simulator proceeds in a topological order to the circuit outputs and computes for each line the good value and the propagated faults. The data structure which represents fault effects at a single line is an array of bits, where each bit corresponds to a single fault. If a bit of this data structure is set then the corresponding fault is observable at this line.

The faults which are observable at the output of a gate can be derived from the faults observable at the inputs of the gate using operations of the set theory [1]. Since the sets of observable faults are stored in bit arrays, the computation of observable faults can be implemented by bitwise operations, so that the runtime is reduced.

In an initial step `Minos` determines equivalent faults, so that only collapsed faults instead of all faults can be simulated. If a sufficiently large number of faults is already detected, `Minos` performs fault dropping in the following way: All faults which are already detected are removed from the bit arrays. The remaining faults are re-assigned to the bits in such a way, that the required size of the bit arrays which store the observable faults is reduced.

Since `Minos` simulates single input patterns, transition faults can easily be handled. Only if the good value of a line has been computed and this new value is different from the old one, a transition fault can be excited at the line. The propagation of a excited transition fault is computed in almost the same way as the propagation of a single stuck-at fault.

The bit arrays of the circuit outputs determine the detectable faults. If a bit of such a array is set, then the corresponding fault is detected. Furthermore, the bitwise boolean sum of all bit arrays of the circuit outputs determine the faults, which can be detected by a linear space compactor consisting of a single parity tree. The detected faults at the outputs of other space compactor designs can analogically be computed.

This is utilized by `Minos` as follows: `Minos` is able to use the data structure of the parity groups obtained by the procedures which are presented in the second and third chapter for the design of space compactors. Therefore, during one simulation process, the simulator `Minos` can determine the faults, which are detected at the functional outputs of the circuit, and also the faults, which are observable at the outputs of one or more linear space compactors. As mentioned above, only additional bitwise operations, which sum up the bit arrays of the circuit outputs according to the parity groups, are required.

Bibliography

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, “Digital Systems Testing and Testable Design,” IEEE Press, 1990.

[2] M. J. Batek and J. P. Hayes, “Test Set Preserving Logic Transformation,” in *Proc. Design Automation Conference,* pp. 454–458, 1992.

[3] N. Benowitz, D. F. Calhoun, G. E. Anderson, J. E. Bauer, and C. T. Joeckel, “An Advanced Fault Isolating System for Digital Logic,” in *IEEE Transactions on Computers,* vol. C-24, pp. 489–497, 1975.

[4] J. M. Berger, “A Note on Error Detecting Codes for Asymmetric Channels,” in *Information and Control,* vol. 4, pp. 68–73, 1961.

[5] B. B. Bhattacharya, A. Dmitriev and M. Gössel, “Zero-Aliasing Space Compaction Using a Single Period Output and Its Application to Testing of Embedding Cores,” in *Proc. Int. Conf. VLSI Design*, pp. 382–387, 2000.

[6] S. Bhatia, T. Gheewala, and P. Varma, “A Unifying Methodology for Intellectual Property and Customs Logic Testing,” in *Proc. Int. Test Conf.,* pp. 639–648, 1996.

[7] D. Bhattacharya, “Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit,” in *Proc. VLSI Test Symposium*, pp. 8–14, 1998.

[8] Mario Blaum, “Codes for Detecting and Correcting Unidirectional Errors,” IEEE Computer Society Press, 1993.

[9] P. Böhlau, “Zero Aliasing Compression Based on Groups of Weakly Independent Outputs in Circuits with High Complexity for Two Fault Models,” in *Lect. Notes in Comp. Science,* vol. 852, pp. 289–306, 1994, Springer.

[10] D. Brand, “Verification of Large Synthesized Designs,” in *Proc. Int. Conf. on Computer Aided Design,* pp. 534–537, 1993.

[11] K. M. Butler and M. R. Mercer, “Quantifying Non-Target Defect Detection by Target Fault Sets,” in *Proc. European Test Conf.,* pp. 91–100, 1991.

[12] F. Brglez, D. Bryan, and K. Kozminski, “Combinational Profiles of Sequential Benchmark Circuits,” in *Proc. IEEE Int. Symp. Circuits and Systems,* pp. 1929–1934, 1989.

[13] F. Brglez and H. Fujiwara, “A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran,” in *Proc. IEEE Int. Symp. Circuits and Systems,* 1985.

[14] F. Brglez, P. Pownall, and R. Hum, “Applications of Testability Analysis: From ATPG to Critical Delay Tracing,” in *Proc. Int. Test Conf.,* pp. 705-712, 1984.

[15] R. E. Bryant, “Graph-Based Algorithms for Boolean Function Manipulation,” in *IEEE Transactions on Computers,* vol. C-35, pp. 677–691, 1986.

[16] K. Chakrabarty, “Zero-Aliasing Space Compaction Using Linear Compactors with Bounded Overhead,” in *IEEE Transactions on Computer-Aided Design,* vol. 17, pp. 452–457, 1998.

[17] K. Chakrabarty and J. P. Hayes, “Efficient Test Response Compression for Multiple-Output Circuits,” in *Proc. Int. Test Conf.,* pp. 501–510, 1994.

[18] K. Chakrabarty and J. P. Hayes, “Test Response Compaction Using Multiplexed Parity Trees,” in *IEEE Transaction on Computer-Aided Design,* vol. 15, pp. 1399–1408, 1996.

[19] K. Chakrabarty and J. P. Hayes, “Zero-Aliasing Space Compaction Using Multiple Parity Signatures,” in *IEEE Transactions on VLSI Systems,* vol. 6, pp. 309–313, 1998.

[20] K. Chakrabarty, B. T. Murray and J. P. Hayes, “Optimal Zero-Aliasing Space Compaction of Test Responses,” in *IEEE Transactions on Computers,* vol. 17, pp. 1171–1187, 1998.

[21] R. Chandramouli and S. Pateras, “Testing Systems on a Chip,” in *IEEE Spectrum,* pp. 42–47, Nov. 1996.

[22] V. Chickermane, E. M. Rudnick, P. Banerjee, and J. H. Patel, “Non-Scan Design-For-Testability Techniques for Sequential Circuits,” in *Proc. Design Automation Conference,* pp. 236–241, 1993.

[23] D. Das and N. A. Touba, “Weight-Based Codes and Their Application to Concurrent Error Detection of Multilevel Circuits,” in *Proc. VLSI Test Symposium,* pp. 370–376, 1999.

[24] D. Das, N. A. Touba, M. Seuring, and M. Gössel, “Low Cost Concurrent Error Detection Based on Modulo Weight-Based Codes,” in *Proc. Int. On-line Testing Workshop,* pp. 171–176, 2000.

[25] S. R. Das, H. T. Ho, W.-B. Jone, and A. R. Nayak, “An Improved Output Compaction Technique for Built-In Self Test in VLSI Circuits,” in *Proc. Int. Conf. VLSI Design,* pp. 403–407, 1994.

[26] U. Dave and J. H. Patel, “A Functional Test Generation Method Using Two-Level Representations,” in *Proc. Design Automation Conference,* pp. 722–725, 1989.

[27] K. De, C. Natarajan, D. Nair, and P. Banerjee, “RSYN: A System for Automated Synthesis of Reliable Multilevel Circuits,” in *IEEE Transactions on VLSI Systems,* pp. 186–195, 1994.

[28] E. B. Eichelberger and E. Lindbloom, “Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test,” in *IBM J. Research and Development,* vol. 27, pp. 265–272, 1983.

[29] R. D. Eldred, “Test Routines Based on Symbolic Logical Statements,” in *J. Assoc. Comput. Mach.,* vol. 6 (1), pp. 33–36, 1959.

[30] J. R. Fox, “Test-point Condensation in the Diagnosis of Digital Circuits,” in *Proc. IEE,* pp. 89–94, 1977.

[31] H. Fujiwara, “Logic Testing and Design for Testability,” MIT Press, Cambridge, 1985.

[32] H. Fujiwara and A. Yamamoto, “Parity-Scan Design to Reduce the Cost of Test Application,” in *IEEE Transactions on Computer-Aided Design,* vol. 12, pp. 1604–1611, 1993.

[33] M. R. Garey and D. S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-completeness,” W. H. Freeman and Company, New York, 1979.

[34] I. Ghosh, N. K. Jha, and S. Dey, “A Low Overhead Design for Testability and Test Generation Technique for Core-Based Systems”, in *Proc. Int. Test Conf.,* pp. 50–59, 1997.

[35] L. H. Goldstein, “Controllability/Observability Analysis of Digital Circuits,” in *IEEE Transactions Circuits Systems,* vol. 26(9), pp. 685–693, 1979.

[36] M. Gössel and E. S. Sogomonyan, “Design of Self-Testing and On-Line Fault Detection Combinational Circuits with Weakly Independent Outputs,” in *J. Electronic testing: Theory and Applications,* vol. 4, pp. 267–281, 1993.

[37] M. Gössel, E. S. Sogomonyan and A. Morosov, “A New Totally Error Propagating Compactor for Arbitrary Cores with Digital Interfaces,” in *Proc. VLSI Test Symposium,* pp. 49–56, 1999.

[38] I. Hamzaoglu and J. H. Patel, “Test Set Compaction Algorithms for Combinational Circuits,” in *Proc. Int. Conf. on Computer Aided Design,* pp. 283–289, 1998.

[39] M. C. Hansen and J. P. Hayes, “High-level Test Generation Using Physically-Induced Faults,” in *Proc. VLSI Test Symposium,* pp. 20–28, 1995.

[40] J. P. Hayes and A. D. Friedman, “Test Point Placement to Simplify Fault Detection,” in *IEEE Transactions on Computers,* vol. C-23, pp. 727–735, 1974.

[41] O. H. Ibarra and S. K. Sahni, “Polynomially Complete Fault Detection Problems,” in *IEEE Transactions Computers,* vol. C-24, pp. 242–249, 1975.

[42] V. Immaneni and S. Raman, “Direct Access Test Scheme – Design of Core Cells for Embedded ASICs,” in *Proc. Int. Conf. on Computer Aided Design,* pp. 488–492, 1990.

[43] International Technology Working Group, “The International Technology Roadmap for Semiconductors: 1999,” Sematech, 1999.

[44] A. Ivanov, B. K. Tsuji, and Y. Zorian, “Programmable BIST Space Compactors,” in *IEEE Transactions Computers,* vol. 45, pp. 1393–1404, 1996.

[45] S. K. Jain and V. D. Agrawal, “Statistical Fault Analysis,” in *IEEE Design Test Comp.,* vol. 2, pp. 38–44, 1985.

[46] N. K. Jha and S. Wang, “Design and Synthesis of Self-Checking VLSI Circuits,” in *IEEE Transactions on Computer-Aided Design,* vol. 12, pp. 878–887, 1993.

[47] W.-B. Jone and S. R. Das, “Space Compaction Method for Built-In Self Testing of VLSI Circuits,” in *Int. Journal of Computer-Aided Design,* vol. 3, pp. 309–322, 1991.

[48] M. Karpovsky and P. Nagvajara, “Optimal Time and Space Compression of Test Responses for VLSI Devices”, in *Proc. Int. Test Conf.,* pp. 523–529, 1987.

[49] Z. Kohavi et al., “Detection of Multiple Faults in Combinational Logic Networks,” in *IEEE Transactions Computers,* vol. C-21, pp. 556–568, 1972.

[50] R. Krieger, B. Becker, and R. Sinkovic, “A BDD-based Algorithm for Computation of Exact Fault Detection Probabilities,” in *Proc. Int. Symp. on Fault-Tolerant Comp.,* pp. 186–195, 1993.

[51] R. Krieger, B. Becker, and C. Ökmen, “OBDD-based Optimization of Input Probabilities for Weighted Random Pattern Generation,” in *Proc. Int. Symp. on Fault-Tolerant Comp.,* pp. 120–129, 1995.

[52] B. Krishnamurthy, and I. G. Tollis, “Improved Techniques for Estimating Signal Probabilities,” in *IEEE Transactions Computers,* vol. 38(7), pp. 1041–1045, 1989.

[53] Y. K. Li, and J. P. Robinson, “Space Compaction Methods with Output Data Modification,” in *IEEE Transactions on Computer-Aided Design,* vol. 6, pp. 290–294, 1987.

[54] S. Ma, I Shaik, and R. S. Fetherston, “A Comparison of Bridging Fault Simulation Methods,” in *Proc. Int. Test Conf.,* pp. 587–595, 1999.

[55] M. Marhöfer, “An Approach to Modular Test Generation Based on the Transparency of Modules,” in *Proc. IEEE CompEuro 87,* pp. 403–406, 1987.

[56] A. Morozov, Va. V. Saposhnikov, Vl. V. Saposhnikov, M. Gössel, “New Self-Checking Circuits by Use of Berger-Codes,” in *Proc. Int. On-line Testing Workshop,* pp. 141–146, 2000.

[57] V. Moshanin, V. Ocheretnij, and A. Dmitriev, “The Impact of Logic Optimization on Concurrent Error Detection,” in *Proc. Int. On-line Testing Workshop,* pp. 81–84, 1998.

[58] B. T. Murray, “Hierarchical Testing with Precomputed Tests,” in *Technical Report CSE-TR-245-95,* Dept. of Electrical Engineering and Computer science, University of Michigan, 1995.

[59] B. T. Murray and J. P. Hayes, “Testing IC’s: Getting to the Core of the Problem,” in *IEEE Computer,* vol 29 (11), pp. 32–38, 1996.

[60] K. P. Parker, and E. J. McCluskey, “Probabilistic Treatment of General Combinational Networks,” in *IEEE Transactions Computers,* vol. 24(6), pp. 668–670, 1976.

[61] B. Pouya and N. A. Touba, “Synthesis of Zero-Aliasing Elementary-Tree Space Compactors,” in *Proc. VLSI Test Symposium*, pp. 70–77, 1998.

[62] J. Rajski and J. Vasudevamurthy, “Testability Preserving Transformations in Multi-Level Logic Synthesis,” in *Proc. Int. Test Conf.,* pp. 265–273, 1990.

[63] I. M. Ratiu, A. Sangiovanni-Vincentelli, and D. O. Pederon, “VICTOR: A Fast VlSI Testability Analysis Program,” in *Proc. Int. Test Conf.,* pp. 397–401, 1982.

[64] S. M. Reddy, K. K. Saluja, and M. G. Karpovsky, “A Data Compression Technique for Built-In Self-Test,” in *IEEE Transactions Computers,* vol. 37, pp. 1151–1156, 1988.

[65] J. P. Roth, “Diagnosis of Automata Failures: A Calculus and a Method,” in *IBM J. Research and Development,* vol. 10, pp. 278–291, 1966.

[66] E. Rudnick, V. Chickermane, and J. H. Patel, “An Observability Enhancement Approach for Improved Testability and At-Speed Test,” in *IEEE Transactions on Computer-Aided Design,* vol. 13, pp. 1051–1056, 1994.

[67] K. K. Saluja and M. Karpovsky, “Testing Computer Hardware through Data Compression in Space and Time,” in *Proc. Int. Test Conf.,* pp. 83–88, 1983.

[68] Va. V. Saposhnikov, A. Morosov, Vl. V. Saposhnikov, M. Gössel, “Design of Self-Checking Unidirectional Combinational Circuits with Low Area Overhead,” in *Proc. Int. On-line Testing Workshop,* pp. 56–67, 1996.

[69] Va. V. Saposhnikov, A. Morosov, Vl. V. Saposhnikov, M. Gössel, “A New Design Method for Self-Checking Unidirectional Combinational Circuits,” in *Journal of Electronic Testing: Theory and Applications,* vol. 12, pp. 41–53, 1998.

[70] J. Savir, “Scan Latch Design for Delay Test,” in *Proc. Int. Test Conf.,* pp. 426–453, 1997.

[71] J. Savir, “Shrinking Wide Compressors,” in *IEEE Transactions on Computer-Aided Design,* vol. 14, pp. 1379–1387, 1995.

[72] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli, “SIS: A System for Sequential Circuit Synthesis,” *Memorandum Nr. UCB/ERL M92/41,* Electronics Research Laboratory, 1992.

[73] S. Seshu, “On an Improved Diagnosis Program,” in *IEEE Transactions on Electronic Computers,* vol. EC-12, pp. 76–79, 1965.

[74] S. C. Seth, L. Pan, and V. D. Agrawal, “PREDICT - Probabilistic Estimation of Digital Circuit Testability,” in *Proc. Int. Symp. on Fault-Tolerant Comp.,* pp. 220-225, 1985.

[75] R. M. Sedmak, “Design for Self-Verification: An Approach for Dealing with Testability Problems in VLSI-based Design,” in *Proc. IEEE Test Conf.,* pp. 112–120, 1979.

[76] M. Seuring, and K. Chakrabarty, “Zero-Aliasing Space Compaction for IP Cores Using Orthogonal Transmission Functions,” in *Proc. VLSI Test Symposium,* pp. 354–361, 2000.

[77] M. Seuring, and M. Gössel, “A Structural Approach for Space Compaction for Sequential Circuits,” in *Proc. Symp. on Defect and Fault Tolerance in VLSI Systems,* pp. 286–293, 1999.

[78] M. Seuring, and M. Gössel, “A Structural Approach for Output Compaction of Sequential Automata Implemented as Circuits,” in *Proc. Workshop on Implementing Automata,* pp. XVI 1–6, 1999.

[79] M. Seuring, M. Gössel, and E. Sogomonyan, “A Structural Approach for Space Compaction for Concurrent Checking and BIST,” in *Proc. VLSI Test Symposium,* pp. 354–361, 1998.

[80] M. Seuring, M. Gössel, and E. Sogomonyan, "Ein strukturelles Verfahren zur Kompaktierung von Schaltungsausgaben für On-line-Fehlererkennung und Selbsttest," in *Proc. Workshop on Test Methods and Reliability of Circuits and Systems,* pp. 22–25, 1998.

[81] E. S. Sogomonyan, "Design of Built-in Self-Checking Monitoring Circuits for Combinational Devices," in *Automation and Remote Control,* vol. 35(2), pp. 280–289, 1974.

[82] J. E. Stephenson, and J. Grason, "A Testability Measure for Register Transfer Level Digital Circuits," in *Proc. Int. Symp. on Fault-Tolerant Comp.,* pp. 101–107, 1976.

[83] S. Tarnick, "Bounding Error Masking in Linear Output Space Compression Schemes," in *Proc. Asian Test Symp.,* pp. 27–32, 1994.

[84] H. J. Wunderlich, "Probabilistische Verfahren für den Test hochintegrierter Schaltungen," in *Informatik-Fachberichte 140,* Springer-Verlag, 1987.

[85] Y. Zorian, "Test Requirements for Embedded Core-Based Systems and IEEE P1500," in *Proc. Int. Conf. on Computer Aided Design,* pp. 191–199, 1997.

[86] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," in *Proc. Int. Conf. on Computer Aided Design,* pp. 130–143, 1998.