Compact Test Set Generation for Test Compression-based Designs

Stephan Eggersglüß*†

*Institute of Computer Science
University of Bremen
28359 Bremen, Germany
segg@informatik.uni-bremen.de

†Cyber-Physical Systems
DFKI GmbH
28359 Bremen, Germany

***Abstract*—The manufacturing test is an important and expensive part of the overall electronic design flow. A main cost factor is the steadily increasing test data volume. Modern designs typically use extra hardware, i.e. test compression hardware, to compress the scan patterns to save test data volume. However, this imposes constraints on the pattern generation process. A high number of unspecified bits is typically needed to compress a test pattern successfully. In this paper, a compact test set generation technique is proposed for test compression-based designs. The proposed technique is based on a fully-specified test set and uses test vector decomposition, multiple-fault-detection and atomic vector ordering in order to build a highly compact test set with a guaranteed percentage of unspecified bits. Experimental results on benchmark and industrial circuits show that the approach is able to achieve a significant pattern reduction compared to previous approaches. Furthermore, it is shown that a higher compaction of the basis test set leads to a higher compaction of the resulting partially-specified test set.**

I. Introduction

The task of the manufacturing test in the EDA design flow is to filter out defective devices. The increasing size of today's circuits leads to steadily increasing test costs. One heavy cost factor is the large test data volume needed to test the circuits. The manufacturer typically undertakes large efforts to decrease the test data volume and, consequently, decrease the test costs. Classically, static or dynamic test compaction algorithms [1]–[5] are used for traditional scan *Design-For-Testability* (DFT) to produce less test patterns by, at the same time, high fault coverage with respect to a target fault model. However, this is not sufficient anymore for today's large *System-on-Chip* (SoC) designs.

Therefore, test compression approaches, e.g [6]–[12], have been developed. Here, the test data is stored on the tester in compressed form. However, this also necessitates the addition of extra on-chip hardware. The additional hardware is used to decompress the test stimulus from the tester feeding the scan chains as well as compacting the responses coming from the scan chains.

The test compression algorithms make use of the circumstance that the tests generated by ATPG tools contain unspecified values, i.e. X-values. Typically only a small portion, e.g. 1% to 5% of the input bits [11], is specified by 0 or 1. The remaining bits are don't cares and used to compress the tests. However, with respect to the compression technique used as well as to the targeted compression ratio, there is a limit of the number of specified bits in one test. If the number of specified bits exceeds a certain threshold, the probability that the compression algorithm cannot compress the test increases drastically. As a result, such a test is likely to be dropped which decreases fault coverage. For instance, if an LFSR-based test compression technique is used, the length l of the LFSR should be chosen such that $l = s_{\text{max}} + 20$ where s_{max} is the maximum number of specified bits in any test [6].

Traditional compaction techniques for ATPG are not well suited for such test compression-based designs, since these approaches achieve high compaction by basically specifying don't care bits in the test to detect additional faults. However, a large number of don't care bits is needed to achieve a high compression ratio as explained above. Therefore, dynamic compaction approaches have to be modified that they can be applied to test compression-based designs. A straight-forward way to incorporate the characteristics of a test compression-based design into a compaction flow is to apply a regular dynamic compaction approach until a certain limit of care bits or a more sophisticated threshold is reached.

More elaborated approaches have been developed. Tailored ATPG approaches were proposed in [13], [14] in which specific compression or BIST architectures are incorporated as constraints into the ATPG process to decrease test data volume. However, the computational complexity grows significantly especially for larger circuits if these additional constraints are enforced. Therefore, these so-called one-step ATPG approaches are particularly not feasible for linear decompressors which provide greater compression [11]. Recently, a new test compression method was proposed which includes the generation of test templates to guide ATPG to produce highly compressible test cubes [15].

Other ATPG-based techniques have been introduced which suffer from the increase in ATPG complexity and long run times. The approach in [16] uses formal techniques to produce test cubes for given single paths which are proven to be minimal in the number of assignments. However, no merging algorithm is given. An optimization procedure is proposed in [17] which compacts a given set of sensitized long paths and relaxes the generated tests by lifting.

ATPG-independent approaches have been introduced to avoid the increase of the ATPG complexity. These approaches take a pre-generated test set as input and generate a new test set in which the number of unspecified values is increased while keeping the same fault coverage. They can be roughly categorized in two different classes. The first class [18]–[21] uses test relaxation techniques and does not change the number of tests in the test set. Care bits not necessary for single fault detection are determined and unspecified by path tracing [18], [19], using hierarchical fault-compatibility [21] or implication graphs [20]. The effectiveness of these approaches is limited due to the lack of flexibility.

The second class [22]–[24] is more flexible and changes the number of tests in the initial test set. By this, the percentage of unspecified bits can be increased compared to approaches of the first category. However, this comes typically at the cost of a pattern count increase. This is acceptable for test compression-based designs since the increase of unspecified bits typically results in the possibility of a higher compression ratio.

The approach described in [22] uses *Test Vector Decomposition* (TVD) [18] where a test is decomposed into its atomic components. A parent pattern t is iteratively split into several test vectors T such that the test vectors in T detect all faults that are detected by t but each with fewer specified bits. A similar technique is also used in [23]. Here, each so-called bottleneck vector whose number of care bits is above a certain threshold is replaced by a set of test vectors meeting the requirements. In order to find a small number of test vectors as replacement, tests of essential faults are merged first. The technique in [24][1] is based on test stripping [20]. Similar to [23], tests whose number of care bits is above a threshold are splitted using pattern duplication [22].

The new technique proposed in this paper introduces a novel compaction methodology for test compression-based designs and uses an extended TVD procedure. A fully-specified test set T_f generated by regular ATPG tools is transformed into a partially-specified test set T_p. In the test set T_p, the number of care bits does not exceed a certain parameterized threshold to meet the test compression requirements. Several new techniques are introduced to keep the overall number of tests in the test set low to decrease the test data volume. A main difference to previous approaches is that the previous approaches operate more in a pattern-focused manner, e.g. pattern duplication or splitting. The proposed approach does not use the pattern structures of T_f but constructs the test set T_p completely new from scratch based on the atomic test vectors.

- A fast greedy test cube merging algorithm for atomic test vectors is introduced. The merging algorithm gains its effectiveness due to the following techniques.
- Diversity-based atomic vector creation – In order to produce a divers set of atomic test vectors, a new dynamically adjusted test relaxation procedure is introduced.
- Multiple detection – In a complete test set, most faults are detected multiple times. This can be leveraged during test set construction to increase the merging capabilities.
- Fault and atomic vector ordering – The proposed merging algorithm uses orderings based on the number of detections and the size of the atomic vectors.

Experimental results on benchmark as well as industrial circuits show that a fully-specified test set can be transformed into a partially-specified test set in a fast and effective way. An interesting observation is that the size of the resulting test set T_p strongly depends on the size of the basic test set T_f. Generally, a higher compaction of T_f leads to a smaller pattern count of T_p. This indicates that the application of regular test compaction techniques is also advantageous for test compression-based designs.

Algorithm 1 Atomic Vector Extraction

```
Input: Set of necessary values $N$, Simulated Test $t$
Output: Set of input assignments $t^a$
while $N$.empty() == false do
  NecessaryValue $n$ = $N$.getAndDeleteElement();
  Gate $g$ = $n$.gate;
  Value $v$ = $n$.value;
  if $g$.isInput() then
    $t^a$.add($g$,$v$); continue;
  end if
  if $v$ == ($g$.controllingValue() $\oplus$ $g$.isInvertingGate()) then
    Gate $p$ = ChooseOneInputWithControllingValue();
    $N$.insert($p$);
  else
    $N$.insert($g$.allInputs())
  end if
end while
```

The paper is structured as follows. Section II reviews the existing TVD technique as well as the newly proposed extensions. Section III gives the overall merging algorithm and introduces the ordering techniques. Experimental results are given in Section IV and conclusions are drawn in Section V.

II. Test Vector Decomposition

Test Vector Decomposition (TVD) is used for a given (fully- or partially-specified) test vector t and a set of Faults F_t detected by t. The task is to split the vector t into its atomic components $t^a_1, \ldots, t^a_n$. Each atomic component t^a_j (or atomic test vector) detects at least one fault $f_i \in F_t$ but contains only a small set of assignments necessary to detect f_i, i.e. they are highly unspecified.

The advantage of using TVD is that the highly unspecified atomic test vectors provide a great flexibility to re-merge the tests to create a test vector with more unspecified bits than before [19] or even reduce the pattern count if all atomic vectors of one test can be merged into other patterns [18].

After fault simulation, the propagation path of the fault f from the fault site to an observation point is known. The underlying idea is that the complete input cone is not necessary to justify the necessary values, i.e. values on the side inputs of the propagation path as well as on the fault site. Controlling values are leveraged to trace the necessary assignments to the inputs to obtain a small set of values. Algorithm 1 shows an example implementation of this technique.

The approach proposed in this paper uses the TVD technique to create atomic test vectors from the given test set. However, compared to previous approaches, the proposed technique focuses on *Multiple Detection* and uses a dynamic metric to guide the TVD procedure which is described after the general flow description. The following TVD flow is used to decompose test set T_f:

1) Fault simulation for each $t \in T_f$ is performed. Fault dropping is generally disabled to support multiple detection of faults.
2) Once, a fault f is detected by fault simulation, the atomic test vector t^a_f is extracted by the procedure given in Algorithm 1.
3) The atomic test vector t^a_f is assigned to f. A check is performed to prevent the assignments of duplicates.

[1] The method is applied for concurrent BIST but is basically also applicable to test compression-based designs.

4) Fault simulation techniques are used to find other faults detected by t^a_f. If such a fault is found, a reference to t^a_f is assigned to this fault to keep the additional memory requirements low.

As a result of this flow, the test set is decomposed into its atomic test vectors and each testable fault has one or more atomic test vectors assigned. Typically, there are many easy-to-detect faults which are detected by many (e.g. hundreds of) atomic test vectors. Preliminary studies have shown that the general effectiveness of the overall procedure is not diminished when the number of atomic test vectors assigned to one fault is bounded by a constant number to prevent memory overhead. Therefore, a maximum of 20 atomic test vectors is used in this work.

The advantage of using multiple detection is that the merging algorithm which will be described in the next section has more flexibility in merging the atomic test vectors and produce a more compact test set. Generally, it has been observed that the merging algorithm is more effective when the diversity of the atomic test vectors is high. Therefore, the TVD procedure is modified to obtain a more divers set of atomic test vectors.

A crucial step during TVD is the selection of the input to trace when more than one controlling value is assumed on the inputs (Function ChooseOneInputWithControllingValue() in Algorithm 1). If a static metric, e.g. based on testability measurements, is used, the algorithm will always follow the same path when it visits the same gate with more than one controlling value. This is disadvantageous for the diversity of the atomic test set. Therefore, a dynamic metric is used to select the input to trace.

A counter $g_c = 0$ is assigned to each gate g in the circuit and incremented each time when the gate is selected to be traced. When a decision which gate input to trace has to be made, the counters of each gate input which assumes a controlling value are evaluated and the input with the lowest counter value is selected. If two or more inputs have the same lowest counter value, the gate input is selected based on testability measurements. This procedure is similar to rotating backtrace proposed in [2].

This selection procedure ensures that the TVD procedure is not using always the same paths through the circuit and, consequently, produces atomic test vectors with a higher diversity.

III. Merge Algorithm

This section describes the general merge algorithm used in this work. It is assumed that the input test set T_f has already been decomposed into atomic test vectors which are assigned to their respective faults as described in the previous section.

It is further assumed that there is a maximum number of specified bits given by the user with knowledge about the applied test compression technique. A low number of specified bits does not guarantee itself that a test will not be dropped by the compression scheme. However, tests above a certain threshold are very likely to be dropped. More elaborated schemes for specific test compression systems can easily integrated or coupled with the proposed algorithm like regular ATPG tools as done in commercial tools.

The overall merge algorithm for the atomic test vectors is shown in Algorithm 2. The detectable fault set F as well as the set of atomic test vectors T^a are given as input. Additionally, a target number of maximum bits (threshold) is given. The flow is influenced by the ATPG-based classical dynamic compaction algorithm. First, an empty test t is created. Then, all faults are processed in a loop in a given order. For each fault, the set of assigned atomic test vectors is also processed in a given order. If one atomic test vector t^a_f of f is found to be compatible to t, the test is merged into t, f is removed from F and the next fault will be processed. If all faults have been processed, the new test is fault simulated and all additionally detected faults are removed from F. Then, t is added to the final test set T_p. This procedure stops when all faults are removed from f, i.e. all faults are detected by T_p.

Algorithm 2 Merge Algorithm

```
Input: Fault set $F = f_1, \ldots, f_n$, Set of atomic test vectors $T^a$, int maxBits
Output: Test set $T_p$
while $F$.empty() == false do
  Test $t$ = $X$;
  for all $f \in F$ do
    for all $t^a_f$ assigned to $f$ do
      if isCompatible($t^a_f$,$t$,maxBits) == true then
        merge($t$,$t^a_f$);
        $F$.remove(f);
        break;
      end if
    end for
  end for
  FaultSimulation($F$,$t$);
  $F$.removeDetectedFaults();
  $T_p$.add($t$);
end while
```

During the procedure, the check of compatibility is very important. Here, the maximum target number of bits *maxBits* is also considered. A test vector t and an atomic test vector t^a are *compatible* if

- they have no conflicting input values, i.e. 0 and 1 on the same input.
- the number of specified inputs of the union of t and t^a does not exceed *maxBits*.

The consideration of *maxBits* during the compatibility check ensures that the final test set T_p does not contain any test vector which exceeds the user-specified maximum number of specified bits.

The advantage of using multiple-detection instead of single-detection is that the merging algorithm is more flexible in combining atomic test vectors since it is able to choose from a larger pool of tests. However, since this is a greedy algorithm,[2] the ordering of the atomic test vectors and the faults, respectively, is very important for its effectiveness. Therefore, a detection-based ordering is proposed in the next subsection.

A. Detection-based Ordering

With each merge operation, the flexibility for future merge operations is further restricted. Performing the “wrong” merge

[2] Formal methods have also been studied for this application. However, the run times for larger circuits were too high for practical application.

TABLE I. PATTERNS OF EXAMPLE

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	det. faults
p_1	0	0	1	1	0	1	0	0	f_1, f_2, f_3
p_2	1	0	1	0	0	0	1	0	f_1, f_4, f_7
p_3	1	0	1	1	1	0	0	0	f_2, f_4, f_5
p_4	1	1	0	0	1	1	1	1	f_2, f_5, f_6

operations too early could be disadvantageous for the final pattern count since it may prevent other beneficous merge operations.

Two different orderings are used in Algorithm 2.

1) The order of the faults in F
2) The order of the atomic test vectors t^a_f assigned to f

The following aspects are considered for the fault ordering of F:

- Number of detections d – Faults with a smaller number of atomic test vectors should be processed first since the merging algorithm has a limited flexibility.
- Size of the shortest atomic test vector s – When the number of detections of two faults f_1, f_2 is equal, a different metric is used. The size of each assigned atomic test vector (number of specified bits) is analyzed and the fault with a larger shortest atomic test vector is processed first because it is expected that shorter atomic test vectors can be merged easier.

More formally, given two faults f_1, f_2 with number of detections d_{f1}, d_{f2} and size of the shortest atomic test vector s_{f1}, s_{f2}, the order $<$ (processed first) is defined as follows:

$$f_1 < f_2 = \begin{cases} 1 & \text{if}(d_{f1} < d_{f2}) \vee ((d_{f1} = d_{f2}) \wedge (s_{f1} > s_{f2})) \\ 0 & \text{else} \end{cases}$$

The second ordering is the ordering of the atomic test vectors assigned to each fault. Generally, it is better to try to merge atomic test vectors with less specified bits first since the probability is higher that the test is compatible to the currently constructed test and the merged test contains fewer specified bits. Given two atomic test vectors t^a_1 and t^a_2 as well as the number of specified bits b_{t1}, b_{t2} of each test, the ordering $<$ (processed first) is defined as follows:

$$t^a_1 < t^a_2 = \begin{cases} 1 & \text{if}(b_{t1} < b_{t2}) \\ 0 & \text{else} \end{cases}$$

The faults and atomic test vectors are ordered after TVD has been finished before the beginning of the merging process.

B. Example

This subsection presents an example for the application of the merging algorithm. A circuit C with eight inputs $i_1, \ldots, i_8$ and seven testable faults $f_1, \ldots, f_7$ is given. The patterns $p_1, \ldots, p_4$ generated to test these faults are presented in Table I. Each line also shows the faults detected by the respective pattern. Table II presents the atomic test vectors generated by the TVD technique. In the following, a care bit threshold of 5 is assumed for the partially-specified test set.

TABLE II. ATOMIC TEST VECTORS OF EXAMPLE

atv	det	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	extracted from
t_1^{a1}	f_1	0	0	1	X	X	X	X	X	p_1
t_1^{a2}	f_1	X	X	1	0	0	X	X	X	p_2
t_2^{a3}	f_2	X	0	1	1	X	X	X	X	p_3
t_2^{a4}	f_2	X	X	X	0	X	X	1	1	p_4
t_2^{a5}	f_2	X	X	1	1	X	X	0	0	p_1
t_3^{a6}	f_3	X	X	X	X	X	1	0	0	p_1
t_4^{a7}	f_4	1	0	1	X	X	X	X	X	p_2
t_4^{a8}	f_4	1	0	X	1	X	X	X	X	p_3
t_5^{a9}	f_5	X	X	0	X	X	1	X	X	p_4
t_5^{a10}	f_5	X	X	1	X	1	X	0	0	p_3
t_6^{a11}	f_6	1	1	X	X	1	1	X	X	p_4
t_7^{a12}	f_7	X	X	X	X	X	0	1	0	p_2

TABLE III. PARTIALLY-SPECIFIED PATTERNS USING DUPLICATION

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	det. faults
$p_{1.1}$	X	X	1	1	X	1	0	0	f_2, f_3
$p_{1.2}$	0	0	1	X	X	X	X	X	f_1
$p_{2.1}$	1	0	1	X	X	X	X	X	f_4
$p_{2.2}$	X	X	X	X	X	0	1	0	f_7
$p_{3.1}$	X	X	1	X	1	X	0	0	f_5
$p_{4.1}$	1	1	X	X	1	1	X	X	f_6

First, Table III show which patterns would be generated if the pattern structures are retained.

Here, p_1 is splitted into two patterns $p_{1.1}$ and $p_{1.2}$ detecting all faults in p_1. While splitting p_2, f_1 can be ignored since it is already detected by $p_{1.2}$. However, f_4 and f_7 cannot be detected together with the threshold of 5. Therefore, $p_{2.1}$ and $p_{2.2}$ are created. Patterns p_3 and p_4 do not have to be duplicated since from the faults detected by p_3 only f_5 have to covered. The same holds for p_4 and fault f_6. For both patterns, the atomic test vectors t_5^{a10} and t_6^{a11} can be used.

Now, the proposed procedure is presented. First, the following fault ordering is generated based on the number of detections:

$$f_7, f_6, f_3, f_4, f_1, f_5, f_2$$

The atomic test vector ordering presented in the Table II (top-down for each fault) is used as second criterion. The produced four patterns are given in Table IV.

The first pattern p_{M1} detects only f_7 since this fault cannot be detected together with any other atomic test vector using the given care bit threshold. Then, t_6^{a11} and t_5^{a9} are merged into p_{M2} detecting f_6 and f_5. For the third pattern p_{M3}, the atomic vectors t_3^{a6} and t_2^{a5} are merged. The remining faults f_4 and f_1 are then detected by p_{M4} which is generated by merging t_4^{a7} and t_1^{a2}. In this simple example, the resulting pattern set is as small as the initial pattern set but with no test exceeding the threshold of 5 care bits.

IV. EXPERIMENTAL RESULTS

The techniques proposed in this paper were implemented in C++ and integrated into an academic test framework running a SAT-based ATPG engine [25]. Experiments were conducted on an Intel Xeon E3-1240 (GNU/Linux, 64bit, 32GByte RAM, 3.4 GHz) in single-threaded mode.

Table V presents experimental results for the academic ISCAS'89 and ITC'99 benchmarks. Column *Init #Pat* presents the size of the fully-specified basis test set generated by the SAT-based ATPG procedure from [25]. Column *Target % spec.* gives the targeted (maximum) number of specified bits. Three different targets were used. The different thresholds are explained by the varying circuit characteristics. For some

TABLE IV. PARTIALLY-SPECIFIED PATTERNS USING MERGING

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	det. faults
p_{M1}	X	X	1	1	X	1	0	0	f_7
p_{M2}	X	X	X	X	X	0	1	0	f_6, f_5
p_{M3}	X	X	1	X	1	X	0	0	f_3, f_2
p_{M4}	1	1	X	X	1	1	X	X	f_4, f_1

circuits, it was not possible to guarantee a high fault coverage with 1 or 5% of maximum bits. The threshold was increased for these circuits. Three different configurations were used for which the resulting pattern count is given in the respective column:

- *Prev. #Pat* – Here, a pattern splitting (duplication) strategy similiar to previous work, e.g. used in [22], [24], was implemented, where the method focuses on the given pattern structures.
- *Single #Pat* – The proposed TVD method as well as the merging algorithm is used in this configuration. However, only one atomic test vector is extracted (single detection).
- *Mult. #Pat* – Here, the full methodology proposed in this paper is used. Atomic test vectors are extracted using TVD and multiple detection and the merging algorithm uses the proposed fault and test ordering.

The smallest pattern count entry for each circuit and each target threshold is marked bold. Comparing pattern splitting and the single detection approach, it can be seen that they produce mostly comparable results if a high threshold (50% X) is used. If the threshold is diminished, the proposed single detection approach has a clear advantage and produces less patterns. For the smallest threshold, the pattern count can even be reduced for some circuits to less than half of the pattern count.

The pattern count reduction can even be improved by using multiple detection. For all circuits and target thresholds, this configuration yields the best results. For b22 for instance, the pattern count is less than 40% of the pattern splitting test set size. The highest reduction compared to the single detection configuration is achieved for b17. Here, the pattern count can be reduced by additional 22%. It is also noticeable that this approach yields less patterns (with the same fault coverage) for half of the circuits when the threshold is set to 50%. These results clearly show the effectivness of the proposed methodology in generating a compact test set with partially-specified tests.

Next, Table VI presents the experimental results for four industrial circuits provided by NXP Semiconductors. The application of the proposed methodology is again applied in single and multiple detection mode for three different thresholds: 1%, 5% and 50% specified bits. However, the 1% care bit threshold was not feasible for two circuits due to the circuit characterics. Differently to the previous experiment, three different test sets were used for each circuit (given in the lines {*#T low, #T med, #T high*}). The test set generated by a commercial ATPG tool with highest compaction effort is marked by an '*'. The other two test sets were generated by SAT-based ATPG [25] using different compaction settings.

Column *Init #Pat* gives the pattern count of the fully-specified initial test set. Columns entitled *t TVD* presents the run time spent for the TVD process in CPU seconds,[3] while column *t merge* gives the run time spent for the merging process. The resulting test set size is shown in columns named *#Pat*.

Generally, it can be seen that the test set size grows (as expected) when reducing the specified bits threshold. However, this is acceptable since it enables the use of more effective test compression techniques. Comparing the run times, the approach using multiple detection needs more run time for TVD as well as for the merging algorithm than the approach using single detection. The reason for this is that more data has to be processed during atomic test vector extraction as well as during the merging procedure. However, the multiple detection technique also yields improved results.

Again, if a high threshold is used, the compaction results between single and multiple detection are mostly comparable with a slight advantage for multiple detection. However, for a small number of specified bits, i.e. 5% and 1%, the multiple detection approach is far more effective and yields the best results for the targeted circuits. Here, the good results can be obtained from test sets generated by a commercial ATPG tool as well as by SAT-based ATPG.

A further interesting observation can be made from the comparison of the different test sets for one circuit. In most cases, the initial size of the fully-specified test set directly influences the final test set size of the partially-specified test set. A large fully-specified test set leads also to a larger partially-specified test set, while a small basis test set is typically transformed into a smaller partially-specified test set. This shows that the proposed methodology benefits from the compaction effort of the inital test set generation.

In summary, the experimental results show that the proposed techniques are more effective in producing a small test set with a target threshold of specified bits than previous pattern duplication techniques. This holds particularly if a small number of care bits is required which is important for designs using modern test compression techniques. It is also shown that the proposed techniques benefits from regular test compaction techniques.

Future work is the coupling with different test compression techniques to reduce the test pattern loss due to the incompatibility with the applied test compression scheme and the application in the field of low power testing.

V. CONCLUSIONS

Current designs use test compression techniques to reduce the amount of test data in the post-production test. This puts constraints on the test generation process. A large number of unspecified bits is necessary that effective test compression techniques can be applied. This paper proposes new decomposing and merging techniques to transform a fully-specified test set into a partially-specified test set in which each test vector is guaranteed to have a lower number of care bits than a specified threshold. The experimental results show that these technique can be effectively applied to academic benchmarks and industrial circuits. Additionally, it is shown that smaller fully-specified test sets lead to smaller partially-specified test sets.

[3] Please note that the main share of the run time is spent by an academic fault simulator not trimmed for efficiency.

TABLE V. EXPERIMENTAL RESULTS – ACADEMIC BENCHMARKS

	Init	Target	Prev.	Single	Mult.	Target	Prev.	Single	Mult.	Target	Prev.	Single	Mult.
circ	#Pat	% spec.	#Pat	#Pat	#Pat	% spec.	#Pat	#Pat	#Pat	% spec.	#Pat	#Pat	#Pat
s35932	47	50	53	24	**20**	5	220	181	**166**	1	1134	1042	**922**
s38417	99	50	106	95	**85**	15	280	231	**213**	5	823	728	**668**
s38584	107	50	120	**93**	**93**	5	827	697	**669**	1	1883	828	**808**
b15	661	50	686	686	**660**	30	813	734	**676**	15	1642	741	**683**
b17	745	50	778	810	**729**	15	1701	1608	**1273**	5	4446	2238	**1760**
b20	335	50	479	484	**438**	30	827	741	**632**	1	–	–	–
b21	323	50	464	455	**417**	30	779	620	**596**	1	–	–	–
b22	327	50	471	507	**449**	30	829	810	**701**	15	1935	844	**736**

TABLE VI. EXPERIMENTAL RESULTS – INDUSTRIAL CIRCUITS

		Single Detection							Multiple Detection						
			50% spec.		5% spec.		1% spec.			50% spec.		5% spec.		1% spec.	
test set	Init #Pat	t TVD	t merge	#Pat	t merge	#Pat	t merge	#Pat	t TVD	t merge	#Pat	t merge	#Pat	t merge	#Pat
p35k: 2912 flipflops, 122500 faults															
#T low*	1571	52.3	8.8	1451	11.1	1610	–	–	1484.2	13.2	1415	28.1	1523	–	–
#T med	981	62.1	7.0	981	10.81	1342	–	–	1536.0	10.3	980	25.6	1322	–	–
#T high	492	53.5	3.9	**488**	9.2	1012	–	–	1080.7	6.0	491	24.9	**991**	–	–
p81k: 4029 flipflops, 363748 faults															
#T low	1068	39.0	43.2	1480	215.7	3842	–	–	722.5	79.0	1118	1387.2	3369	–	–
#T med*	386	27.7	33.0	1025	233.6	4237	–	–	788.0	69.3	**847**	1303.9	3489	–	–
#T high	383	25.3	37.9	1296	206.5	3823	–	–	605.9	63.2	1032	1063.5	**3289**	–	–
p100k: 5915 flipflops, 349526 faults															
#T low	2084	63.9	17.1	2068	55.1	2083	166.2	3907	742.9	22.3	2054	258.8	2056	1056.0	3536
#T med*	2054	61.4	17.4	2049	55.5	2135	162.3	3715	794.6	22.8	**2048**	260.6	2073	1125.2	3552
#T high	2048	61.4	16.9	2049	57.5	2059	175.7	3883	729.6	22.4	**2048**	247.1	**2052**	1043.2	**3450**
p330k: 18016 flipflops, 1226414 faults															
#T low	4328	422.8	159.4	4020	348.4	4027	1309.4	5470	3689.6	250.0	3688	1461.8	3696	9153.4	4807
#T med*	2963	296.1	118.9	2903	314.5	2946	1348.0	5332	2991.4	173.4	2794	1525.7	2802	9982.4	4707
#T high	1730	216.0	88.9	1852	270.2	1892	1060.3	4426	2729.5	137.4	**1822**	1214.9	**1817**	6941.5	**3876**

VI. ACKNOWLEDGMENT

This work has been supported by the Institutional Strategy of the University of Bremen, funded by the German Excellence Initiative.

REFERENCES

[1] P. Goel and B. C. Rosales, “Test generation and dynamic compaction of tests,” in *International Test Conference*, 1979, pp. 189–192.

[2] I. Pomeranz, L. N. Reddy, and S. M. Reddy, “COMPACTEST: A method to generate compact test sets for combinational circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 7, pp. 1040–1049, 1993.

[3] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, “Cost-effective generation of minimal test sets for stuck-at faults in combinational logic circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 12, pp. 1496–1504, 1995.

[4] I. Hamzaoglu and J. H. Patel, “Test set compaction algorithms for combinational circuits,” in *International Conference on Computer-Aided Design*, 1998, pp. 283–289.

[5] X. Lin, J. Rajski, I. Pomeranz, and S. M. Reddy, “On static test compaction and test pattern ordering for scan designs,” in *International Test Conference*, 2001, pp. 1088–1097.

[6] B. Koenemann, “LFSR-coded test patterns for scan designs,” in *European Test Conference*, 1991, pp. 237–242.

[7] A. Chandra and K. Chakrabarty, “System-on-a-chip test data compression and decompression architectures based on golomb codes,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 355–368, 2001.

[8] I. Bayraktaroglu and A. Orailoglu, “Concurrent application of compaction and compression for test time and data volume reduction in scan designs,” *IEEE Transactions on Computers*, vol. 52, no. 11, pp. 1480–1489, 2003.

[9] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, “Embedded deterministic test,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 776–792, 2004.

[10] S. Mitra and K. S. Kim, “XPAND: An efficient test stimulus compression technique,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 163–173, 2006.

[11] N. A. Touba, “Survey of test vector compression techniques,” *IEEE Design & Test of Computers*, vol. 23, no. 4, pp. 294–303, 2006.

[12] D. Czysz, G. Mrugalski, N. Mukherjee, J. Rajski, P. Szczerbicki, and J. Tyszer, “Deterministic clustering of incompatible test cubes for higher power-aware edt compression,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 8, pp. 1225–1238, 2011.

[13] S. Hellebrand, B. Reeb, S. Tarnick, and H.-J. Wunderlich, “Pattern generation for a deterministic BIST scheme,” in *International Conference on Computer-Aided Design*, 1995, pp. 88–94.

[14] R. Dorsch and H.-J. Wunderlich, “Tailoring ATPG for embedded testing,” in *International Test Conference*, 2001, pp. 530–537.

[15] A. Kumar, M. Kassab, E. Moghaddam, N. Mukherjee, J. Rajski, S. M. Reddy, J. Tyszer, and C. Wang, “Isometric test compression with low toggling activity,” in *International Test Conference*, 2014, pp. 1–7.

[16] M. Sauer, S. Reimer, I. Polian, T. Schubert, and B. Becker, “Provably optimal test cube generation using quantified Boolean formula solving,” in *ASP Design Automation Conference*, 2013, pp. 533–539.

[17] M. Sauer, S. Reimer, T. Schubert, I. Polian, and B. Becker, “Efficient SAT-based dynamic compaction and relaxation for longest sensitizable paths,” in *Design, Automation and Test in Europe*, 2013, pp. 448–453.

[18] A. El-Maleh and A. Al-Suwaiyan, “An efficient test relaxation technique for combinational circuits,” in *VLSI Test Symposium*, 2002, pp. 53–59.

[19] K. Miyase and S. Kajihara, “XID: Don’t care identification of test patterns for combinational circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 2, pp. 321–326, 2004.

[20] M. A. Kochte, C. G. Zoellin, M. E. Imhof, and H.-J. Wunderlich, “Test set stripping limiting the maximum number of specified bits,” in *IEEE International Symposium on Electronic Design, Test and Applications*, 2008, pp. 581–586.

[21] S. N. Neophytou and M. K. Michael, “Test set generation with a large number of unspecified bits using static and dynamic techniques,” *IEEE Transactions on Computers*, vol. 59, no. 3, pp. 301–316, 2010.

[22] I. Pomeranz and S. M. Reddy, “Reducing the number of specified values per test vector by increasing the test set size,” *IEE Computers and Digital Techniques*, vol. 153, no. 1, pp. 39–46, 2006.

[23] A. H. El-Maleh, M. I. Ali, and A. A. Al-Yamani, “Reconfigurable broadcast scan compression using relaxation-based test vector decomposition,” *IET Computes and Digital Techniques*, vol. 3, no. 2, pp. 143–161, 2009.

[24] M. A. Kochte, C. G. Zoellin, and H.-J. Wunderlich, “Concurrent self-test with partially specified patterns for low test latency and overhead,” in *IEEE European Test Symposium*, 2009, pp. 53–58.

[25] S. Eggersglüß, K. Schmitz, R. Krenz-Bååth, and R. Drechsler, “Optimization-based multiple target test generation for highly compacted test sets,” in *IEEE European Test Symposium*, 2014, pp. 1–6.