Reusing Scan Chains for Test Pattern Decompression

Rainer Dorsch Hans-Joachim Wunderlich

University of Stuttgart
Breitwiesenstr. 20-22, D-70567 Stuttgart, Germany
phone +49-711-7816-215, fax +49-711-7816-288
{rainer.dorsch,wu}@informatik.uni-stuttgart.de

Abstract

The paper presents a method for testing a system-on-a-chip by using a compressed representation of the patterns on an external tester. The patterns for a certain core under test are decompressed by reusing scan chains of cores idle during that time. The method only requires a few additional gates in the wrapper, while the mission logic is untouched. Storage and bandwidth requirements for the ATE are reduced significantly.

Keywords: System-on-a-Chip, Embedded Test, BIST

1 Introduction

The ITRS roadmap [23] predicts a significant cost increase of automatic test equipment (ATE) to test the next generation of system chips. One reason is the high test data volume, which has to be stored in the expensive fast memory, other reasons are bandwidth and speed requirements. The test application time per chip increases, too, due to the limited bandwidth and the high test data volume.

These problems may be addressed by embedded test circuitry [26], which moves parts of the test functionality of the ATE onto the chip under test. These embedded testers are not used after production test and their hardware overhead should be minimized.

The test data stored on the external ATE are typically statically and dynamically compacted test vectors [12, 14]. Examining these test sets shows a considerable amount of the bits in the compacted test sets are originally unspecified and randomly filled before they are stored on the external ATE. This implies that often a large part of the ATE memory contains random data, and a large part of the ATE-chip bandwidth is used to transport random data.

BIST test pattern generators (TPG) using a "store & generate" approach, like reseeding of LFSRs [17] or cellular automata [1] make use of unspecified bits in test sets. Advanced decompression schemes emulate a multi-polynomial, multi-seed TPG on a CPU [15] or add sophisticated dedicated hardware to use variable length seeds [22]. These generators are relatively small and reduce the size of the test set, approximately to the number of specified bits of the uncompressed vectors.

A hybrid external/logic-BIST approach [9] generates the unspecified bits using a pseudo-random pattern generator (PRPG) on-chip and applies the specified bits directly from the ATE.

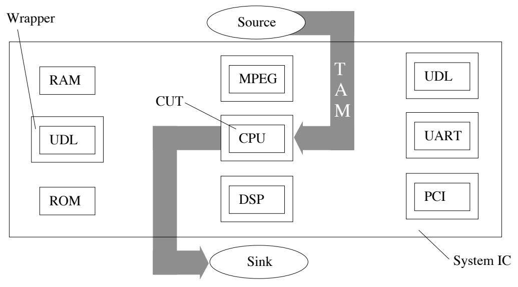
In this paper an architecture is presented, which decompresses the test patterns of a Core A, using the scan chains of an idle Core B. The architecture is called REusing Scan chains for test Pattern decompressIoN (RESPIN) and reduces the required memory and bandwidth from ATE to the SOC under test significantly and generates on-chip precomputed test patterns, while it may reorder the patterns and may embed additional link patterns.

Test methods for SoCs are usually based on test access mechanisms (TAM) and wrappers which isolate the cores from their environment and allow testing them separately [11, 16, 19, 20, 25, 27]. In the presented method, only a multiplexer and some additional wiring within the test wrapper of a core are needed to decompress test patterns.

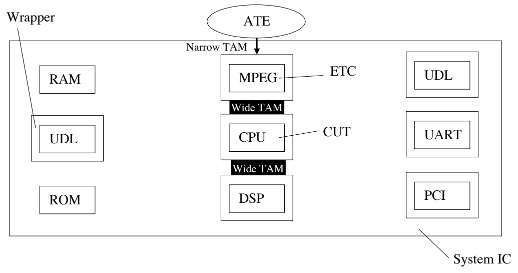
Fig. 1 illustrates the test architecture of RESPIN. Instead of using the grey medium sized test access mechanism (TAM) (Fig. 1a) from the ATE directly to the core under test (CPU in this example), a *narrow TAM* (narrow is here used for low data width and low frequency) is used to transport *compressed test data* from the source to an *embedded tester core* (MPEG core in the example) (Fig. 1b). The scan chains of the embedded tester core (ETC) are used for the decompression of the patterns for the core under test (CUT). The *decompressed test data* are propagated on a *wide TAM* from the ETCs to the CUT. The wide TAM also propagates test responses to a compactor, which serves as pattern sink. This may be a small LFSR or another core implementing system functionality (DSP in the example) [4, 21, 24]. In practice,

This work was supported by the DFG under grant Wu 245/1-4.

the CUT may be stimulated by one or several ETCs.



(a) Standard test architecture.



(b) Test architecture of RESPIN

Figure 1. Test architectures.

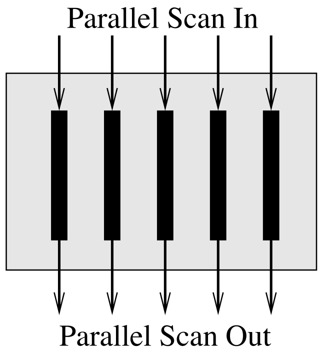
In the next section, the test architecture is introduced in detail, in Section 3, the test flow is presented. Section 4 contains the algorithm computing the data supplied by the ATE on the narrow TAM in order to generate the precomputed test patterns on the wide TAM. Section 5 presents experimental results for the ISCAS89 [2] benchmarks.

2 Test Architecture

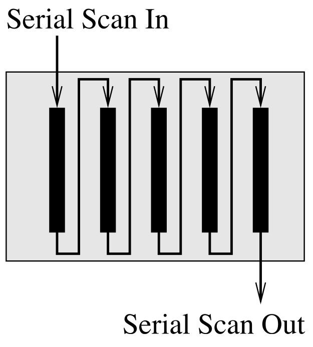
The RESPIN architecture assumes that both the ETC and the CUT employ full scan design. Extensions to more complex architectures are possible.

2.1 CUT and ETC

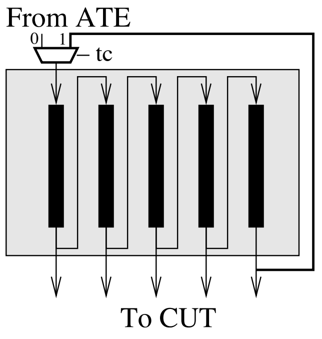
Embedded cores employing a full scan design are usually equipped with parallel test access to the scan chains as shown in Fig. 2(a) in order to reduce the time required to scan in a test pattern. Additionally, an embedded core is often equipped with serial test access as shown in Fig. 2(b) to ease debugging on board level. Serial test access is discussed as a mandatory element of the proposed P1500 standard for embedded core test [18], parallel test access may be an optional element of P1500.



(a) Parallel Test Access



(b) Serial Test Access



(c) ETC Architecture

Figure 2. RESPIN test architecture.

We assume that the CUT employs full scan design and has at least parallel test access. The test of CUT consists of scanning in and out test data through the parallel test access as shown in Fig. 2(a) and capturing the system response on the scanned in test patterns.

For the ETC we assume, that it has parallel and serial test access to its scan cells. To the wrapper of the ETC, a single multiplexer (controlled by the tc signal) and a feedback wire has to be added (Fig. 2(c)). Depending on the input of the additional multiplexer, the ETC operates either in serial test access mode ($tc = 0$) or in a mode shifting the scan chain content circularly ($tc = 1$). These modes are called *serial mode* and *circular mode*, respectively.

The parallel test access outputs of the ETC are connected to the parallel test access inputs of the CUT. The control signal tc of the ETC is connected to the scan enable signal of the CUT. Table 1 shows whenever the CUT captures the system response, the ETC is in serial mode and loads a bit from the encoded test data. Whenever the CUT scans-in test data, the ETC is in circular mode. For using cores as ETC both their shift and capture clocks must be compatible with the CUT clocks.

t_c =scan enable	0	1
CUT	capture mode	scan mode
ETC	serial mode	circular mode

Table 1. Operation modes of ETC and CUT

2.2 Example

As an example assume a CUT containing 3 scan chains with 3 scan cells each. The embedded tester core has 5 scan chains with 5 scan cells each (Fig. 3). The number $\widehat{l_C}$ of scan cells in the longest scan chain of the CUT is in the example $\widehat{l_C} = 3$. The ETC has $l_E = 25$ scan cells. In the general case, the ETC may have scan chains of different lengths. In this example a *serial mode cycle* is always followed by $\widehat{l_C} = 3$ *circular mode cycle*s.

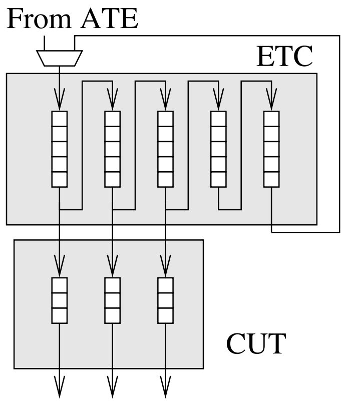


Figure 3. Example for a test architecture with a simple ETC and CUT.

2.3 Dependency Analysis

For many test methods, dependencies in the pattern generator, e.g. linear dependencies in LFSR based pattern generators [6], may prohibit the generation of certain test patterns for the CUT. This subsection analyses the dependencies in the RESPIN architecture. It will be shown that an arbitrary pattern can be scanned into the CUT, if some conditions are satisfied. Moreover, if these conditions do not hold, minimal architectural changes allow the generation of any pattern in the CUT.

Assume that the scan cells of the ETC are numbered ascending in serial scan direction, starting from 0. Assume further, a bit is scanned into the ETC from the ATE. p serial mode clock cycle later it is in scan cell

$$i_p = p \cdot (\widehat{l_C} + 1) \bmod l_E \qquad (1)$$

of the ETC. The bit will be overwritten by another bit from the ATE $p_e > 0$ serial mode cycles after it was scanned-in with $i_{p_e} = 0$ and $\nexists p.(0 < p < p_e \wedge i_p = 0)$. p_e depends on the greatest common divisor $g.c.d.(\widehat{l_C} + 1, l_E)$

$$p_e = \frac{l_E}{g.c.d.(\widehat{l_C} + 1, l_E)}. \qquad (2)$$

Theorem 1 *An arbitrary pattern can be realized in the CUT in at most $l_E \cdot (\widehat{l_C} + 1)$ cycles if these four conditions*

1. *The greatest common divisor satisfies the condition*

$$g.c.d.(\widehat{l_C} + 1, l_E) = 1. \qquad (3)$$

2. *The ETC has at least as many scan chains as the CUT.*
3. *No scan chain in the CUT is longer than the scan chain of the ETC it is connected to.*
4. *No scan chain of the ETC is connected to more than one scan chain of the CUT.*

are satisfied. □

PROOF The theorem is shown in two steps:

1. It is shown, that the content a scan cell in the ETC is mapped to at most one scan cell of the CUT.
2. It is shown, that any pattern may be realized in the ETC after a finite number of serial mode cycles.

Step one follows from conditions two to four, since the content of the scan chain of the ETC is copied into the scan chain of the CUT it is connected to.

To show step two, the first condition of the theorem is applied to Eqn. (2). This yields $p_e = l_E$, i.e. a bit stays in the ETC for l_E serial mode cycles. This implies that always a permutation of the l_E bits previously scanned in is contained in the ETC. Hence, within l_E serial mode cycles, equivalent to $l_E \cdot (\widehat{l_C} + 1)$ clock cycles, any pattern may be realized in the ETC. ■

For each bit i in the ETC, Eqn. (1) defines implicitly how many serial mode cycles $p(i) < l_E$ before the bit i has been scanned into the ETC.

An immediate consequence of Theorem 1 is that each bit in a scan cell of the CUT was scanned into the ETC from the ATE $q(i)$ clock cycles before. For $q(i)$ holds $q(i) = q(j) \rightarrow i = j$.

If two cores do not satisfy the conditions of Theorem 1, a few minor changes in the test configuration will help (Fig. 4).

Condition 1: Either l_e or $\widehat{l_C}$ have to be increased until Eqn. (3) holds. Inserting some additional flip flops into the feedback wire of the wrapper increases l_e and is paid by some additional hardware. Applying a few more circular clock cycles increases $\widehat{l_C}$ at the cost of longer test application time.

Condition 2: Several cores may be combined to a single ETC with a sufficient large number of scan chains.

Condition 3: Obviously, the longer scan chains of the ETC must be used. If sufficiently long scan chains are not available, Condition 3 may be satisfied by not connecting the short scan chains of the ETC. If a scan chain of the ETC is not connected to the CUT, the length of the next scan chain is increased by the number of scan cells of the unconnected scan chain. Fixing this problem may also involve to use more than one ETC.

Condition 4: This is a design restriction which can always be followed.

3 Test Flow

The test flow shown in Fig. 5 is similar to the standard test flow for scan design. For each core under test, one or several cores serve as ETCs.

To increase the compression efficiency of the encoding algorithm, in a first iteration only random pattern resistant faults are considered. Random pattern resistant faults are determined by fault simulation with random patterns. The number of the patterns is a parameter specified by the user. In a second iteration, deterministic patterns for the faults not detected in the first iteration are encoded.

An ATPG tool is used to generate deterministic test patterns for the random pattern resistant faults without filling the "don't care" bits in the patterns. Next, the encoding algorithm computes the compressed test information using the ETC scan chain configuration and the deterministic test patterns. When the compressed test information is decoded by the ETC, the deterministic patterns and link patterns are generated. Both are fault simulated against all faults.

The fault efficiency reached after the second iteration is guaranteed to be not lower than the fault efficiency reached by the ATPG tool generating the deterministic patterns.

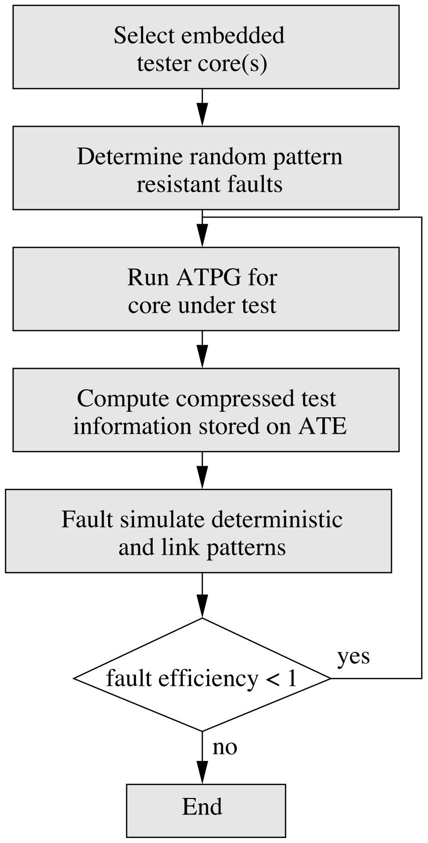

Select embedded tester core(s)
Determine random pattern resistant faults
Run ATPG for core under test
Compute compressed test information stored on ATE
Fault simulate deterministic and link patterns
fault efficiency < 1
yes
no
End

Figure 5. RESPIN test flow.

To continue the example of the previous section, assume now that the ATPG tool found three deterministic test patterns A, B, and C shown in Fig. 6 for the CUT of Fig. 3. The vectors contain the bit values 1, 0, and - ("don't care") and are shown in the scan chains of the CUT. Now we have to find the compressed test information which is needed to stimulate the ETC in such a way that it generates the three deterministic patterns. The algorithm to compute the compressed test information is given in the next section.

4 Computing Compressed Test Information

The unencoded test data for the CUT is a test set T of full-scan patterns $t \in T$ which may contain "don't care" bits and may be applied in an arbitrary order to the CUT. The

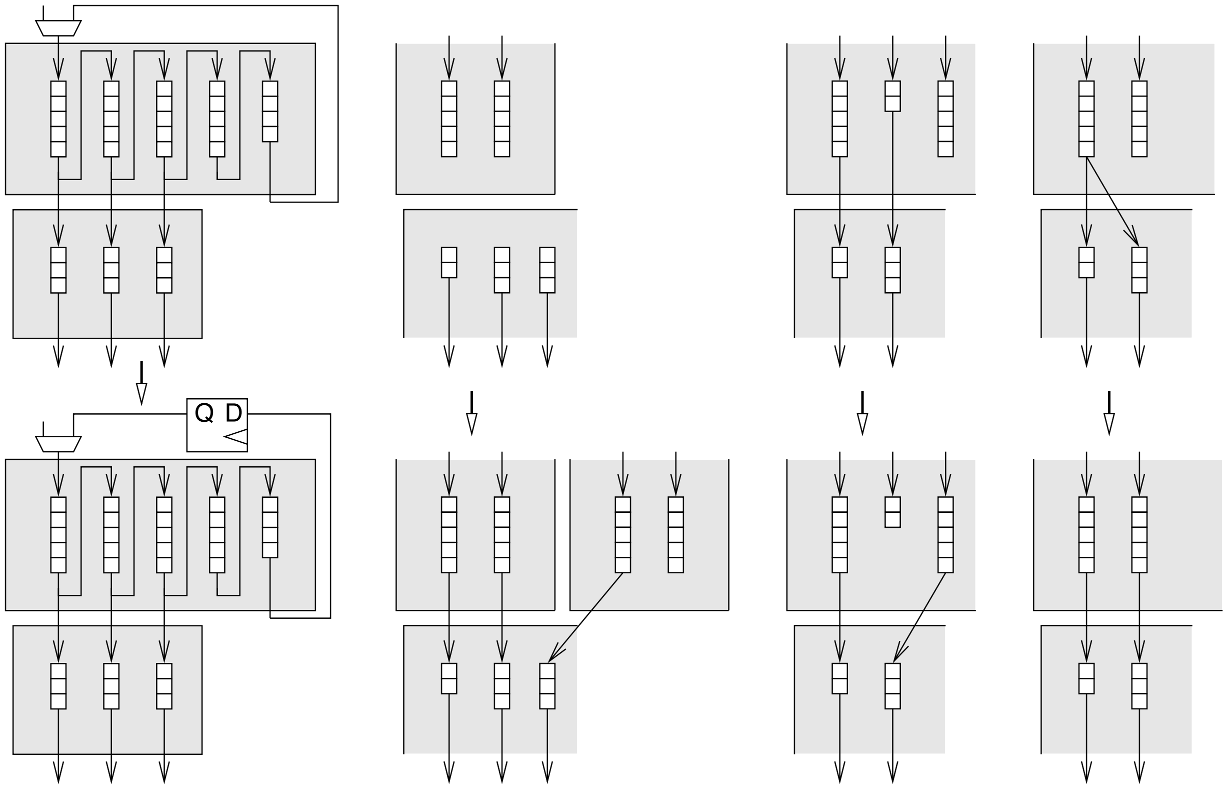


Figure 4. Fixing Conditions 1 to 4.

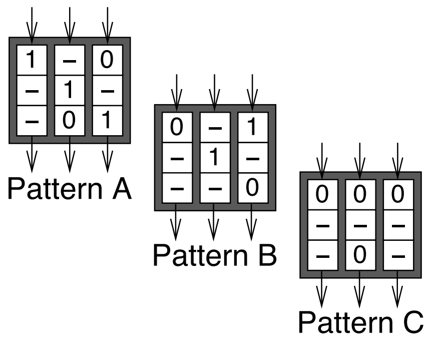


Figure 6. Three deterministic test patterns A, B, and C for the CUT

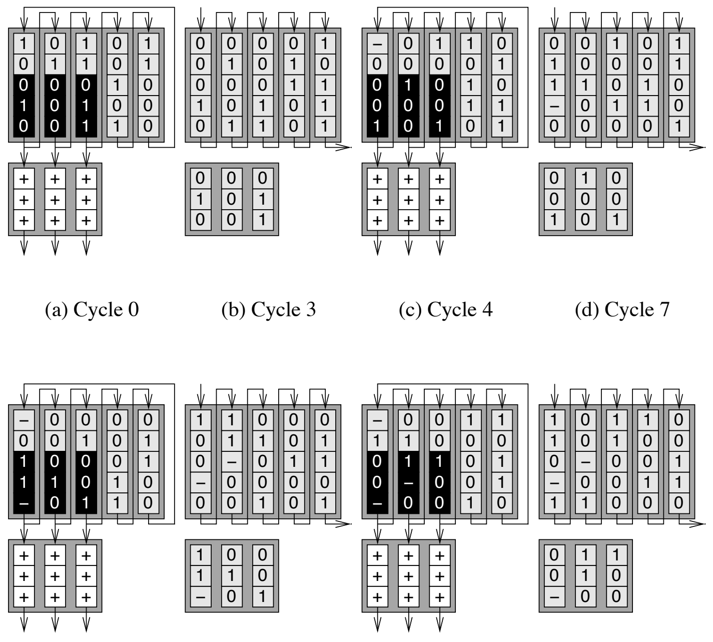
compressed test data is an input bit stream $a_0, a_1, \ldots, a_{i_e}$. If the input sequence is applied to the ETC, controlled as described in section 2, it generates an output sequence which contains the test set T and intermediate link patterns.

4.1 Example

This subsection demonstrates the compression algorithm using the example introduced in Sections 2 and 3, and the next subsection introduces the formal algorithm. The initial state (i.e. reset state or scanned in) of the ETC is shown in Fig. 7a.

During $\widehat{l_C} = 3$ cycles in circular mode, the bits in scan chains of the ETC are scanned circularly as indicated by the arrows. The dark bits in the ETC are copied into the CUT using the wide TAM. By comparing with Fig. 6 it is found, that pattern C is encoded already by the initial state. The resulting states of the ETC and CUT after the circular cycles are shown in Fig. 7b.

Next, the ETC runs in the serial mode for one cycle, scanning in a bit which is not yet known and thus represented as unspecified. Its value will be determined as soon as it is required to generate an unencoded pattern out of T. The CUT performs a capture operation and the system response (indicated by -) on the test pattern is stored in the scan chains of the CUT. Fig. 7c shows the resulting states



(a) Cycle 0 (b) Cycle 3 (c) Cycle 4 (d) Cycle 7

(e) Cycle 8 (f) Cycle 11 (g) Cycle 12 (h) Cycle 15

Figure 7. Encoding the example test set.

of ETC and CUT after the serial mode cycle.

None of the two unencoded patterns A and B is compatible with the pattern copied into the CUT (dark in Fig. 7c) during the next cycles in circular modes. Fig. 7d shows the test pattern copied into the CUT after the three cycles in circular mode. This pattern does not match any of the deterministic patterns, thus it is a link pattern.

Fig. 7e shows the system one cycle in serial mode later. The dark bits in the ETC encode pattern A and are copied into the CUT during the next three cycles in circular mode. The result is shown in Fig. 7f.

During the next cycle in serial mode a third unknown bit is scanned into the ETC. The CUT captures the system responses as shown in Fig. 7g. Pattern B may no be merged with the dark bits in the ETC scanned into the CUT next. To merge the pattern and the ETC content bit 4 of the second scan chain has to be fixed to 1, since it is specified in pattern B and unspecified in the ETC. Here it is tracked back easily that this was the bit a_0 scanned into the scan chains of the ETC from the ATE in cycle 4. Fig. 7h shows the system three cycles in circular mode later, the pattern is in the CUT and the relevant bit is now specified both in the ETC and CUT.

Since only a_0 was fixed during the encoding process, $a_0 = 1, a_1 = a_2 = a_3 = -$ are the data stored as compressed test data on the ATE. If $a_0 \dots a_3$ are transported on the narrow TAM from the ATE to the ETC, the three deterministic test patterns and a link pattern are copied on the wide TAM for the ETC into the CUT. For larger benchmarks, the fraction of unspecified bits in the compressed test set is significantly lower. The next subsections give a formal description of the algorithm.

4.2 Definitions

In this subsection, fundamental definitions simplifying the notation of the encoding algorithm are introduced.

Table 2 contains the truth table of the function $MERGE : \{0, 1, -\}^2 \rightarrow B$ which is true whenever two bits may be merged, i.e. do not contradict each other (an unspecified bit (-) may be merged with both 0 and 1). If the $MERGE()$ function is applied to a vector of bits, it is evaluated bitwise and returns a boolean value. The two vectors may be *merged*, if all bits may be merged.

	0	1	-
0	1	0	1
1	0	1	1
-	1	1	1

Table 2. Table for the operation $\mathrm{MERGE}(a, b)$**.**

Table 3 contains the truth table of the intersection function $\cap : \{0, 1, -\}^2 \rightarrow B$, which intersects two bits. The operation is only defined for bits which may be merged. If the operation is applied to a vector of bits, the operation is performed bitwise and the result is a vector of the same dimension.

	0	1	-
0	0	*	0
1	*	1	1
-	0	1	-

Table 3. Table for the intersection operation $\cap(a, b)$. * means undefined.

Algorithm 1 RESPIN compression algorithm

```
Initialize compressed test information;
$T = T_0$ (test set);
$k = 0$;
while $T \neq \emptyset$ do
  $\mathbf{c} = a_{k-q(0)}a_{k-q(1)}a_{k-q(2)} \cdots a_{k-q(3)}$.
  for all $\mathbf{t} \in T$ do
    if MERGE($\mathbf{c}, \mathbf{t}$) then
      Restrict $\mathbf{c}$ by $\mathbf{c} = \cap(\mathbf{c}, \mathbf{t})$;
      $T = T \setminus \{\mathbf{t}\}$;
    end if
  end for
  $k = k + 1$;
end while
```

4.3 Algorithm

Algorithm 1 computes the compressed test information out of a set of deterministic patterns and the scan chain configuration of the ETC and the CUT. First the compressed test data $a_0a_1 \ldots$ are initialized with $-$. A counter variable k, which numbers the cycles in serial mode, is initialized with 0. All deterministic patterns are encoded iteratively in the while loop. First, the pattern $\mathbf{c}$, which is copied into the CUT in serial mode cycle i, is extracted from the compressed ATE data using the mapping function $q(i)$, where i is the bit position of the scan element in the CUT

$$\mathbf{c} = a_{k-q(0)}a_{k-q(1)}a_{k-q(2)} \cdots a_{k-q(l_C)}.$$

The extraction is done by reference, i.e. if $\mathbf{c}$ is modified, also the corresponding bits $a_{k-q(i)}$ in the compressed test information are modified. If a pattern $\mathbf{t}$ mergable with $\mathbf{c}$ is found, $\mathbf{c}$ is restricted to that pattern by intersecting $\mathbf{c}$ and the pattern $\mathbf{t}$. This fixes the bits in $\mathbf{c}$, which are specified in $\mathbf{t}$, to the value they have in $\mathbf{t}$. The encoded pattern $\mathbf{t}$ is removed from the set of unencoded patterns T.

If the number of scan cells in the CUT and the ETC are equal, the algorithm to compute the compressed test information stored by the ATE is identical with the algorithm introduced in the context of non-linear feedback shift registers used for deterministic test pattern generation [8].

4.4 Test Application Time

Due to the insertion of link patterns into the test sequence the test application time of RESPIN may be higher than the test application time for a compacted test set in an architecture as shown in Fig. 2(a).

The test application time of RESPIN may be reduced by increasing the bandwidth of the internal wide TAM or external narrow TAM. Implementing more but shorter scan chains requires an internal TAM with a higher bandwidth and reduces the test application time. Increasing the bandwidth of the external narrow TAM reduces the test application time by reducing the number of link patterns embedded into the deterministic test set. [13]

5 Experimental Results

In this section, experimental results for a full-scan version of the ISCAS89 [2] circuits containing multiple scan paths of length 100 bit each are presented. We only report results for benchmarks which are not testable by random patterns plus a few deterministic patterns (like s35932), which could be applied without encoding. The RESPIN algorithm was implemented in C++ for encoding test patterns. The random pattern resistant faults were determined by applying 10,000 random patterns. The deterministic test patterns were generated by a commercial ATPG tool. The ATPG patterns contained the bit values 0, 1, and “don’t care”. The ATPG patterns were encoded by RESPIN. The current implementation of RESPIN does not yet use the embedded link patterns for fault dropping. 100% fault efficiency was reached in all cases.

5.1 Test Data Volume Reduction

This subsection compares the test data volume of the compressed RESPIN test data stored on ATE with a statically and dynamically compacted test set, which would be stored on ATE in the standard test architecture shown in Fig. 2a. The length of the scan chains of the embedded tester core was selected as 100. Since $\widehat{l_C} + 1 = 101$ is a prime number, Eqn. (3) is satisfied. Column 1 of Table 4 contains the name of the benchmark circuit, Column 2 shows the number of pseudo-primary inputs of the CUT, Column 3 shows the number of ATPG patterns used as input of RESPIN, Column 4 shows the CPU time on a SUN

Bench- mark	PPIs	# ATPG patterns	CPU in min	compact TS in bits	RESPIN in bits	reduct.	Add-on in bits	reduct.
s420.1	34	30	0.05	2,584	362	86%	254	76%
s838.1	66	143	0.5	10,494	1,925	82%	190	80%
s5378	214	36	0.3	25,038	618	98%	1,478	92%
s9234.1	247	356	5	36,556	8,006	78%	646	76%
s13207.1	700	377	3	168,000	1,963	99%	2,331	97%
s15850.1	611	257	5	75,768	5,139	93%	628	92%
s38417	1664	1532	204	154,752	31,020	80%	23	80%
s38584.1	1464	268	8	194,712	3,627	98%	2,862	97%

Table 4. Compressed test data volume in bits compared to compacted test sets.

Ultra10/450 MHz for the pattern encoding in minutes, Column 5 shows the test data volume of a statically and dynamically compacted test set in bits generated by a commercial ATPG tool, and Column 6 shows the compressed test data volume in bits, which needs to be stored on the ATE using the RESPIN architecture. Column 7 shows fraction the test data volume of RESPIN compared to the compact test set test data volume. Column 8 shows the number of additional bits which are computed in the second iteration to guarantee complete fault coverage. Column 9 shows the reduction after the second iteration.

The ATPG patterns used as input for RESPIN are neither statically nor dynamically compacted. It was experimentally observed, that in few cases, static and dynamic compaction before the application of RESPIN reduces the compressed test data volume slightly, but in more cases the compressed test data volume increased significantly, when precompacted patterns were encoded.

5.2 Comparison with Other Methodologies

RESPIN requires only minor changes of the SOC hardware and scan path architecture. The ETC acts as a simple filter, and dedicated hardware pattern generators are not needed. Despite its very simple structure, RESPIN outperforms the hardware schemes proposed so far for embedded testing, or provides comparable results at significantly less hardware costs.

Table 5 compares the RESPIN with other known structures, which may be used for embedded testing. A multi-polynomial, multi-seed random BIST scheme [15] applies 10,000 random patterns and encodes patterns for the remaining faults in seeds for LFSRs. The scheme is emulated on a microprocessor, the resulting test data volume in bits is shown in column JETTA98. Methods which reuse existing accumulator structures to generate test patterns are shown in column DATE00 [3] and ITC98 [10]. DATE00 implements a genetic algorithm, which computes the seeds for an accumulator. When the accumulator is seeded and running, it generates patterns which achieve full coverage for the CUT. The total test length for DATE00 is at most 5,000 patterns. ITC98 implements for the same test architecture as DATE00 a symbolic algorithm computing the seeds. The test length for ITC98 is always 10,000 patterns. A method doing on-chip decompression using Golomb codes [5] does only pattern reordering and does no pre-processing for random-pattern testable faults. The compressed test data volume shown in column TCAD01 is significantly higher. A LFSR-based TPG using variable length seeds and reusing part of the scan chain of the CUT is added as a dedicated hardware structure to the CUT [22]. 10,000 random patterns and deterministic patterns encoded with the variable length seeds are applied to the CUT. The compressed test data volume in bits is shown in column TOC98. As in most other methods 10,000 random patterns were applied and the undetected faults were encoded using the RESPIN method. Column RESPIN shows the results.

Bench- mark	JETTA98 [15]	DATE00 [3]	ITC98 [10]	TCAD01 [5]	TOC98 [22]	RESPIN
s420.1	503	408	476	-	-	362
s838.1	3,246	1,452	3,432	-	-	1,925
s5378	759	3,262	3,424	14,085	-	618
s9234	11,766	24,700	7,410	22,250	5,346	8,006
s13207	10,796	-	12,600	41,658	5,877	1,963
s15850	11,826	-	15,886	40,717	6,316	5,139
s38417	71,491	-	73,216	92,054	16,797	31,020
s38584	11,529	-	43,920	104,111	3,996	3,627

Table 5. Encoded test information in bits of several decompression methodologies.

6 Conclusions

IN uses scan chains of idle cores to decompress test patterns for another core of a SOC. The RESPIN architecture may be integrated with small modifications into a chip with a test infrastructure supporting the access and isolation of cores during test. The embedded cores must employ full scan design and require a serial TAM interface and a parallel TAM interface. The ATE interface is the same as for a standard test procedure, except that ATE with less memory and lower bandwidth may be used to test the cores on the chip.

- The results of the presented on-chip decompression methodology achieves similar results as dedicated on-chip decompression hardware [22] and outperforms previously known methodologies reusing chip functionality [7, 10]. This implies, that often there is no need for a dedicated decompression hardware.
- Applied on the ISCAS89 benchmarks RESPIN reduces the test data volume to be stored by the ATE down to a fraction between 1 and 30% of a statically and dynamically compacted test set.

Acknowledgement

The authors wish to thank our students Georgi Outcharov and Pattara Kiatisevi for implementing the programs for the experiments, Jens Künzer for comments on the test architecture of RESPIN, and the reviewers for many invaluable comments.

References

[1] S. Boubezari and B. Kamiñska. A deterministic built-in self-test generator based on cellular automata structures. *IEEE Transactions on Computers*, 44(6):806–816, 1995.

[2] F. Brglez, D. Bryan, and K. Kozminski. Combinatorial Profiles of Sequential Benchmark Circuits. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, pages 1229–1234. IEEE, 1989.

[3] S. Cataldo, S. Chiusano, P. Prinetto, and H.-J. Wunderlich. Optimal Hardware Pattern Generation for Functional BIST. In *Proceedings of the Design Automation and Test in Europe (DATE)*, 2000.

[4] K. Chakrabarty and J. P. Hayes. On the Quality of Accumulator-Based Compaction of Test Responses. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(8):916–922, August 1997.

[5] 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*, 20, March 2001.

[6] C. L. Chen. Linear Dependencies in Linear Feedback Shift Registers. *IEEE Transactions on Computers*, C-35(12):1086–1088, December 1986.

[7] S. Chiusano, P. Prinetto, and H.-J. Wunderlich. Non-Intrusive BIST for Systems-on-a-Chip. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 644–651, Atlantic City, NJ, 2000. IEEE, IEEE.

[8] W. Daehn and J. Mucha. Hardware Test Pattern Generators for Built-In Test. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 110–113, Cherry Hill, NJ, 1991.

[9] D. Das and N. A. Touba. Reducing Test Data Volume Using External/LBIST Hybrid Test Patterns. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 115–122, Atlanta, NJ, 2000. IEEE.

[10] R. Dorsch and H.-J. Wunderlich. Accumulator Based Deterministic BIST. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 412–421, 1998.

[11] I. Ghosh, S. Dey, and N. K. Jha. A Fast and Low Cost Testing Technique for Core-based System-on-Chip. In *Design Automation Conference*, pages 542–547, June 1998.

[12] P. Goel and B. C. Rosales. Test Generation and Dynamic Compaction of Tests. In *Digest of Papers Test Conference*, pages 189–192, 1979.

[13] X. Gu, S. S. Chung, F. Tsang, J. A. Tofte, and H. Rahmanian. A fast timing closure technique for industrial use of logic bist. Presentation at the European Test Workshop (ETW), May 2001.

[14] I. Hamzaoglu and J. H. Patel. Test Set Compaction Algorithms for Combinational Circuits. In *Proceedings of the International Conference on CAD (ICCAD)*, November 1998.

[15] S. Hellebrand, H.-J. Wunderlich, and A. Hertwig. Mixed-Mode BIST Using Embedded Processors. *Journal of Electronic Testing Theory and Applications (JETTA)*, 12(1/2):127–138, 1998.

[16] V. Immaneni and S. Raman. Direct Access Test Scheme-Design of Block and Core Cells for Embedded ASICs. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 488–492, October 1990.

[17] B. Koenemann. LFSR-Coded Test Patterns for Scan Design. In *Proceedings of the European Test Conference (ETC)*, pages 237–242, München, 1991.

[18] E. J. Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel. Towards a Standard for Embedded Core Test: An Example. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 616–627. IEEE, 1999.

[19] M. Nourani and C. Papachristou. Structural Fault Testing of Embedded Cores Using Pipelining. *Journal of Electronic Testing Theory and Applications (JETTA)*, 15(1/2):129–144, August/October 1999.

[20] P1500 Scalable Architecture Task Force. Preliminary Outline of IEEE P1500 Scalable Architecture for Testing Embeddeb Cores. In *VLSI Test Symposium*, May 1999.

[21] J. Rajski and J. Tyszer. Accumulator-Based Compaction of Test Responses. *IEEE Transactions on Computers*, 42(6):643–650, June 1993.

[22] J. Rajski, J. Tyszer, and N. Zacharia. Test Data Decompression for Multiple Scan Designs with Boundary Scan. *IEEE Transactions on Computers*, 47(11):1188–1200, November 1998.

[23] Semiconductor Industry Association. *The International Technology Roadmap for Semiconductors (ITRS)*. International SEMATECH, Austin, TX, 1999.

[24] A. P. Ströle. Test Response Compaction using Arithmetic Functions. In *Proceedings of the VLSI Test Symposium (VTS)*, pages 380–386. IEEE, 1996.

[25] P. Varma and S. Bhatia. A Structured Test Re-Use Methodology for Core Based System-Chips. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 294–302, Washington, DC, October 1998.

[26] Y. Zorian. Testing the Monster Chip. *IEEE Spectrum*, 36(7):54–60, July 1999.

[27] Y. Zorian, E. Marinissen, and S. Dey. Testing Embedded-Core-Based System Chips. In *Proceedings of the IEEE International Test Conference (ITC)*, pages 130–143. IEEE, 1998.