Test Pattern Overlapping - a Promising Compression Method for Narrow Test Access Mechanism SOC Circuits

Ondřej Novák, Jiří Jeníček

Abstract **− This paper describes research results obtained in the field of test pattern compression and decompression. We refer the hardware test pattern decompression system DyRESPIN built-in on a System on Chip, which uses test patterns compressed by the compressing algorithm called COMPAS. COMPAS reorders and compresses test patterns previously generated in an ATPG in such a way that they are well suited for decompression by the scan chains in the embedded tester cores. We report improvements that have been done recently on COMPAS. COMPAS algorithm has to manipulate with enormous amount of data when compressing test sets of large circuits and the CPU time grows rapidly with the growing number of test vectors. The CPU time problem was solved by using a test vector initial encoding by sparse vectors and by using a dynamic structure for storing the pre-calculated parameters of candidate vectors to be used in the near future algorithm loops for overlapping with the actual scan chain content. This arrangement allows the algorithm to skip unnecessary computations. The improvements cause that the CPU time grows approximately linearly with the size of the tested circuit. DyRESPIN uses a built-in processor for test control, the embedded RAM memory for storing both the compressed test vectors and the partial reconfiguration bit streams and the FPGA part of the chip for the wrapped cores implementation. The highly compressed test vectors are transferred from the memory to those selected cores that are reconfigured into the embedded tester cores. The patterns are decompressed within the internal scan chains of the embedded tester cores and they are simultaneously fed into the parallel scan chains of the cores under test with the help of the Test Access Mechanism (TAM) and standard wrappers. After having tested the first cores under test the TAM of the SoC is partially reconfigured with the help of the partial reconfiguration bitstreams stored in the RAM memory and the till now untested cores are tested by those cores that start to serve as embedded testers.**

Index Terms — **Circuit testing, Testing, Memory management**

Manuscript received Fabtuary 6, 2008.

Ondřej Novák with the Technical University Liberec, Hálkova 6, 461 17 Liberec I, Czech Republic, E- mail: ondrej.novak@tul.cz.
Jiří Jeníček with the Technical University Liberec, Hálkova 6, 461 17 Liberec I, Czech Republic, E- mail: ondrej.novak@tul.cz.

Acknowledgment. The research was supported by the research grant of the grant IQS108040510 of the Czech Academy of Sciences.

I. Introduction

Deterministic test spares testing time and the on chip hardware overhead is low. However the test sizes has been pushing test costs up due to the necessity of using more powerful ATEs and if the test access mechanism (TAM) is narrow the test application time becomes to be critical, too. In order to minimize the data transfer through the TAM, *compacted* and *compressed* test sets are used. By the term *compact test set* is meant a test set, which is created in the automatic test pattern generator (ATPG) from test patterns by merging as many as possible patterns. An original test pattern usually detects one or more possible circuit faults and contains several don't care bits. The original patterns are merged in such a way that resulting patterns detect multiple faults and do not contain don't care bits while the test set fault coverage remains unchanged.

Test data compression is a non-intrusive method that can be used to compress the pre-computed test set to a much smaller test set, which is then stored in the ATE memory. An on-chip decoder is used to generate the original test set from the compressed one. Many contributions containing different decompression mechanisms were published; let us mention [1], [3], [5][7][18], [27], [34]. It is not straightforward to compare the compression methods because some authors demonstrate the efficiency on decompression of random resistant faults only and other authors compress and decompress the whole ATPG deterministic test sequence. The usefulness of a compressing algorithms and decompressing automaton is influenced not only by the compression ratio but also by the complexity of the decompressing automaton and by the computational complexity of the algorithm for finding the compressed test sequence.

Increasing number of transistors results in increasing ATPG computation time and memory consumption. Many of published test optimization techniques are dedicated to sequential test optimization. To handle time consuming test generation, it is often necessary to parallelize test generation process [35], [15], [36]. Concurrently generated ATPG output has to be than effectively compressed.

In this paper, we present results of our previous research done in the field of test pattern compression based on test pattern overlapping [16] and hardware decompression based on wrapper reconfiguration [26]. The results give us a possibility to construct a system that combines both of the mentioned methodologies.

II. COMPAS – TEST PATTERN COMPRESSION TOOL

The main idea is to maximally overlap those patterns that are serially shifted into the scan chain. This approach was firstly described in [6]. The method uses an algorithm for finding contiguous and consecutive scan chain vectors for the actual scan chain vector. These vectors are checked whether they match with one or more remaining test patterns, which were previously generated and compacted with the help of some ATPG and which were not employed in the scan chain sequence yet. Similar approach was used in [33]. The compacted test vectors were reordered by a heuristic algorithm to attain maximal overlapping. A disadvantage of the mentioned methods is that they are either computationally complicated and thus they are not usable for large circuits or the obtained amount of test data stored in an ATE is greater than the data amount in other compression methods. We present an algorithm, which speeds up the computation by searching for the successors of given starting pattern (usually the all zero seed) and which improves the compression efficiency by fault simulation after every test pattern application. This algorithm uses test vectors with don't care bits instead of the compacted ATPG test vector test set, which enables us to combine test pattern compaction and compression to be well suited with the decompression in a scan chain. The algorithm is implemented in the COMPAS (COmpressed test PAttern Sequencer) software tool. It speeds up and improves the algorithm [26] by taking into account possible future conflicts between overlapping patterns, it uses more efficient pattern coding and it remembers information that could be useful in future algorithm loops. COMPAS is able to prepare test sequences for the most complex circuits in short time. COMPAS can be used also for preparation of test sequences of cores under test (CUT) that are designed according the IEEE 1500 standard [23]. Test data can be effectively decompressed with the RESPIN test architecture [7]. This architecture reuses scan chains of different cores for updating the tested core scan chain content. Latest version of the algorithm used in [26] has been further enhanced to lower CPU time and memory consumption.

III. MEMORY CONSUMPTION IMPROVEMENT

Uncompressed test data generated by an ATPG are stored as a plain text file, each fault corresponds to a single vector in form of sequences of '0', '1' and 'X' characters standing for log. 0, log. 1 and unspecified bit (hereafter DC bit, DCB or 'X').

This data organization allows many concurrent ATPG processes. Each ATPG can generate test vectors for small group of faults or for single fault. Outputs of all ATPGs are then merged into a single file.

Size of the file can be a problem; because data files are very large for larger circuits (e.g. b19 from ITC99 benchmark set has more than 2.5 GB uncompressed test data).

For optimal algorithm decision it is necessary to load all test data at once into a computer memory, so new method needs to be developed for storing the test data in a computer memory. Simple loading of the file is not possible for large circuits.

A. New data encoding

One more stage of compression has to be performed instead of simple loading of text file into memory. First stage compresses plain text data from a file and stores them in a memory. Second stage uses compressed data from a memory to do pattern overlapping compression.

Three different encodings of test vectors are used in the program. Data produced by an ATPG are stored as a plain text: '0', '1' and 'X'. Each character is stored in on a hard drive as an 8bit char type. Large amount of uncompressed test vector data are encoded into two different forms when loaded into a computer memory, depending on which one consumes less memory.

The first encoding is a quite straightforward conversion of the eight bit character vector into two bits. By this encoding each 8 bit character is reduced to 2 bits, and 6 bits are saved (75% of memory).

The second encoding creates so called sparse vector, which means that only care bits and their positions are saved. 32 bit integer I used as a basic data type; one bit is used for actual value, and rest 31 bits are used to note the position. Scan chain with maximal length of 2^{31} can be encoded this way. DC bits are not stored at all. As not only one byte but four are used to encode a single care bit, this compression method is useful only if total amount of care bits is lower than 25%, otherwise it consumes more memory than the original vector. The first method can certainly compress to 25% of the original, a vector to be encoded more effectively by using the sparse vector than by the first approach must have less than 6.25% care bits. However, Tab. 1 with the numbers of DCBs in benchmark circuits shows that it should not be a problem.

It is decided for each vector separately which method should be used. So it could be guaranteed that at least 75% of memory will be saved. For larger circuits often more than 95% can be saved by a proper encoding.

IV. RUNTIME IMPROVEMENT

A. Algorithm description

At first, a Test Pattern List (TPL) together with the corresponding Undetected Fault List (UFL) is generated for the tested circuit. An ATPG tool that enables generating non-compacted test patterns has to be used. At least one three state test vector with bit values 0, 1 and X, where X means don't care value has to be generated for each considered fault. In this way we can distinguish, which pattern belongs to which fault.

TABLE 1:
CARE BIT PERCENTAGE IN TEST DATA OF DIFFERENT BENCHMARK CIRCUITS

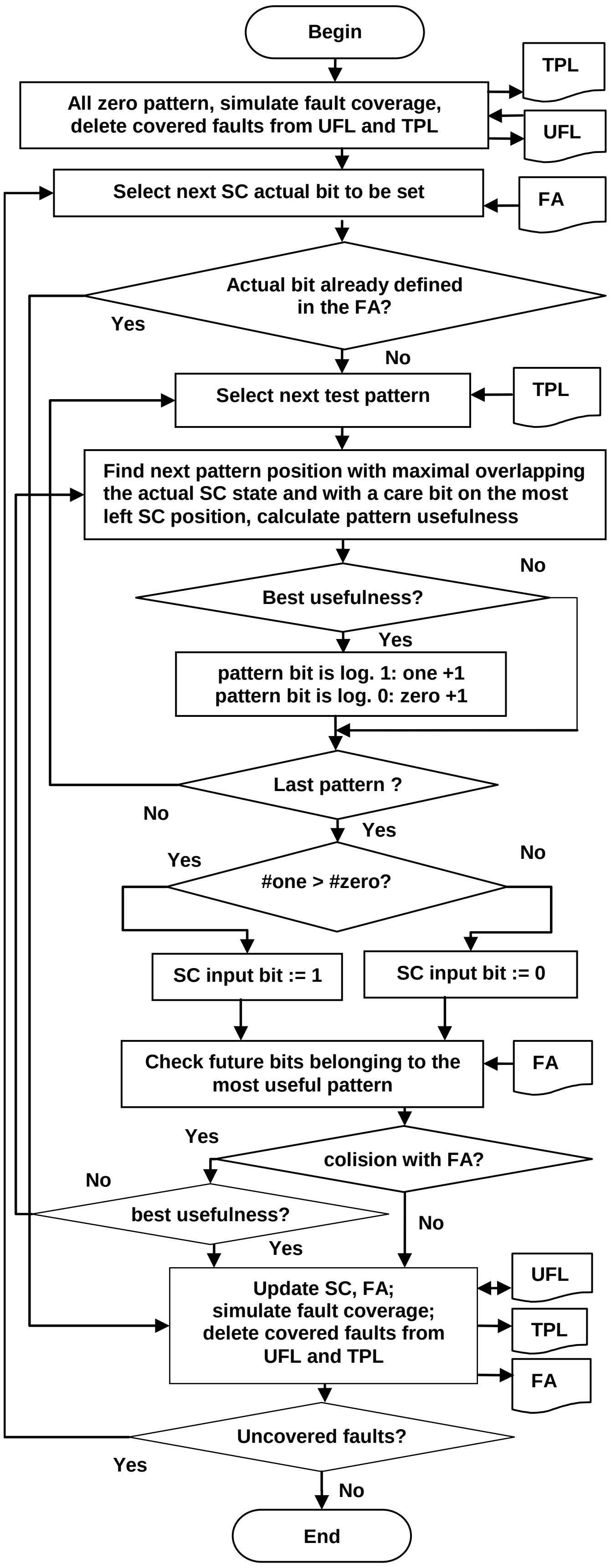
Circuit	Gate count	Care bits[%]
c17	6	56,36
c432	160	43,67
c499	202	82,32
c880	383	17,78
c1355	546	86,31
c1908	880	55,16
c2670	1193	7,92
c3540	1669	25,32
c5315	2307	7,39
c6288	2416	76,24
c7552	3512	13,1
s27_comb	10	45,09
s1196_comb	529	26,01
s1238_comb	508	26,48
s1494_comb	647	50,58
s5378_comb	2779	4,05
s9234_comb	5597	5,44
s13207_comb	7951	1,19
s15850_comb	9772	1,38
s35932_comb	16065	0,26
s38417_comb	22179	0,84
s38584_comb	19253	0,39

The main loop of the algorithm of finding bits to be stored in the ATE memory is described in Fig. 1. Let us suppose (without loss of generality) that the SC is reset before testing, which means that the all zero pattern is considered to be used as the first one (algorithm allows to start with any known scan chain state). The fault coverage of this pattern is simulated and the detected faults are deleted from the UFL, test patterns corresponding to the detected faults are deleted from the TPL. Then the algorithm tries to compact the test set by overlapping resting patterns with the actual scan chain state. The algorithm finds, whether log. 0 or log. 1 is better to be used as the next most left chain bit. To do this the algorithm finds positions of all patterns, in which the actual chain bits maximally overlap the pattern and for which the actual bit to be introduced into the SC has not a don’t care value. After finding the position the algorithm has to count the usefulness U of the treated pattern. The pattern usefulness U is calculated according to the following formula:

$$U = t *(overlapped\_cares + shift) + global\_cares$$

where *overlapped_cares* – the number of the pattern care bits that overlap the SC; *shift* – the amount of non-overlapped bits in pattern; *global_cares* - the global number of the pattern care bits; *t* – Experimentally fixed parameter; we obtained good results when we set *t = number_of_primary_inputs / 2.*

Then the algorithm compares the number of the most useful patterns with log. 1 on the actual position and the



Begin
TPL
All zero pattern, simulate fault coverage, delete covered faults from UFL and TPL
UFL
Select next SC actual bit to be set
FA
Actual bit already defined in the FA?
Yes
No
Select next test pattern
TPL
Find next pattern position with maximal overlapping the actual SC state and with a care bit on the most left SC position, calculate pattern usefulness
No
Best usefulness?
Yes
pattern bit is log. 1: one +1
pattern bit is log. 0: zero +1
Last pattern ?
No
Yes
Yes
No
#one > #zero?
SC input bit := 1
SC input bit := 0
Check future bits belonging to the most useful pattern
FA
Yes
colision with FA?
No
best usefulness?
No
Yes
UFL
Update SC, FA; simulate fault coverage; delete covered faults from UFL and TPL
TPL
FA
Uncovered faults?
Yes
No
End

Fig. 1: Pattern overlapping algorithm

number of patterns with log. 0 on this position. If the number of ones is greater than the number of zeros the input actual bit is fixed to log. 1 in the other case to log. 0. This way of setting the actual bit guarantees that a maximum number of the most useful patterns could be encoded. When searching for the most useful pattern we check whether the exercised pattern matches with bits which will be necessary to be generated in the future clock cycles because of some previously selected patterns. These bits are stored in a Future Array (FA) together with their effectiveness and pattern identification numbers If some position of FA is reserved for a logical value that is clashing with the exercised pattern bit value we compare the usefulness of both patterns and the winner is used in future considerations. After bit selection the fault simulation is performed and the faults and patterns, which correspond to the covered faults, are removed from the lists. If there are not remaining faults in the Undetected Fault List the algorithm is finished.

B. Proposed optimization

Basic principle of the compression method remains the same as in previous chapter, but several steps of the algorithm can be skipped, if they can not influence the solution.

One possible state of the compression algorithm is shown in Tab. 2. To make explanation easier, the basic version of algorithm [26] without bit prediction is used, as the principle of the optimization remains the same.

TABLE 2
EXAMPLE OF VALID ALGORITHM STATE

						Searched bit						
Step	5	4	3	2	1	0						
SC content						?	0	0	1	1	1	0
vector A			1	X	X	X	0	0				
vector B			1	X	X	1	0	0				
vector C					X	1	X	0	1	1		
vector D	0	X	1	1	1	1						

All remaining vectors are overlapped as much as possible with current scan chain state during search of the next compressed bit (marked with question mark). At the given moment, three vectors are overlapping (vectors A,B,C) and one vector (D) could not be overlapped. It is not possible to store DC bit in the compressed sequence, so only vectors 2, 3 and 4 are useful. They all have '1' at the current position, so value '1' is shifted into the scan chain and stored as the next bit of the compressed sequence. The fault simulation of the current scan chain state is performed after that, the detected faults and their corresponding vectors are removed from the memory. Than the algorithm goes to the beginning and tries to overlap all remaining vectors again.

It is important, that vector A has only useless DC bits in step 1 and in the two following steps (2 and 3). Those bits can not be contained in the solution; on top of that they will never collide with any other selected bit. Because of that it is possible to omit the calculation of the possibility of vector overlapping for vector A in steps 1, 2 and 3. The vector A is not useful for calculations in step 1 and 2; in step 3 it could be overlapped without collision. Vectors B and C have DC bits in the following steps, but during calculation it is not certain, if their bit (value '1') will be chosen as a solution. Vector D has care bits on the next position, so its overlap has to be evaluated. If a vector has a DC bit in the actual position, it is only necessary to evaluate how many DC bits follow the actual position. This computation is done for each new found sequence of DC bits in vector only once. It is also faster than finding how much is a vector overlapped. Every time when vector overlapping is evaluated, the program finds if DC bits series follows, and how long is this DC bit sequence. If there is at least one DC bit, the overlapping evaluation is omitted.

Vectors are stored in a dynamic structure from Fig. 2 according to the number of steps needed to reach a care bit. Only the vectors from entry '0' of steps_to_care_bit array are checked, others can not influence the solution. It is obvious, that it is necessary to store only distance to the next care bit, so that each vector will be saved only once. Solution is chosen after evaluation of all vectors in entry '0', and vectors are placed into the proper entries of steps_to_care_bit array according to the distance to the next care bit. After evaluation and replacement of all the vectors from entry '0', whole array is shifted one position to the right and the algorithm is ready for the next loop.

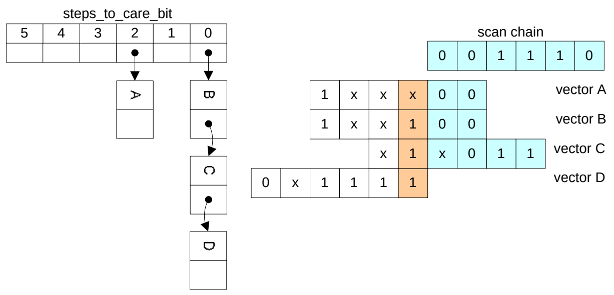


Fig. 2: Dynamic structure for calculation omit decision

Amount of the DC bits in the uncompressed test data file of several circuits from ISCAS85 and ISCAS89 benchmark set is noted in Tab. 1. The data contain a lot of DC bits, and the percentage of DC bits grows with the circuit size. That is why it is possible to skip a lot of calculations.

V. TEST ACCESS MECHANISM (TAM)

A test session can be controlled by a tester or by a BIST controller. It could be advantageous to use an embedded processor instead of a specialized controller with a RAM. As the RAM size is limited, the test set has to be as small as possible. Further testing speed improvement could be obtained by minimizing the amount of data transferred between the processor and the tested cores. From this reason it is worthwhile to send the compressed data from the processor to the decoders that are placed closely to the tested cores and to leave the decoders to decode the patterns independently on the processor activity. This arrangement can speed up testing as the clock frequency of the core flip-flops could be higher than the processor clock frequency and the processor can prepare next data during decoding the previous pattern (Figure 3). Another problem arises when using cores with the SCs that contain internal flip-flops; if we have to guarantee not corrupting test patterns by CUT responses and simultaneously catching all test responses we have to scan in and scan out the whole test pattern after each system clock application. The RESPIN (Reusing Scan Chains for Test Decompression) test architecture [7] solves both pattern decompression and reducing the data traffic between tester and CUT.

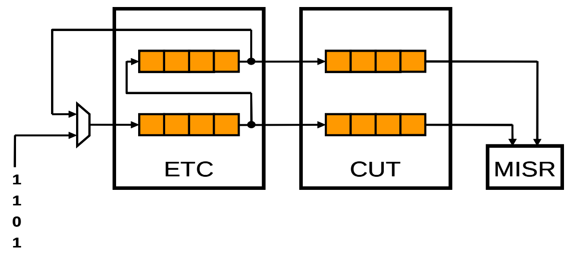


Fig. 3. ETC and CUT in the RESPIN architecture

The RESPIN architecture temporarily divides the circuit into the core under test (CUT) and the embedded tester core (ETC). The data transfer mechanism between the tester and ETC can be denoted as a narrow TAM as the demanded transfer capacity is low. The TAM between the ETC and CUT is wide as the data transfer is done parallel and on a higher clock frequency. The ETC chains are concatenated into a serial scan chain; a feedback tap connects the ETC last chain output with the first bit input through a multiplexer. According the multiplexer control input, ETC can either load a bit from the tester or shift the scan chain circularly. The parallel chains of the CUT are connected with the parallel ETC chain outputs. This test pattern updating mechanism guarantees that the patterns, which are shifted through the CUT SC during several test steps, are not mixed with the CUT responses. An additional multi input MISR connected to the SC outputs can be exploited for capturing all the test responses. The conditions for effective testing are: the ETC has at least the same number of chains as the CUT; the CUT chains are not longer than the corresponding ETC chains and the number of scan cells of the CUT and the total number of ETC scan cells incremented by one have not a common divider. If it is not possible to find an ETC core that fulfils the above mentioned conditions, more than one core can be used for creating the ETC.

A. Reconfiguration

The novel FPGA circuits are dynamically reconfigurable at runtime. These dynamically reconfigurable FPGA circuits have a capability to change the behavior of one part of the circuit; the rest part is fully operational without changes and without interruption. Generally, each memory-based FPGA can be reconfigured dynamically. In the currently known dynamically reconfigurable devices two techniques are used: "partial configuration" and "Multiple-context configuration memory" [31].

Reconfiguration of the TAM for a SoC testing seems to be an efficient exploitation of the partial reconfiguration capability of FPGAs. As the Atmel FPGAs can efficiently perform the fine grained reconfiguration we decided to use it for an implementation of the self-testable SoC (System on Chip) design. The diagnostic system uses RESPIN architecture which is based on the IEEE 1500 standard. The partial reconfiguration is used for connection among ETCs, CUT and the feedback multiplexer.

The main advantage of the proposed solution is that all the reconfiguration bitstreams are stored inside the chip. Thereafter the reconfiguration process can be controlled by the embedded processor and the only communication between the tested SoC and the external test supervisor is a request for execution the test and checking the results of the done tests.

VI. EXPERIMENTAL RESULTS

Fig. 4 shows the COMPAS CPU time improvement against [26]. The new algorithm performs better for larger circuits, and it corresponds with amount of DC bits. Average speedup is 114% for all measured circuits and 181% for circuits with more than 10.000 gates.

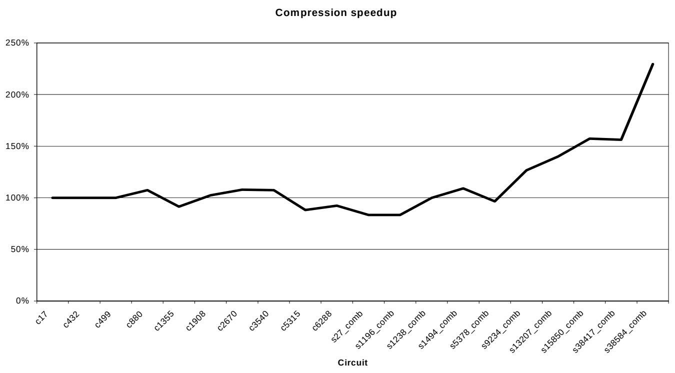


Fig. 4: Speedup of the compression part of the algorithm

Tab. 3 shows the resulting numbers of stored bits for some well known test pattern compression methods and for the proposed algorithms. In the second column we plotted the test data volume for ATPG vectors, which were compacted only [3]. Next column shows the number of stored bits for statistical coding of the test patterns from the previous column [1]. Next results correspond to a combination of statistical coding and LFSR reseeding [18]. Next columns summarize results of compression with parallel/serial scan chains [27], frequency directed codes [5]. The results for the method of Embedded Deterministic Test are presented in the next column [18]. The column RESPIN++ shows the numbers of bits stored in the ATE for the RESPIN++ architecture given in [32]. We can see that the number of bits, which are stored in a memory, is substantially lower for the proposed method than for other pattern compressing methods. We have to note that a majority of the tabulated pattern compression methods do not use a fault simulation after encoding a new test pattern (with the exception of the method [32]). These methods use compacted test sequences, the fault coverage was simulated during test pattern generation in the ATPG in the process of pattern compaction. The number of fault simulations in these cases corresponds with the total number of non compacted test patterns. In case of COMPAS and RESPIN++ the ATPG patterns were generated without any simulation, fault simulation is performed after a pattern encoding. The number of fault simulations is equal to the length of the final compressed sequence. Lengths of the compressed sequences are the same as in previous work [26], because both optimizations do not change the principle of the algorithm. That means that the results should be exactly the same, but due to optimizations the results should be obtained faster and with smaller memory footprint. This is true especially for larger circuits, because their test data generated by an ATPG contain large amount of don't care bits.

The experimental diagnostic system was built on the FPSLIC^{TM} AT94K40AL circuit. It is a dynamically reconfigurable programmable SoC, which integrates Atmel SRAM, FPGA and an 8-bit AVR processor [11].

The FPSLIC circuit is connected to PC through JTAG interface. A user is able to program both main parts of IC – program for AVR processor and/or static content of FPGA. Testing with the RESPIN architecture requires reconfiguring circuit cores several times during the test. Each core in the SoC is surrounded by the wrapper [14]. The wrapper allows connecting the core with the defined surrounding cores either in the functional mode or in the test mode. The Test Access Mechanism (TAM) takes care of the on-chip test pattern transport. The TAM and wrappers form the infrastructure for access to individual cores providing tests of all cores. Whereas the core wrapper is defined and standardized by the IEEE 1500 standard, the design of test access mechanism is excluded from this standard and assumed to be addressed by the SoC designer. Partial FPGA reconfiguration was used as an efficient way how to form the low area demanding TAM for multiple embedded core SoC design. The FPGA consists of a number of generic cells called LUTs. In our system the LUT is used for connecting the test core terminal and a LUT of the TAM. By this arrangement two LUTs are needed to form one wire interconnection between 1-bit core test input and output terminal in the FPGA.

The testing system uses an 8-bit AVR processor, an SRAM memory and a dynamic reconfigurable FPGA accessible both from the processor and from the FPGA. In the FPGA we programmed wrapped cores, the MISR, the controller and detached area of the TAM. The AVR processor was used for data processing, for handling the data with the hardware controller and for partial reconfiguration of the TAM before initiation of the core test. Test patterns together with TAM configurations were stored in the embedded SRAM. The processor controls the test scheduling and communicates with the hardware controller. The RAM is used for storing the compressed test sequence. For each test pattern the processor gives the controller a command to run the test cycle independently on the processor. This arrangement enables the hardware controller and the processor to work concurrently and to speed up the test. The hardware controller drives core wrappers and the TAM by the WSC signals. During the test cycle the AVR transports one test bit from the memory to the port tdi and informs the controller about availability and suitability of test data. At the end of the test session, the processor shifts data through the port tdo from the MISR where the responses were accumulated and compares the resulting signature with the sample one stored in the RAM (Figure 5). After finishing the first CUT test the TAM is partially reconfigured and the next core is assigned as a CUT and it is tested through a newly reconfigured ETC. As the granularity of configurable blocks of the FPGA is relatively fine only a small part of the configuration memory has to be replaced by a new content (In Fig. 4 denoted by the gray color).

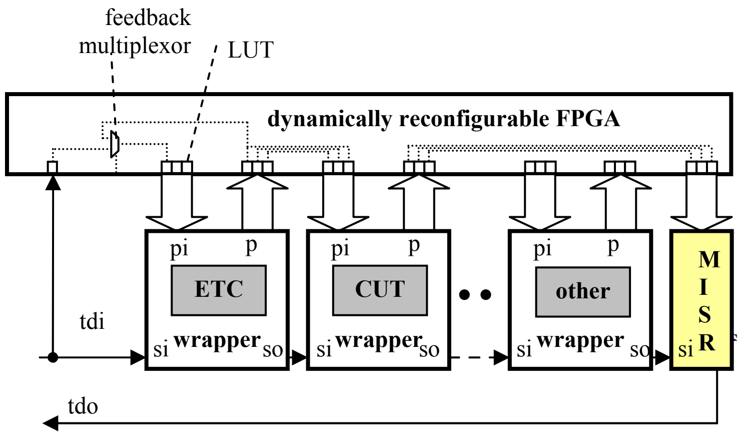


Fig. 5: An example of TAM configuration (given by dotted lines). The TAM is reconfigured by reprogramming LUTs of the reconfigurable FPGA blok

The ISCAS benchmark circuits (S298, S382, S444 and S1423) were used as cores in the experiment. The system with three cores S1423 designed in the SoC used 73% of the FPGA AT94K40 resources. Reconfiguration takes several thousands of clock cycles of processor. Number of clock cycles depends on the design to be reconfigured. In our case the reconfiguration time is less than 1 ms in case of 4 MHz processor clock. The circuit has 36 Kbytes of available RAM memory (20 – 32 Kbytes for program and 4 – 16 Kbytes for data). The size of one reconfigurable bitstream, which was used in the diagnostic system, was 2 Kbytes. The more cores are used in RESPIN architecture the more reconfigurable bitstreams are needed for arranging the ETC–CUT structure. Nevertheless the spent RAM memory amount was acceptable. In case of lack of the RAM memory the bitstreams can be reloaded from a PC. The test time depends on the longest parallel chain and on the number of bits of the compressed test. In our

TABLE 3
COMPARISON OF MEMORY REQUIREMENTS FOR DIFFERENT TEST PATTERN COMPRESSION TECHNIQUES

Circuit name	MinTest [3]	Stat. Coding [1]	LFSR Reseed-ing [10]	Illinois Scan [15]	FDR Codes [5]	EDT [17]	RESPIN++ [18]	COMPAS (proposed)
	# of bits	# of bits	# of bits	# of bits	# of bits	# of bits	# of bits	# of bits
s13207	163,100	52,741	11,285	109,772	30,880	10,585	26,004	4,024
s15850	58,656	49,163	12,438	32,758	26,000	9,805	32,226	7,737
s38417	113,152	172,216	34,767	96,269	93,466	31,458	89,132	21,280
s38584	161,040	128,046	29,397	96,056	77,812	18,568	63,232	6,675

case the test time is about 0.3 ms for the best possible clock frequency of the FPGA (40 MHz).

VII. CONCLUSION

The COMPAS compression tool demonstrates that it is possible to apply the method of test pattern compression through pattern overlapping for relatively large circuits and that the resulting test data volume is kept very low. COMPAS uses as input test patterns non compacted original ATPG test vectors with don't care bits. The patterns are overlapped and the resulting test sequence can be decompressed by the scan chain. The decompressed patterns are simulated by the fault simulator whether they cover any other additional fault. This mechanism reduces the number of test patterns that have to be used for testing since the interleaving patterns that appear in the scan chain between the original patterns cover the random testable faults. These faults are usually tested by the random pattern sequence in mixed/mode testing algorithms and the proposed method avoids using this random testing phase. We have solved the problem of long CPU time for enumerating the compressed test sequence by multiple usage of test bit usability evaluation during the process of finding the test sequence and by skipping pattern recalculation for cases when don't care bit groups are present in the patterns. This was enabled by using a concatenated list of pattern pointers. Problem of extreme memory consumption has been solved by using two new data encodings during the compression. New test data encoding effectively reduces memory footprint of the COMPAS program to less than 25% of the original. The algorithm is also capable of compression of data generated by concurrently running ATPG processes

The proposed method of compression and compaction of test patterns is very well suited for testing combinational circuits with Boundary Scan because it does not require any additional hardware for test pattern decompression. It can be used also for testing sequential cores with multiple scan chains. To do this we can use the RESPIN architecture. For the use of the compressed test sequence in the multi scan chain system the sequence is reordered in order to be correctly decompressed within the RESPIN architecture. Following the IEEE 1500 standard [23] we do not require extra hardware with the exception of one multiplexer and a feedback wire in every core. The sequence generated by COMPAS can be used for less time consuming sequential core testing than it is possible in the mixed-mode testing approaches [26].

We have verified that the proposed diagnostic system is applicable on a SoC. We have placed the system together with simple functional cores on the AT94K FPSLIC circuit. The diagnostic system uses the dynamic and partial reconfiguration feature of the embedded FPGA. This is advantageous because it saves resources of the FPGA devoted for switching the TAM busses. For larger cores the system can be built on the large Xilinx FPGA circuits with embedded processor and RAM memory block. The property of dynamic reconfiguration of the FPGA part could be an advantage that can save the FPGA resources. We can conclude that the diagnostic system is well suited for a SoC architecture with a processor, RAM block embedded FPGA and ASIC. The memory requirements for storing the test data are lower than it is in case of other comparable methods; the test time is very low, too.

REFERENCES

[1] Abhijit Jas, Jayabrata Ghos-Dastir, and Nur A. Touba: Scan Vector Compression/Decompression Using Statistical Coding. Proc. VTS 1999

[2] Bayraktaroglu, I., and Orailoglu, A.: Decompression Hardware Determination for Test Volume and Time Reduction through Unified Test Pattern Compaction and Compression. Proc. of VTS 2003

[3] Bernhart et al.: OPMISR: the foundation for compressed ATPG vectors. Proc. ITC, 2001, pp. 748-757

[4] Brglez, F., Bryan, D., Kozminski, K.: Combinational Profiles of Sequential Benchmark Circuits. Proc. of. Int. Symp. on Circuits and Systems, 1989, pp. 1929-1934

[5] C Chandra, A. – Chakrabarty, K.: Frequency/Directed Run Length (FDR) Codes with Application to System/on/Chip Test Data Compression. Proc. VTS 2001, pp. 42-47

[6] Daehn, W., Mucha, J,: Hardware Test Pattern Generation for Built-in Testing. Proc. of ITC, 1981, pp. 110-113

[7] Dorsch, R. and Wunderlich, H-J:Reusing Scan Chains for Test Pattern Decompression.Proc. IEEE ETW, 2001, pp.24-32

[8] Hellebrand, S., - Liang, H.G. – Wunderlich, H.J.: A mixed mode BIST scheme based on reseeding of folding counters. Proc. of ITC, 2000

[9] http://direct.xilinx.com/bvdocs/userguides/ug070.pdf. [cit 9.5.2006]

[10] http://iko.kes.vslib.cz, [cit 10.27.2006]

[11] http://www.atmel.com/dyn/resources/prod_documents/2818s.pdf [cit 20.5.2006]

[12] http://www.cerc.utexas.edu/itc99- benchmarks/bench.html

[13] http://www.cerc.utexas.edu/itc99-benchmarks/bench.html

[14] IEEE Computer Society. IEEE Standard Testability Method for Embedded Core-based Integrated Circuits - IEEE Std 1500-2005. IEEE, New York, 2005.

[15] Irion, A.; Kiefer, G.; Vranken, H.; Wunderlich, H.-J. : Circuit Partitioning for Efficient Logic BIST Synthesis, Proc. DATE, 2001, pp.88-93

[16] Jenicek J. J., Novak O :Test Pattern Compression Based on Pattern Overlapping**,** In *Design and Diagnostics of Electronic Circuits and Systems*. Los Alamitos: IEEE Computer Society, 2007, pp. 29-34

[17] Koenemann, B.: LFSR – coded test patterns for scan designs. Proc. Europ. Test Conf., Munich , Germany, 1991,

[18] Krishna, C.V., Touba, N.A.: Reducing Test Data Volume Using LFSR Reseeding with Seed Compression. Proc. of ITC 2002, pp321-330

[19] Lee H. K., and Ha, D. S.: HOPE: An efficient parallel fault simulator. Proc of the IEEE Design Automation Conference, pp. 336-340, June 1992

[20] Lee H. K., and Ha, D. S.: On the generation of test patterns for combinational circuits. Technical Report 12_93, Department of Electrical Eng., Virginia Polytechnic Institute and State University

[21] Li, L and Chakrabarty K.: Test Data Compression Using Dictionaries with Fixed-Length Indices. Proc. VTS, 2003

[22] Li, L and Chakrabarty K.: Test Set Embedding for Deterministic BIST Using a Reconfigurable Interconnection Network. IEEE Trans. on Comp. Aided Design of IC, Vol. 23, No. 9, Sept 2004, pp.1289-1305

[23] Marinissen, E. J.- Zorian, Y. - Kapur, R. Taylor T., and Whetsel. L.:Towards a Standard for Embedded Core Test: An Example. Proc. of ITC, pp. 616–627. IEEE, 1999.

[24] Marinissen, E.J., Arendsen, R., Bos, G.: A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores. Proceedings IEEE, ITC, 1998

[25] Novák, O., Nosek, J.: Test Pattern Decompression Using a Scan Chain, Proc. of IEEE International Symposium on Defects and Fault Tolerance in VLSI Systems 2001, pp. 110 – 115.

[26] Novak, O., Pliva, Z., Jenicek, J., Mader, Z., Jarkovsky, M.: Self Testing SoC with Reduced Memory Requirements and Minimized Hardware Overhead. Defect and Fault Tolerance in VLSI Systems, 2006. Proc. of DFT'06. pp. 300 – 308

[27] Pandey, A. R. – Patel, H. J.: Reconfiguration Technique for Reducing Test Time and Test Data Volume in Illinois Scan Architecture Based Designs. Proc. IEEE VLSI Test Symp, 2002, pp. 9-15

[28] Pandey, A. R. – Patel, H. J.: Reconfiguration Technique for Reducing Test Time and Test Data Volume in Illinois Scan Architecture Based Designs. Proc. IEEE VLSI Test Symp, 2002, pp. 9-15

[29] Rajski, J. et al.: Embedded Deterministic Test . IEEE Trans. on CAD, vol. 23, No. 5, May 2004, pp. 776-792

[30] Rao, W., Oraiologlu, A.: Virtual Compression through Test Vector Stitching for Scan Based Designs. DATE 2003

[31] Scandaliaris, J., Moreno, J.M., Cabestany, J., Buttel, P., Rachet, A., Kadlec, J., Hermanek, A., de Saint Romain, D., Habay, G., Donati, A.: A General Design Flow for Dynamically Reconfigurable FPGAs (D_FPGAs). http://www.reconf.org/Files/Publications/RAW03_UPC.pdf [cit 22. 5. 2006]

[32] Schafer, L. - Dorsch, R.- Wunderlich, H.J.: RESPIN++- Deterministic Embedded Test. Proc. European Test Workshop, 2002, pp. 37-42

[33] Su, C., and Hwang, K.: A Serial Scan Test Vector Compression Methodology. Proc. ITC 1993, PP. 981-988

[34] Wolf, F. G. and Papachristou C.: Multiscan-based Test Compression and Hardware Decompression Using LZ77. Proc. of ITC 2002, pp. 331-339

[35] Wolf, J.M.; Kaufman, L.M.; Klenke, R.H; Pylor J.H.; Waxman, R.: An Analysis of Fault Partitioned Parallel Test Generation, IEEE Trans. On Computer-Aided Design of ICs and Systems, Vol. 15, 1996

[36] Wu, D.M.; Lin, M.; Reddy, M.; Jaber, T.; Sabbavarapu, A.; Thatcher, L.: An Optimized DFT and Test Pattern Generation Strategy for an Intel High Performance Microprocessor, Proc. Int. Test. Conf., 2004, pp. 38-47