VecTHOR: Low-cost compression architecture for IEEE 1149-compliant TAP controllers

Sebastian Huhn*† Stephan Eggersglüß*† Rolf Drechsler*†

*University of Bremen, Germany
†Cyber-Physical Systems, DFKI GmbH
huhn,segg,drechsle}@informatik.uni-bremen.de

Abstract—This work presents a new dynamically configurable compression architecture to be integrated directly into the test access mechanism of System-on-Chip (SoC) designs using IEEE 1149 compliant interfaces. The proposed technique reduces the test data volume without losing the full legacy support, no extra IO pins are needed and the additional allocated hardware resources are negligible. Particularly, this technique is suitable for board as well as in-field testing, which both use typically a Test Access Mechanism (TAM) like IEEE 1149. Here, strong memory limitations exist on the test equipment, which restrict the testing or debugging capabilities for complex designs. Various benchmarks for random test data, representing highly pre-compressed test data, as well as fully-specified test data for selected industrial circuit designs were run and discussed to evaluate this new approach. These experiments clearly show a high test data volume reduction. Additionally, a noticeable reduction of the overall number of required test cycles are achieved for most of the test cases.

I. INTRODUCTION

For several years, the design and fabrication of Integrated Circuits (ICs) don’t pursue solely the target to produce ICs, which only fulfill one single task but complex ICs are designed, suitable for several comprehensive tasks at once. For this purpose, whole SoC designs with several nested modules are realized. Due to this increasing complexity, there is also a high risk of physical defects accrued while manufacturing - not only for this reason production tests are indispensable. Besides these production tests, the high modularity of the SoC designs strictly requires extensive board and in-field testing capabilities. The Test Data Volume (TDV) effects the consumption of limited memory resources within the test equipment. This means that the number of tests is limited, which can be loaded simultaneously to the test equipment. Thus, the test costs increase, which typically claim a high share of the overall production costs. Particularly, the memory resources are often stronger limited in later phases like board and in-field testing or while debugging, e.g. test failure analysis, which can lead to complexity margins in the test procedures.

Several research works, e.g. [1]–[8], exist, which embed compression hardware into the design. These approaches are quite expensive regarding the hardware overhead and often depend on additional IO pins. Furthermore, most of these compression techniques require test data, which have to fulfill certain criteria to be compressible, e.g. large number of X values. Thus, these compression techniques are mostly not suitable for Functional Verification (FV). FV addresses test requirements for complex SoC designs with numerous modules, e.g. for testing inter-module communication or power-up, initialization or trimming sequences. In case of high modular SoC designs, these FV test cases or the resulting FV test data respectively tend to be extensive, which is often conflicting with the limited memory resources. Currently, no generic compression techniques for FV test data are available to decrease the steadily increasing test costs to the best of our knowledge.

Generally, the overall number of top-level IO pins is limited, which implies that only a subset of IO pins from the sub-modules is routed to the top-level. Thus, only the top-level pin-set is accessible in later test phases, e.g. board or in-field testing. Due to the increasing number of modules within SoC designs, ensuring the accessibility of every single module is a challenging task. To tackle this problem in those SoC designs, Test Access Port (TAP) controllers are used as a general TAM. Typically, a highly reduced clock frequency is applied while transferring data via this TAP, which means that the TDV or the number of test cycles highly affects the overall costs. Due to presence of these TAMs, they are suitable for being combined with the compression architecture as long as the full legacy support of underlying access protocol is not violated and the full legacy support is provided.

This work proposes Test Vector Transmitting using enhanced compression-based TAP controllers (VecTHOR), a new compression technique to be integrated in standardized TAP controllers. VecTHOR uses a code-based compression technique, which is able to involve fixed code-words as well as dynamically configured code-words for handling heterogeneous test data. Thus, this technique can be applied to reduce the TDV of existing test sequences measurably without effecting the logical test behavior itself. Thus, no re-verification of this test data is required. However, this architecture is not specifically meant for Automatic Test Pattern Generation (ATPG) test data. It is meant to be applied for FV tests while board and in-field testing as well as for the test failure analysis using decided debugging equipment. Typically, all these tests access the Circuit Under Test (CuT) via a standardized TAP controller like IEEE 1149. Thus, all proposed modifications of the TAP controller are completely compliant to the IEEE 1149 standard and a full legacy support is still ensured. In fact, it is possible to control the compression hardware completely by the default pin set. As a result, no additional IO pins are required. Finally, a suitable retargeting framework is drafted to complete this new compression architecture VecTHOR.

Several experiments show that VecTHOR achieves a noticeable TDV reduction on industrial as well as high entropic test data. These experiments show that this technique allows to reduce the overall number of required test cycles while processing fully-specified test data as well. Furthermore, the embedded compression hardware causes only a negligible overhead in the design itself, which has been proven by synthesizing the extended design.

The structure of this paper is as follows: Section II distinguishes this paper from related works. Furthermore, Section II also describes the general idea and the expected benefits of this approach. In Section III, a brief introduction into FV is given, especially accessing a circuit for transferring test data via test access mechanisms. Afterwards, Section IV presents the developed extension of a reference TAP controller, suitable to realize a decopression-based data interface, Section V drafts an algorithm for retargeting the test sequence to take advantage of the proposed compression technique. A collection
of experimental results is shown in Section VI. Finally, Section VII discusses possible future work and summarizes the drawn conclusions.

II. RELATED WORKS

Various works have been published with different approaches focusing on the demands of ATPG, Embedded Deterministic Test (EDT) [8] or of scan chain data compression [1], [5]. Generally, EDT techniques achieve a very high compression ratio, which mostly scales with the share of unspecified values (X-values) in the incoming test data. This compression ratio varies strongly depending on the test data. For fully-specified test data is used for instance in FV, EDT cannot be applied. Furthermore, EDT generally requires access to designated IO pins. Thus, in case of SoC designs embedding sub-modules with EDT, these IO pins are mostly not accessible at the top-level. Consequently, the access is also limited while board testing or debugging using dedicated debug equipment. Due to this fact, TAP structures are still more needed in these scenarios, which allow tunneling EDT data to a specific sub-module within the SoC design as well.

It follows a brief overview of the published methods for TDV compression for ATPG test data. The authors of work [3] proposed a static encoding, which was extended by taking advantage of a static run-length encoding [1]. This run-length encoding allows to handle repeating bits within the test vector efficiently. The proposed method of paper [9] uses the well-known Huffman-Encoding or even the powerful LZ77-algorithm [5], which is based on a dynamically growing dictionary. However, significant more hardware resources are allocated. The authors in [6] introduce a new way of encoding - the Golomb-Code -, which takes advantage of previous parts of the test sequence. Additionally, a powerful method of compressing multiple scan chains by a dictionary-based approach was published in [4]. In particular, [7] proposes a new compression technique, which works independently of Don’t Care values. Thus, no Don’t Care values have to be injected by the ATPG tool before transmitting the test data.

**Concurrent-JTAG (CJTAG)** [2] was designed to accelerate FV using TAP structures and is already realized by industrial implementations. However, CJTAG requires highly modified devices regarding their TAP controllers, so that the compliance with standardized JTAG is no longer ensured. Furthermore, two additional IO-pins must be embedded in the top-level. CJTAG achieves the speed-up by parallelization necessitating structural requirements for the CuT as well as for test sequences. Additionally, the authors in [10] propose a compression scheme for FPGA configuration bitstreams, which have to be integrated in-between the TAP and the FPGA core. Due to the fact that such a bitstream file is dominated by trailing zeros characteristically, this hardware bases on a Run-Length Encoding, which is suitable for this specialized application field only.

Here, we focus on serial test data transfer into the CuT via a centralized JTAG controller: It is targeted to develop an extended TAP controller, which applies a new proposed compression technique on incoming Data Register (DR) data. This new proposed technique provides the full legacy support for standardized JTAG operations without increasing the complexity of the controller significantly. The proposed compression architecture can be applied on any data, e.g. fully-specified FV test data, for which most of the previously published techniques are not applicable.

III. PRELIMINARIES

In the context of test data generation, different methodologies can be invoked: One very common technique to generate test vectors is ATPG. Depending on the referenced fault model, only logical knowledge of the circuit structure is included into this test generation process. In contrast to this methodology, test vectors generated in the context of FV try to model real use case scenarios, e.g. a chip initialization for the test setup, in later application. Such a scenario can be derived out of the circuit’s specification and probably consists of a whole start-up sequence or even more decided functions like some kind of coupling between devices or a protocol negotiation for the case the CuT implements a network controller. As initially mentioned, both ATPG and FV target the identification of physical faults, which were potentially caused within the production process and threatening the correct function.

Independent of the test data type, at least one communication channel, e.g. between a SoC containing several sub-modules and the test equipment, has to be available. Since strong limitations regarding the number of IO pins exist, it is not possible to route every IO pin of each sub-module to the top-level entity. For that reason a centralized master TAP is integrated into the top-level: Such a TAP provides an access port and is managed by the TAP controller, which implements a specific interface protocol. One very common protocol is implemented within a Joint Test Action Group (JTAG) controller: This interface consists of only five pins, which have to be accessible from outside: TCK, TRST, TDI, TDO, TMS. This JTAG is standardized within IEEE 1149.1 The signal Test Clock (TCK) provides the reduced clock for test purposes and Test Reset (TRST) implements a way to reset the TAP controller. Test Data-In (TDI) interface pin is used to transfer data into the device. In contrast, Test Data-Out (TDO) allows to transfer data out of the circuit. Finally, the Test Mode Select (TMS) signal controls the Finite State Machine (FSM) of the complete TAP controller. Only very few IO pins are occupied by using JTAG. Every additional pin would increase both production costs and also the necessary design effort. In case of hierarchical SoC designs, these capabilities can be used to access specific sub-modules.

The test data has to be converted into a bit string to be loaded into the CuT. After this conversion has been done, all of these bits are transferred strictly serialized via TDI into the TAP controller. As prerequisite, the controller must be in the correct state, whereby all state transitions are synchronized using TCK. JTAG implements a strict separation between loading IR data, i.e. data encoding JTAG instruction and DR data, i.e. payload data like test data. In the IEEE 1149 standard, some JTAG instructions are defined, e.g. bypass for daisy-chaining. Every single instruction can be addressed by a unique opcode, which is reserved with respect to the global instruction length.

After the complete test sequence has been transferred, the data processing by the CuT is started. Depending on the type of test, it is possible that all test data is stored in one exposed Test Data Register (TDR), which is accessible by the CuT as well. For example, this TDR can contain different bit fields, which represent memory addresses, control signals or direct encoded opcodes of the CuT instruction set.

IV. DECOMPRESSOR-BASED TAP-CONTROLLER

This section describes the necessary extensions, which have to be realized for establishing VecTHOR within a custom design using a IEEE 1149 TAP at top-level. This includes a mechanism to activate the new compression techniques, a code-based Dynamic Decompressing Unit (DDU) for received test data and an instrument to configure the DDU based on the test data to be processed, all within the TAP controller. Furthermore, a retargeting mechanism is required to determine a suitable configuration for the DDU as well as pre-processing test data wrt. the current decompressor configuration. This retargeting mechanism has to be integrated into the existing test data flow. In particular, the following items are realized:
1) In Subsection IV-A, two further JTAG instructions are integrated, which activate the data compression by compr_data and the configuration preloading by compr_preload. Therefore, the FSM of the underlying TAP controller has to be extended. Additionally, a further compression technique is developed in this work: μ-compr, which allows to take advantage of test data previously sent.

2) A suitable decompressor unit is designed in Subsection IV-B, i.e., for handling the substitution between compressed and decompressed bit strings, and connects this unit with the TDR.

3) In the next Section V, a suitable framework is established, which is able to process an incoming test vector automatically: At first, in Subsection V-A a suitable configuration for the DDU is derived out of the test vector. Afterwards, in Subsection V-B the incoming testdata are processed in such a way that valuable parts of the original data are replaced wrt. the DDU configuration.

In particular, the above mentioned μ-compr extension consists of a partial run-length encoding of complete code-words, which improves the compression ratio even more. Furthermore, it reduces the number of additional needed test cycles compared to the compr technique.

A. TAP-FSM extension

One very valuable property of this compression technique is that all changes performed to the interface are completely transparent towards legacy compatibility. Only two additional opcodes 0110 and 0100 are reserved, which represent the new JTAG instructions compr_data and compr_preload. Here, compr_preload allows to load the determined configuration to the DDU. If neither compr_data nor compr_preload instruction is loaded at all, the FSM is traversed normally as shown in Figure 1 and the processed data is not affected by any decompression. For the case that the instruction compr_data or compr_preload is selected, a TRST trigger or even unloading this instruction by reselecting leads to normal operation mode again.

![Fig. 1: FSM of IEEE 1149: TMS @ edges](image)

The FSM of the legacy JTAG is shown in Figure 1. In comparison, the modified FSM is visualized in Figure 2 and contains two additional states compr_dr and compr_exit, which can be reached while the namely compr_data instruction is selected. In general, the transfer of compressed data can be described by the following five steps separated by time $t_i$:

1) After the instruction compr_data has been loaded within the Load_IR phase, the state capture_dr is reached at

$t_{i-1}$. Now, TMS is assigned to 0, so that successor state compr_dr is selected at $t_{i-1}$.

2) TDI is captured on a rising edge of TCK at time $t_i$ and is interpreted as a chunk of the overall Compressed Data Word (CDW). This capturing process is repeated $n$ times up to $t_{i+n}$ as long as the FSM remains within the compr_dr state. Meanwhile, every received chunk gets stored into an exposed Compress Register (CR). The length of this register determines the chunk size $l_{CR}$. As a sanity check, exceeding this boundary leads to a Test Logic Reset (TLR), i.e., $l_{CR} \leq n$ is no longer valid.

3) Afterwards, to complete a CDW, TMS has to be changed to 1 causing a state transition to compr_exit. Within compr_exit state, the TAP-controller applies the decompressor on CR data and writes the Uncompressed Data Word (UDW) to the exposed TDR within one test cycle $t_{i+n+1}$.

4) Depending on the TMS signal at time $t_{i+n+2}$, the FSM reaches the state update_dr by leaving compr_dr phase with $TMS = 1$, otherwise a further CDW can be processed directly by driving $TMS = 0$, so the FSM passes over to compr_dr again.

5) In case of using the μ-compr technique while compr_data is selected, the transition between compr_exit and update_dr with $TMS = 1$ at $t_{i+n+2}$ has to be enriched by evaluating the TDI signal: If $TDI = 0$ holds, then nothing changes. If $TDI = 1$ holds, then the state compr_exit will be passed through a loop to itself over and over again as long as $TDI = 1 \land TMS = 1$. Every cycle iteration decompresses the last received CDW again and writes the mapped UDW to the exposed TDR.

Additionally, the modified FSM in Figure 2 shows that the DDU configuration invokes both new states analogously. Thus, all received configuration data is stored at the DDU register instead of the exposed TDR.

B. Implementation of TAP-decompressor

Besides the required FSM extension, another important aspect of the proposed compression architecture is the decompressor, which realizes a mapping function $\Psi(CDW) \rightarrow UDW$ between one successfully received CDW with length $c$ with $c < CS$ and a specific UDW with length $u$ with $u \in \{1, 4, 8\}$. Additionally, the function $B(CDW \rightarrow \beta$ maps every CDW to a scalar value $\beta$, which represents a metric calculated by $u - c$ and determines how valuable it would be to use this replacement - so called benefit of this replacement.
Additionally, a mechanism has to be established, which ensures the completeness of the mapping function $\Psi$. This means that single bits, which are not directly coverable by replacements, have to be handled individually.

Consequently, the decompressor unit receives the CDW after the FSM reached the state compr_exit. The maximal length of the CDW is determined by Chunk Size (CS). For instance, a CS value of 3 offers a good trade-off between the number of possible encodings and the hardware overhead. For that reason, CS is assumed to be 3 in this work. In general, the underlying technique can be scaled with higher CS easily. For every single increment of CS, overall $2^{CS}$ additional ones are available.

Using CS = 3 allows the following $\sum_{i=0}^{3} 2^i = 15$ possible CDW:

- $\emptyset$, ‘00’, ‘1’, ‘000’, ‘01’, ‘10’, ‘11’

Table I shows an exemplary realization of $\Psi$ representing the default DDU configuration, which has been used for the experiments of Section VI. In the exemplary implementation, it is possible to configure up to $2^{CS}$ CDW dynamically. This implementation contains two special UDWs ‘0’ and ‘11’ with even negative $\beta$ values, which are mapped to a single bit 1 and 0 respectively. While using VecTHOR, it is possible that the retargeting algorithm inserts a Single Bit Injection (SBI) for covering a ‘0’ or ‘1’ at a specific position in the test vector, which could not be covered otherwise. Thus, SBIs are important to ensure the completeness of the compression technique, i.e. this ensures that every sequence can be successfully processed.

Table II applies compr technique on example data test clock, reset signal and internal states of TAP controller’s FSM, particularly shift_dr and compr_exit as described above.

Another important aspect of this method is the small hardware overhead for the synthesized design. For this reason, the original legacy design, the modified one supporting the new proposed compr technique as well as the one supporting the extended $\mu$-compr technique have been synthesized. Due to the fact that the legacy TAP controller allocates a small number of resources only, the number of additional flip flops and gates, which are required by the proposed techniques, is quite negligible.

C. Exemplary application

Finally, the compr technique is demonstrated in Table II. Due to the page limitation, the $\mu$-compr and the DDU configuration are not described here, which means that the initial $\Psi$ is assumed. The compr technique is applied on an uncompressed test vector containing 18 bits. For this example, the given exemplary mapping function $\Psi$ in Table I is assumed as well as the drafted retargeting scheme presented by Algorithm 2 in Section V. Every single bit of the incoming test data is assigned to a byte and bit index. As shown, VecTHOR identifies designated parts of the uncompressed vector and encodes them with suitable CDWs, which are highlighted in Table II by underbraces. Due to the fact that the overall bit number is not completely divisible by 8 or even 4, the two last bits at position 2.0 and 2.1 have to be handled as SBIs. This example shows that SBI enables processing every kind of test data to a sequence of CDW replacements successfully. Finally, this compr techniques generates a compressed bit string, which contains only 13 bit. Accordingly, a compression ratio of nearly 28% is achieved in this example by integrating VecTHOR into the TAP mechanism.

V. RETARGETING FRAMEWORK

The following section drafts an algorithm to configure the DDU in subsection V-A and subsection IV-B an algorithm to retarget existing test data wrt. to the DDU configuration.
A. Automatic DDU configuration

The algorithm 1 shows the underlying procedure for determining a suitable configuration of the DDU. At first, the original vector $\Omega$ is serialized and processed in such a way that all sub-sequences with length of $u$, i.e. a supported UDW length, are handled separately (line 2 ff.). The occurrences of possible UDW permutations are counted within a data container $\Gamma$. Afterwards, the entries are post-processed by removing existing intersections (line 5 to 15). Within this step, the benefit of all pairwise intersections are evaluated (line 9) and the less valuable one is removed by decreasing the corresponding counter in $\Gamma$. Finally, $\Gamma$ is sorted by decreasing number of occurrences (line 16) and the most valuable $2^{2\gamma}$ ones are emplaced in the DDU configuration (line 17 ff.) by invoking the compr_preload.

Algorithm 1 Retargeting algorithm: configure

Require: $\Omega$, $\Psi$
1: $\Gamma := \{\}$ \{Empty data container for counting occurrences\}
2: for all $(i, j)$ s.t. $\omega_i \omega_j \subseteq \Omega \wedge i \leq j \wedge \text{distance}(i, j) \in u$ do
3: \quad increaseOccurrencesCtr($\omega_i, \omega_j$) in $\Gamma$
4: end for
5: for all $\gamma \in \Gamma$ do
6: \quad ($\omega_i, \omega_j$) := $\gamma$
7: \quad \{if $\exists (\omega_k, \omega_m)$ in $\Gamma\{\gamma\}$ \ s.t. $i \leq k \leq j \wedge i \leq m \leq j$ then \}
8: \quad $\delta := (\omega_k, \omega_m)$
9: \quad if calculateGlobalBenefit($\gamma$) $\leq$ calculateGlobalBenefit($\delta$) then
10: \quad \quad decreaseOccurrencesCtr($\gamma$) in $\Gamma$
11: \quad else
12: \quad \quad decreaseOccurrencesCtr($\delta$) in $\Gamma$
13: \quad end if
14: end if
15: end for
16: sortByDecreasingOccurrencesCtr($\Gamma$)
17: for all $\gamma_i \in \Gamma$ \ s.t. \ $i \in [0, 2^\gamma)$ do
18: \quad emplaceInDDU($\gamma_i$)
19: end for

B. Structural retargeting algorithm

Besides this DDU configuration set, a further algorithm is required to identify valuable segments of the original test vector wrt. to the current DDU configuration, executes their replacements and generates a wave trace or even a complete test bench. In general, this means that the original vector $\Omega$ must be retargeted to a suitable vector $\Omega$ as shown in Algorithm 2. $\Omega$ consists of data bits $\omega_1$, ..., $\omega_U$ and $\Omega$ consists of compressed data bits $\overline{\omega_1}$, ..., $\overline{\omega_U}$ to be stored in the test equipment.

The replacements $\Delta$ must be calculated to determine $\Omega$. Every single replacement $\delta \in \Delta$ maps to an underlying CDW as well as the range $(s_i : s_j)$ within the bit vector $\Omega$, i.e. $\omega_i$, $\omega_j$, getting replaced by $\delta$. This set of replacements can be directly processed into the desired $\Omega$, which is transmitted to the TAP controller. Initially, $\Delta$ is calculated as follows: At first, the set $\Delta$ of all possible replacements are determined (line 1). As a following step, valuable start points $InitSet$, which have a benefit $\beta$ greater than zero, are extracted and sorted by $\beta$ decreasingly (line 2 f.). Afterwards, all replacements are inserted into $\Delta$ as long as they are not conflicting with previously inserted ones (line 6 to 12). If the complete coverage of all data bits $\omega_k$ with $k \in [1, U]$ is not achieved yet, the remaining gaps must be filled. For simplifying the shown algorithm, only SBIs are invoked. This means that the uncovered single bit $\omega_i$ is replaced by a suitable replacement $\delta_i$ with $\beta \leq 0$ (line 16 f.). Subsequently, the complete coverage is ensured, the set of replacement is sorted and converted into the final vector containing compressed data bits $\Omega$ (line 22 to 24), which finalizes the retargeting process. In case of using $\mu$-

Algorithm 2 Retargeting algorithm: compress

Require: $\Omega$
1: $\Delta := \{(CDW, s_i, s_j) | \Psi(CDW) = \omega_i \omega_j \subseteq \Omega \wedge i \leq j\}$
2: $InitSet := \{\delta \in \Delta | B(getCDW(\delta)) > 0\}$
3: $InitSet := sortByDecreasingBenefit(InitSet)$
4: $\Delta := \emptyset$
5: \{Processing start points\}
6: for all $init \in InitSet$ do
7: \quad $(CDW, s_i, s_j) := init$
8: \quad if $\exists (i, j) : s_i \text{isCoveredBy}(\Delta)$ then
9: \quad \quad continue \{Skip conflicting replacement\}
10: end if
11: \quad $\Delta := \Delta \cup init$ \{Add replacement\}
12: end for
13: $\Omega := \emptyset$
14: \{Filling existing uncovered areas in bit vec\}
15: for all $i \in [1, U]$ do
16: \quad $\delta_i := (\Psi^{-1}(\omega_i), s_{i'}, s_{i'})$
17: \quad $\Omega := \Omega \cup \delta_i$
18: end for

Ensure: Every data bit is covered.
19: \{Generating sequence of compressed data bit\}
20: $\Delta := sortByIncreasingPost(\Delta)$
21: \{Next step only in case of $\mu$-compr.\}
22: $\Delta := mergeConsecutiveIdenticalCDWs(\Delta)$
23: for all $\delta_i \in \Delta$ do
24: \quad $\Omega := \Omega \cup getCDW(\delta_i)$
25: end for
26: return $\Omega$

compr, the merging step has to be executed just before starting the concatenation (line 21) by substituting specific not-empty replacements with $\emptyset$.

VI. EXPERIMENTAL RESULTS

This section describes the experimental setup and the execution of the experiments itself. One important element of the implementation is a TAP controller, which provides a JTAG interface fully compliant with IEEE 1149.1 standard [11]. For modeling a real use case scenario, a TDR was appended to this design, which provides a serial & 8-bit parallel dual port memory interface connected to the TAP controller. This is done in such a way that all received data is stored into the TDR, when the standardized JTAG instruction bypass is loaded. Additionally, a Functional Logic (FL) module was integrated to this design as well representing an FL block in later designs. A Verilog test bench TB was written to emulate the underlying JTAG protocol, which allows to evaluate this proposed compression technique for TDV reduction. The test bench allows to serve the JTAG protocol and transfers data into an exposed TDR. The test bench TBLeg transfers all test data by using legacy JTAG only. In contrast, TBCompr takes advantage of the proposed compression technique.

Different use cases require heterogeneous test data throughput. Because of this, multiple test data sizes starting with 256 bytes ($N_1$) up to 2048 bytes ($N_4$) were assumed. All these test data ($RTDR_N_1$, ..., $RTDR_N_4$) were generated by a pseudo-random number generator based on the Mersenne Twister algorithm, which determines $\{0, 1\}$ as elements of bit strings. These random bit strings are characterized by a very high entropy, so that the lower bound for compression ratio should be determined [12], which can be achieved by VecTHOR while processing, e.g. precompressed test data.

The results of the random test data runs are shown in Table III. Additionally, further benchmarks were also run for fully-specified test data for various industrial circuits containing 35k to 378k nets, which were provided by NXP Semiconductors as listed in Table III. Here, the average value of all patterns is used. Generally, three different measurements are captured: leg
While using real test data as input data, the compression ratio processor with TDV can be achieved for fully-specified test data on industrial designs. The collected data shows that random test runs require a slightly lower number of overall needed test cycles than the overall size of TDI in bit. Moreover, in most of the cases the overall number of required test cycles is reduced measurable as well. An analysis has shown that this is connected with the drafted retargeting algorithm producing an underlying retargeting process, e.g. by using formal methods, that directly correlates with the limited memory resources to be allocated at the test equipment.

VII. CONCLUSIONS & FUTURE WORK

This paper proposed VecTHOR: A new low-cost compression architecture for IEEE 1149-compliant TAP controllers connected with a suitable framework, which provides a retargeting mechanism for existing test vectors. VecTHOR facilitates to take advantage of TDV reduction by causing only a slight increase of the design size and manageable overhead in the overall number of test cycles.

Several experiments have proven VecTHOR’s noticeable TDV reduction: A compression ratio of almost half of the TDV can be achieved for fully-specified test data on industrial designs. Furthermore, in most of the cases the overall number of required test cycles is reduced measurable as well. An average TDV compression ratio of more than one quarter could be observed even for high entropy test data, which clearly recommends the usage of VecTHOR as a low-cost compression architecture for compliant IEEE 1149 TAP controllers.

Future work focuses on improving the effectiveness of the underlying retargeting process, e.g. by using formal methods, which may lead to a reduced or even lower amount of test cycles compared to the legacy operation mode. As shown by the extended µ-compr technique, it can be expected that an advanced reuse of transmitted CDWs internally increases these benefits even more. Consequently, future work could focus on a higher interlacement of compressed data or extend the capabilities of addressing CDWs, which were not only the direct predecessor but n transmissions in the past. Finally, VecTHOR could be enhanced to be applied on TDO data as well or for being connected with a serial interface of the FL block by embedding a timing-aware parallel-to-serial mechanism.

VIII. ACKNOWLEDGMENT

The work of S. Huhn has been supported by the Graduate School SyDe and the work of S. Eggersglüß by the Institutional Strategy of the University of Bremen, both funded by the German Excellence Initiative.

REFERENCES