Reconfigurable TAP Controllers with Embedded Compression for Large Test Data Volume

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

*University of Bremen, Germany
{huhn,segg,drechsle}@informatik.uni-bremen.de

†Cyber-Physical Systems, DFKI GmbH
28359 Bremen, Germany

***Abstract*—The increasing modularity of state-of-the-art integrated circuit designs leads to new requirements in terms of accessibility during testing and debugging, particularly in post-silicon phases. IEEE 1149.1 *Test Access Port* (TAP) controllers are typically introduced to the design and certain external hardware equipment is incorporated to enable the required access. However, transferring large data through this TAP causes high costs. Thus, an embedded compression architecture is introduced to the TAP to significantly reduce the test application time and the test data volume. Here, the retargeting of the test data is a crucial task.**

This work presents a partition-based formal retargeting technique to take advantage of embedded compression while processing even large and high-entropic test data. The proposed technique tackles the shortcomings of previously proposed retargeting approaches, which require an impractical computational effort for large test data volume or cause an adverse impact on the test application time. For evaluating the proposed method, several different test data sets have been processed to determine suitable parameter sets. As shown by the results, this method allows to compress even huge and high-entropic test data in average by 37.3% and to compress functional verification tests for state-of-the-art industrial designs by up to 62.5%. Furthermore, any adverse impact on the test application time is completely avoided and the procedure always finishes within reasonable run-time.

I. Introduction

The semiconductor industry fabricates *Integrated Circuits* (ICs), which include a steadily increasing number of nested sub-modules. This leads to new challenges in the field of testing, particularly during system-level, board-level or in-field testing as well as for post-silicon debug. For these applications, a dedicated test access mechanism is introduced to the top-level of the *Circuit-under-Test* (CuT) such that the sub-modules can be still accessed in later production phases.

The IEEE 1149.1 Std. specifies such a *Test Access Port* (TAP), which is frequently used in state-of-the-art industrial designs. This TAP allows to transfer test data into the circuit while utilizing specialized external test or debug equipment. In general, such an equipment has strong resource limitations and provides only low-bandwidth transfer, which lead both to restricted testing capabilities.

Several techniques have been proposed over the past years, which introduce additional hardware blocks to the CuT to reduce the *Test Data Volume* (TDV). One prominent candidate is the *Embedded Deterministic Test* (EDT) [1], which achieves a significant TDV reduction for test patterns that were determined by *Automatic Test Pattern Generation* tools. However, the targeted data has to inherit certain properties. Thus, this technique is not arbitrary applicable on any kind of data. Furthermore, the EDT interface has to be accessed and served by the specialized test or debug equipment, which is both not possible in post-silicon phases.

978-1-5386-0362-8/17/$31.00

Other compression techniques, which are well-known from the field of software compression, have been implemented as stand-alone hardware-blocks [2], [3]. For instance, work [4] drafts a technique that provides run-length-encoding capabilities. A static encoding scheme is proposed in work [5], which works well for precomputed test data. Besides this, dictionary-based approaches, e.g., [6], exist, which include a statically programmed dictionary that works well for a priori known test data. Further techniques like the *Lempel-Ziv* (LZ) [7], [8] algorithm or the Golomb-Coding [9] have been also implemented. These techniques are able to compress large data volume easily but introduce a significant hardware overhead. Furthermore, other approaches invoke a statistical encoding scheme, e.g., word-level Huffman encoding [10], multi-level Huffman encoding [11] or even more enhanced schemes [12], [13]. However, none of them can neither incorporate certain hardware constraints nor provide a standardized interface.

An embedded codeword-based compression technique, which is directly integrated into the TAP controller, has been proposed in [14]. This technique offers a powerful mechanism to achieve a significant reduction of the TDV as well as of the *Test Application Time* (TAT) while causing only a slight hardware overhead. The test data has to be processed once by a preceding retargeting procedure to take advantage of this embedded compression. Two different types of retargeting techniques have been proposed in the literature: structural as well as formal techniques. Generally, both techniques achieve a significant TDV reduction. The structural technique proposed in [14] allows a very fast retargeting. However, a measurable overhead concerning the TAT is introduced, particularly, while processing high-entropic test data. In contrast, the formal technique of [15] avoids any increase of TAT even in case of high-entropic data processing but introduces a huge computational effort. Due to a very high run-time, the application of formal techniques on large TDV is limited.

This work proposes a new formal optimization-based technique, which incorporates a parameterizable partitioning scheme to, eventually, determine multiple sets of optimal codewords. The application of this scheme allows to process even large test data by using formal techniques while consuming only reasonable run-time. Consequently, massive advantages are accomplished concerning the TAT reduction compared to the previously proposed structural approach [14].

Experiments are conducted on commercially representative functional verification test data for a state-of-the-art softcore microprocessor as well as on random test data, which is known to be worst compressible due to the high entropy [16].

The structure of this paper is as follows: Section II briefly describes the underlying codeword-based compression architecture for TAP controllers and the existing retargeting techniques. Subsequently, Section III draws the partitioning scheme for the optimization-based procedure and the configuration of the individual sets of codewords. Experimental results are shown in Section IV distinguishing this work against previous approaches and a beneficial parameter set is identified. Finally, Section V summarizes the paper.

II. Background

A TAP is typically integrated into the IC to address the emerging challenges in the field of testing as well as of debugging by combining several important functions, e.g., providing data access and test control features. The IEEE 1149.1 Std. (JTAG) [17] specifies a mechanism which is proven to be well working, allocates only slight hardware resources in sense of area and requires only five additional pins at top-level. This TAP supports a certain set of instructions, which are controlled by a central control unit. The wide dissemination of JTAG has led to the development of a codeword-based compression architecture with run-length encoding capabilities, called *VecTHOR* as presented in [14], which will be briefly described in the following Subsection II-A. Finally, the existing retargeting approaches are briefly introduced in Subsection II-B.

A. Embedded Compression Architecture

VecTHOR can be seamlessly integrated into a standardized IEEE 1149.1-compliant TAP controller and ensures the full legacy support of the underlying JTAG protocol while allocating only slightly more hardware resources. These properties are achieved by extending the underlying control unit of the TAP controller and by enhancing the instruction set: Two additional instructions have been inserted to control the compression scheme, e.g., enabling or disabling the compression mode. Furthermore, a certain role has been assigned to the *Test Mode Select* pin: Controlling specific operations as long as the compression mode is active. The main component of VecTHOR is the *Dynamic Decompressing Unit* (DDU), which holds a dictionary of configurable entries with different codewords.

The overall flow is shown in Figure 1 and is as follows:

1) A retargeting framework is applied on the *Uncompressed Test Data* (UTD), i.e., the original incoming test data, to determine the configuration and generate the corresponding *Compressed Test Data* (CTD).
2) The dictionary is dynamically programmed in the preloading phase before the transfer of the CTD is performed. Thus, it is required to determine a Configuration $\mathcal{C}$ that defines the codewords which, subsequently, will occur in the CTD. In fact, this CTD consists of numerous concatenated codewords (all included in $\mathcal{C}$).
3) Finally, the compression mode is activated and the CTD is transferred to the TAP. Following this scheme means that the DDU implements a function Ψ, which decompresses on-chip the CTD with respect to the programmed $\mathcal{C}$ to restore the identical UTD, i.e., $\Psi(\mathsf{CTD}, \mathcal{C}) = \mathsf{UTD}$ holds. Hereby, Ψ splits the CTD again into single codewords, which become resolved by the dictionary.

For the sake of simplicity, more advanced features have not been taking into account in this description, e.g., the mechanism to ensure that each and every possible incoming data can be represented or the invocation of the run-length encoding capability. We refer to [14], [15] for more information.

B. Retargeting Techniques

The results of the retargeting procedure has a strong influence on the overall effectiveness of the compression technique. In particular, the selection of *beneficial* codewords, which are configured in the preloading phase, is most crucial for the final TDV and TAT reduction.

Structural Approach [14]: This structural approach tries to identify a suitable set of codewords based on their individual number of occurrences, i.e., selecting the codewords which occur most frequently in the UTD. In particular, all possible permutations of all bit sequences (codewords) of supported lengths (possible length of a dictionary entry) are considered as candidates for codewords and their number of occurrences are determined. The most frequent codewords are selected in a greedy-like manner with respect to a cost metric for being contained in the Configuration $\mathcal{C}$. Due to the greedy manner, the determined set of codewords is not optimal. Here, optimal means a set of codewords which is most *beneficial* in sense of TDV and TAT reduction.

Formal Approach [15]: This formal approach tackles the shortcomings of structural approaches by utilizing a variation of the Boolean Satisfiability (SAT) problem. The SAT problem asks the question whether a satisfying solution for a given Boolean function exists. This Boolean function $\Omega : \{0,\ 1\}^n \rightarrow \{0,\ 1\}$ is *satisfiable* if an assignment of all variables exists such that $\Omega = 1$ holds, otherwise it is *unsatisfiable* [18].

An extension is the *Pseudo-Boolean* (PB)-SAT problem that allows to integrate weights into the formula, which are used to evaluate the costs of the determined solutions. Finally, the *Pseudo-Boolean-Optimization* (PBO) problem, which is actually used in the formal retargeting approach, extends the PB-SAT problem by an objective function $\mathcal{F}$. Thus, $\mathcal{F}$ allows to assess the quality of the determined solution, i.e., it is possible to determine the optimal solution with respect to certain (user-defined) optimization criteria evaluated by $\mathcal{F}$.

The given retargeting task is translated into a PBO problem Φ that is built up incrementally by the following considerations:

Executing Basic Retargeting: The Boolean SAT instance is built by processing the UTD, i.e., determining the CTD and the codewords, which are defined by $\mathcal{C}$. It has to be ensured that the equivalence holds when the determined CTD is restored to the UTD on-chip, which is done by applying the decompressing unit (whose dictionary was dynamically configured with the predetermined codewords).

Considering Hardware Constraints: The SAT instance is extended by Pseudo-Boolean elements, i.e., weights which are added to certain literals. This allows to consider the hardware constraints, namely the maximal number of configurable entries meaning the size of the embedded dictionary.

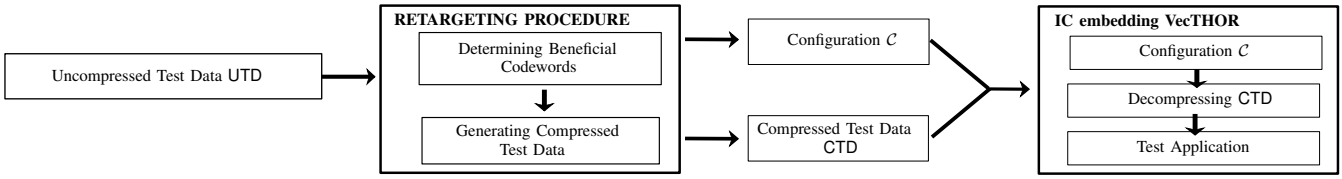

Figure 1: Overall compression flow of VecTHOR [14]

Apply Optimization: Defining a suitable optimization criteria for reducing the result TDV or the TAT, respectively.

Afterwards, a powerful PBO solver [19] is invoked to calculate a fulfilling and most beneficial solution to the given PBO problem. The configuration $\mathcal{C}$ as well as the actual compressed test data can be directly extracted out of the determined solution such that the advantage of the embedded compression architecture can be taken. The disadvantage is that this approach needs excessive run-time for large TDV.

III. Partitioning Scheme

The proposed partitioning and reconfiguration scheme is described in the following section as follows: At first, the basic principle of the proposed scheme is drafted, followed by a detailed application example, the required modification of the formal model and, finally, a discussion about the most suitable parameter set for the scheme.

As shown in [15], the application of formal techniques provides a powerful mechanism to determine the most beneficial CTD as well as $\mathcal{C}$ leading to a reduction of the TDV. This approach works well for small and mid-sized test data volume, though, the maximum size of test data, which can be processed,is strictly limited. This is due to the fact that the PBO instance scales non-linearly with the size of the input test data, the size of the PBO instance explodes by processing large data. Even if well-engineered PBO solvers are available, the run-time as well as the memory consumption, which is required to solve the instance, is completely unsustainable.

A. Basic Principle

The basic idea of this work is to introduce a partitioning scheme to a formal optimization-based retargeting procedure. It is expected that this will significantly reduce the required run-time and memory consumption to determine the CTD as well as $\mathcal{C}$. Primarily, it is targeted to, firstly, avoid a strong adverse impact on the TAT as structural approaches cause and, secondly, consuming only feasible run-time for the determination.

In fact, partitioning is a well-known approach to separate hard computational tasks into multiple sub-tasks that are easier to solve. Generally, this works fine if it is possible to divide the overall computational task into multiple sub-tasks, which can be processed independently. In particular, simply applying such a partitioning on an optimization-based procedure implies that typically only local optima are determined and, consequently, the local optima can strongly deviate from the global optimum.

For transferring this idea to the given retargeting task, the overall test data is split into different chunks. Each chunk is then processed individually (sub-task), which strongly reduces the required computational effort. As already stated above, two critical challenges arise when partitioning is introduced:

1) The independence of the sub-tasks is not given, thus, the data dependencies between these sub-tasks have to be considered by the formal model.
2) A strong deviation exists between the local and the global optimum – in sense of highest TDV and TAT reduction – which will affect the effectiveness of the retargeting procedure. To avoid an adverse impact on the global effectiveness, each and every local optima must be reached which implies that the local as well as the global optima converge.

Local means that only one chunk (sub-task) of the overall test data is processed, hence, a determined configuration works best only for the specific chunk. Analogously, global refers a configuration which is determined by considering all test data at-once and works best globally. However, this calculation requires a very high run-time for large TDV.

To address this deviation, it is targeted to accomplish all local optima by reconfiguring the dictionary for each chunk, however, this causes significant more configuration data. This data overhead is tackled by introducing a partial reconfiguration scheme, which is aware of the current state of the dictionary and reconfigures entries only if beneficial.

B. Introducing Partitioning

In this work, partitioning means to split the overall UTD into single parts –as stated in Definition 1 formally– and to retarget each of these parts individually. To ensure the correctness of the retargeting, several additional aspects have to be considered while selecting and processing those partitions, e.g., the equivalence must hold between UTD and the CTD after decompression on-chip. Thus, the selection process of a single partition must follow the scheme as stated in following two definitions:

Definition 1. *Let UTD be a sequence of bits $(u_1, u_2, .., u_N)$, which represents the data to be partitioned and $\#UTD = N$ the number of bits in the sequence.*

Then, a partition $P_{i,j}$ is defined as a coherent sub-sequence $(u_i, u_{i+1}, u_{i+2}, ..., u_j)$ with $i \leq j$ and $i \geq 1$ and $j \leq N$.

Furthermore, the size of partition $P_{i,j}$ is defined by $\#P_{i,j} = j - i + 1$.

Definition 2. *Let $P_{i,j}$ and $P'_{i',j'}$ be two partitions of the UTD.*

Then, the partitions P and P' are free of intersecting bits, i.e., $j < i'$ or $j' < i$, respectively. That means $P_{i,j} \cap P'_{i',j'} = \varnothing$, i.e., no bit position in UTD is included in more than exactly one partition.

Definition 3. *Let* UTD *be the data to be partitioned and* N *the length of this data.*

Then, psize *is the (maximum) partition size, which means that* $\forall P_{i,j} : \#P_{i,j} \leq$ psize *is valid for all partitions.*

Furthermore, a complete partitioning of UTD *is defined by the ordered set of partitions* $P_1, P_2, ..., P_m$ *with* $\sum_{l=1}^{m} (\#P_l) = N$ *such that* $\forall P_{i,j}, P_{i',j'} : j = i' - 1$*. This means that the ordered sequence of partitions covers all bit positions of* ***UTD*** *in a strict ascending order.*

Even if the partition selection follows this sophisticated scheme, the determined $\mathcal{C}$ is just optimal for a single partition (local optimum). Preliminary experiments have shown that if one set of locally optimal codewords is applied to the complete data stream, the effectiveness of the approach is reduced, which leads to a low reduction of the TDV as well as the TAT compared to the ratios when globally optimal codewords are applied.

Multiple Configurations: To tackle this problem, multiple configurations are determined, one configuration for each processed partition. This new configuration is used to reconfigure the dictionary individually (partition-wise) as stated below. Figure 1 shows that the embedded dictionary is configured by $\mathcal{C}_i$ before the specific compress test data chunk c_i is transferred to the circuit-under-test (holding the TAP controller providing VecTHOR).

As shown in Figure 1, the embedded dictionary is configured by $\mathcal{C}_i$ before the specific compressed test data chunk c_i is transferred to the circuit-under-test (holding the TAP controller with embedded compression). This solves the problem concerning local and global optima well, however, this also introduces large configuration data leading to additional data bits as well as cycles.

Partial Reconfiguration: To avoid an adverse impact on the TDV and, particularly, on the TAT, the reconfiguration of the dictionary is done only partially. Consequently, the retargeting procedure considers the current state $\mathcal{S}$ of the dictionary, i.e, the codewords, which have been configured while processing the previous partition and targets to *reuse* already configured entries for the following partition to, eventually, reduce the overall configuration data as stated in Lemma 1.

Lemma 1. *Given is an ordered sequence of partitions* $(P_1, P_2, ..., P_m)$ *and the current state* S_0 *of the dictionary at point of time* $t = 0$*, i.e., the codewords for all entries within. The point of time* $t = 0$ *represents the initial state of the dictionary (due to the synthesis of the TAP controller).*

Then, the retargeting procedure ρ *receives a partition* P_i *and the current state* S_{i-1}*, which is determined due to previous configurations* $\mathcal{C}_j$ *with* $0 \leq j \leq i - 1$*. Furthermore, a configuration* $\mathcal{C}_i$ *is partial if only a subset of entries is included.*

C. Example

To demonstrate this partial reconfiguration scheme, Table I shows exemplary data for multiple reconfigurations of the embedded dictionary by applying $\mathcal{C}_0$ to $\mathcal{C}_n$. As stated, each $\mathcal{C}_i$ was determined by ρ that processes the partition P_i (representing a chunk of the UTD) with respect to the current state of the dictionary. Column **No.** shows the number of the dictionary entry, column $\mathcal{C}_0$ represents the default state of the dictionary, i.e., the default configuration which is programmed due to synthesis. It is assumed that the dictionary holds 12 dynamically configurable entries, which can contain half-byte or byte-long entries[1]. The columns $\mathcal{C}_1$ up to $\mathcal{C}_n$ represents the state **after** applying the configuration $\mathcal{C}_i$. The last line $\sum \mathcal{C}$ shows the size of the current configuration in bit.

TABLE I: Reconfigurations of codewords C_0 to C_n

No.	$\mathcal{C}_0$	$\mathcal{C}_1$	...	$\mathcal{C}_{n-1}$	$\mathcal{C}_n$
1	1111	01011010		1010	0111
2	0101	1110		00010110	0000
3	0110	0101		0110	11110001
4	00110011	10010110		01111000	00110010
5	01010101	1100		0011	$\mathcal{C}_{n-1}$
6	1010	0101		00011111	01010101
7	0000	10001011		0111	$\mathcal{C}_{n-1}$
8	10101010	11101010		1010	$\mathcal{C}_{n-1}$
9	1000	10001111		$\mathcal{C}_{n-2}$	$\mathcal{C}_{n-2}$
10	1001	10010100		$\mathcal{C}_{n-2}$	1000
11	0001	0100		$\mathcal{C}_{n-2}$	$\mathcal{C}_{n-2}$
12	11001100	11101111		$\mathcal{C}_{n-2}$	$\mathcal{C}_{n-2}$
$\sum \mathcal{C}$ [bit]	$-^A$	79		44	36

^{A} Default configuration due to synthesis.

In case of $\mathcal{C}_{n-1}$ only entries 1 to 8 are included. Thus, entries 9 to 12 remain in the previous state. $\mathcal{C}_n$ also configures the dictionary partially, in particular, the entries 11 and 12 are not reconfigured either. In fact, these both entries remain in the state in which they have been set several configuration phases ago. This example clearly shows that the size of the configuration data directly scales with the number of included entries as well as with the length (half-byte or byte-long) entry - the resulting size varies from 79 bit $(\mathcal{C}_1)$ and 36 bit $(\mathcal{C}_n)$. Thus, it is necessary to consider the resulting size of configuration data while applying the partitioning scheme.

D. Integration of Reconfiguration

In Subsection II-B, an approach is briefly introduced to invoke a PBO-solver to solve the retargeting task. Generally, the PBO instance used to retarget the first partition P_0 can be created in a similar way. To retarget any succeeding partition $P_{i,j}$, all clauses and variable assignments have to be removed from the PBO instance, which refers to the UTD chunk $(u_i, u_{i+1}, ..., u_j)$. Furthermore, the configuration state S of the dictionary has to be extracted and stored after each and every retargeting operation and, consequently, considered within every following one. For this purpose, a function σ was implemented, which includes the current state dictionary S, receives a codeword CW and checks if CW is currently included in the dictionary such that

$$\sigma(CW) = \begin{cases} 1 & \text{if } CW \in S \\ 0 & \text{else} \end{cases}$$

Besides this, the optimization function has to be modified such that the configuration of an already configured codeword CW $(\sigma(CW) = 1)$ does not allocate any additional cost in sense of configuration data.

E. Determining Parameter Set

At least two main parameters have to be adjusted properly, which both strongly influence the size of the configuration data

[1]For a fair comparison, exactly the same parameters (hardware constraints) are assumed as in both [14] and [15].

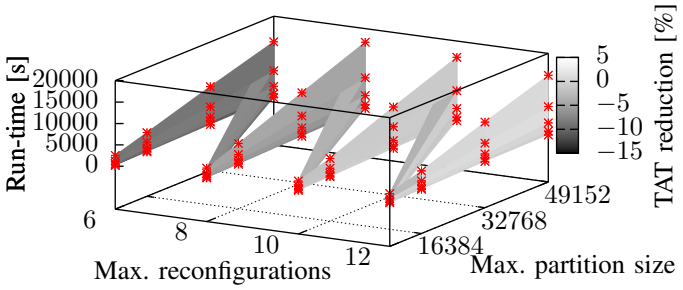

Figure 2: Parameter Identification

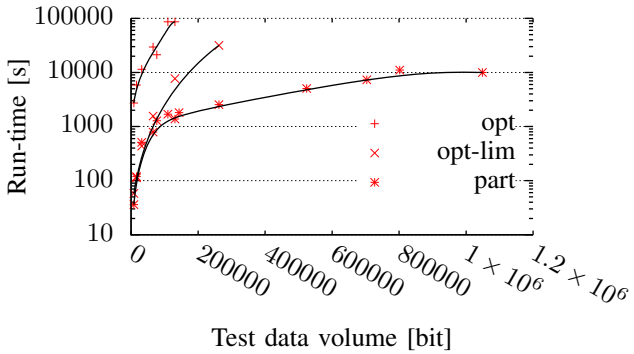

Figure 3: Comparison of run-time of retargeting techniques

as well as the consumed run-time for the retargeting procedure. Consequently, we investigated the effect of the following two parameters for random test data:

Maximal Number of Codewords in Reconfiguration: As already shown in Table I, the overall size of the configuration $\mathcal{C}_i$ directly scales with the number of codewords (included in $\mathcal{C}_i$). Furthermore, this number can not exceed the overall number of dynamically configurable entries. We have investigated reconfiguration ratios between full reconfiguration (100%), half-reconfiguration (50%) and the two intermediate ratios 66.7% and 83.3%. However, these shares represent the maximal number, hence, this does not necessarily mean that this number of entries is actually reconfigured.

Maximal Partition Size: The maximum size of a partition (psize) controls the computational effort, which is required to process this partition. This effort, i.e., the size of the PBO instance scales non-linearly with the size of the input data for the retargeting procedures. Different partition sizes of 8K, 16K, 32K and 48K were investigated.

Figure 2 shows the parameter study for the investigated psizes (8K, 16K, 32K, 48K) at the y-axis and the maximum reconfigurations (6, 8, 10, 12) at the x-axis while processing high-entropic test data with sizes from 16K to 1024K. Each experimental run is represented by a data point and the resulting run-time in sec. is plotted at the z-axis. Besides this, the TDV reduction is stable with an average ratio of 37.3% and a variance of 2.3%. However, the TAT varies between a slight reduction but also between a slight increase (compared to non-compressing data transfer). Hence, the TAT reduction in % is represented by the gray coloring scheme: the lighter the gray, the higher the TAT reduction and vice versa. As shown by Figure 2, determining a suitable parameter set implies a trade-off between a reduced TAT or a reduced run-time. The results show that the parameter set of 12 reconfigurations and a partition size of 32.768 bits represents a fair compromise.

IV. Experimental Results

This section describes the experimental evaluation of the proposed retargeting technique, which introduces a partitioning scheme to a formal optimization-based procedure. Eventually, the results are clearly distinguished against other existing approaches as well as against the standardized JTAG without any compression.

Two different classes of test data were considered for the evaluation: high-entropy, random data with sizes of 8192 bytes (RTDR_8192) and 1048576 bytes (RTDR_1048576) as well as functional verification test data for a state-of-the-art softcore microprocessor. These function data was generated by cross-compiling different test cases of *MiBench* [20]. The setup work of [15] is used to ensure a fair comparison, i.e., two different testbenches have been implemented, both simulating the data transfer to the CuT: Firstly, testbench $\mathrm{TB}_{\mathrm{LEG}}$ implements a reference TAP controller [21] and, secondly, testbench $\mathrm{TB}_{\mathrm{COMPR}}$ embeds a TAP controller using a codeword-based compression technique [14]. Both testbenches are fully compliant with IEEE 1149.1 Std. [17]. The setup assumes that the functional logic block includes a *Test Data Register* (TDR), which operates as a data sink. Different clock-gating schemes have been implemented to prove the practicability.

All experiments were executed on an *Intel Xeon E3-1270v3 3.5 GHz* processor with *32 GB* system memory. The *Time-Out* (TO) was set to 86.400s and the *Memory-Out* (MO) was set to 30GB. The original and uncompressed test data was processed by the developed retargeting framework. This framework is written in C++ and invokes *clasp 3.1.4* as PBO solver [19] and, eventually, generates the compressed test data as well as the configuration for the embedded compression architecture.

The Tables II and III show the TDV as well as the TAT of the conducted experiments. In particular, the following retargeting techniques were conducted: **heur**, the structural technique [14], the optimization-based techniques of work [15]: **opt** as well as **opt-lim**, which alters the parameters of the PBO solver. The column **part** presents the resulting TDV and TAT while using the proposed partitioning scheme with psize $= 16K$ and (up to) a full-reconfiguration including the reconfiguration data and time. Figure 3 shows the run-time comparison (logscale) for different TDVs of both formal techniques [15] versus the proposed part technique.

The TDV of part is stable and comparable to opt-lim as shown in Table II. For instance, part allows to compress 1024K high-entropic test data by 37.3% and functional verification data by up to 62.8%. Furthermore, the experiments generally show that the resulting TDV reduction is lower for high-entropic than for functional verification test data. Concerning the TAT reduction for the most critical high-entropic test data, the resulting TAT reduction of part is slightly lower compared to opt-lim while processing the hard-to-compress high-entropic data. However, the ratios are significant less compared to using heur, though, the ratios are lower compared to opt-lim while applying the proposed technique on functional verification data. In case of

TABLE II: Benchmarks: Processing random test & functional verification data considering TDV

No.	test name	size [bit]					data reduction [%]			
		leg	heur [14]	opt [15]	opt-lim [15]	part	heur [14]	opt [15]	opt-lim [15]	part
1	RTDR_8192	65.536	48.529	42.068	43.057	42.575	26.0	35.8	34.3	35.0
2	RTDR_16384	131.072	89.952	TO	83.476	84.312	31.4	TO	36.3	35.7
3	RTDR_32768	262.144	177.991	TO	166.299	164.700	32.1	TO	36.6	37.2
4	RTDR_65536	524.288	355.682	TO	MO	327.448	32.2	TO	MO	37.5
5	RTDR_131072	1.048.576	711.858	TO	MO	657.303	32.1	TO	MO	37.3
6	patricia	76.864	29.827	31.678	31.781	28.807	61.2	58.7	58.7	62.5
7	bmath	109.793	48.825	TO	50.013	46.545	55.5	TO	54.4	57.6
8	blowfish	143.232	73.579	TO	71.242	67.442	48.6	TO	50.3	52.9
9	cjpeg	703.648	423.315	TO	TO	372.719	39.8	TO	TO	47.0
10	djpeg	801.922	481.904	TO	TO	417.133	39.9	TO	TO	48.0

TABLE III: Benchmarks: Processing random test & functional verification data considering TAT

No.	test name	#data-cycles					TAT reduction [%]			
		leg	heur [14]	opt [15]	opt-lim [15]	part	heur [14]	opt [15]	opt-lim [15]	part
1	RTDR_8192	65.541	71.847	64.420	64.494	65.744	-9.6	1.7	1.5	-0.3
2	RTDR_16384	131.077	151.351	TO	128.427	130.160	-15.5	TO	2.0	0.7
3	RTDR_32768	262.149	300.151	TO	257.168	257.912	-14.5	TO	1.9	1.6
4	RTDR_65536	524.283	600.532	TO	MO	517.048	-14.5	TO	MO	1.4
5	RTDR_131072	1.048.581	1.201.183	TO	MO	1.046.149	-14.6	TO	MO	0.2
6	patricia	76.869	51.622	54.870	54.926	50.705	32.8	28.6	28.5	34.0
7	bmath	109.798	82.875	TO	80.839	85.680	24.5	TO	26.4	22.0
8	blowfish	143.237	126.427	TO	55.426	114.233	11.7	TO	61.3	20.3
9	cjpeg	703.653	715.545	TO	TO	626.735	-1.7	TO	TO	10.9
10	djpeg	801.927	803.661	TO	TO	707.152	-0.2	TO	TO	11.8

processing 64K, heur causes a TAT **increase** of 14.5% while part achieves a TAT reduction of 1.6%.

As clearly shown in Figure 3, the run-time of the proposed approach scales with increasing data volume in a feasible way. For instance, part consumes 119s to process 16K and 9987s to process 1024K, i.e, the scale factor of run-time against volume increase is $\frac{83.9}{64} \approx 1.31$. In comparison to this, opt-lim scales with $\frac{279.9}{16} \approx 17.49$ while processing 16K to 256K before exceeding the TO. opt has already exceeded the TO for 128K.

V. Conclusions

This paper proposed a new formal optimization-based retargeting technique to process even large and high-entropic test data such that the advantages of compression-based TAPs can be leveraged. The proposed retargeting technique incorporates a partition-scheme to reduce the resulting search space of the underlying formal model such that the computation terminates in reasonable time. The basic embedded-dictionary principle of VecTHOR is enhanced by a partial reconfiguration capability to avoid any loss in effectiveness due to the reconfiguration overhead. Experiments have shown that the run-time was significantly reduced compared to other existing formal techniques. In particular, the scaling factor of run-time vs. TDV was reduced by magnitudes from 17.49 to 1.31 while retaining the TDV and the TAT reduction ratios.

VI. Acknowledgment

This work was supported by the University of Bremen's graduate school SyDe, funded by the German Excellence Initiative, by the subproject P01 'Predictive function' of the Collaborative Research Center SFB1232, funded by the German Research Foundation, by the Institutional Strategy of the University of Bremen, funded by the German Excellence Initiative and by the German Research Foundation under contract number EG 290/5-1.

References

[1] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 776–792, 2004.

[2] F. G. Wolff and C. Papachristou, "Multiscan-based test compression and hardware decompression using LZ77," in *Int'l Test Conf.*, 2002, pp. 331–339.

[3] K. U. Irrgang and T. B. Preußer, "An LZ77-style bit-level compression for trace data compaction," in *Field Programmable Logic and Applications*, 2015, pp. 1–4.

[4] A. Jas, J. Ghosh-Dastidar, and N. Touba, "Scan vector compression/decompression using statistical coding," in *VLSI Test Symp.*, 1999, pp. 114–120.

[5] V. Iyengar, K. Chakrabarty, and B. Murray, "Deterministic built-in pattern generation for sequential circuits," *Journal of Electronic Testing*, vol. 15, no. 1-2, pp. 97–114, 1999.

[6] L. Li and K. Chakrabarty, "Test data compression using dictionaries with fixed-length indices," in *VLSI Test Symp.*, 2003, pp. 219–224.

[7] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[8] M. A. A. E. Ghany, A. E. Salama, and A. H. Khalil, "Design and implementation of FPGA-based systolic array for LZ data compression," in *Circuits and Systems*, 2007, pp. 3691–3695.

[9] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 355–368, 2001.

[10] K. Ilambharathi, G. S. N. V. V. Manik, N. Sadagopan, and B. Sivaselvan, "Domain specific hierarchical Huffman encoding," *Cornell University Library*, vol. abs/1307.0920, 2013.

[11] W. R. A. Dias and E. D. Moreno, "Code compression using multi-level dictionary," in *IEEE Latin American Symp. on Circuits and Systems*, 2013, pp. 1–4.

[12] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel huffman coding: An efficient test-data compression method for ip cores," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 26, no. 6, pp. 1070–1083, 2007.

[13] ——, "Test data compression based on variable-to-variable huffman encoding with codeword reusability," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1333–1338, 2008.

[14] S. Huhn, S. Eggersglüß, and R. Drechsler, "VecTHOR: Low-cost compression architecture for IEEE 1149-compliant TAP controllers," in *IEEE European Test Symp.*, 2016, pp. 1–6.

[15] S. Huhn, S. Eggersglüß, K. Chakrabarty, and R. Drechsler, "Optimization of retargeting for IEEE 1149.1 TAP controllers with embedded compression," in *Design, Automation and Test in Europe*, 2017, pp. 578–583.

[16] K. Balakrishnan and N. Touba, "Relationship between entropy and test data compression," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 386–395, 2007.

[17] "IEEE standard for test access port and boundary-scan architecture - redline," *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001) - Redline*, pp. 1–899, 2013.

[18] N. Eén and N. Sörensson, "An extensible SAT solver," in *Int'l Conf. on Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, vol. 2919, 2004, pp. 502–518.

[19] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "Conflict-driven answer set solving," in *Int'l Joint Conf. on AI*, 2007, pp. 386–392.

[20] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *IEEE Int. Workshop on Workload Characterization, 2001.*, Dec 2001, pp. 3–14.

[21] I. Mohor, "JTAG test access port (TAP)," 2009, http://opencores.org/project,jtag.