



IEEE Standard for Describing On-Chip Scan Compression

IEEE Computer Society

Sponsored by the
Test Technology Standards Committee

1450.6.1TM

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
13 July 2009

IEEE Std 1450.6.1TM-2009

IEEE Standard for Describing On-Chip Scan Compression

Sponsor

Test Technology Standards Committee
of the
IEEE Computer Society

Approved 13 May 2009

IEEE-SA Standards Board

Abstract: This standard defines how the necessary information is passed from scan insertion to pattern generation and from pattern generation to diagnosis such that different tool vendors could be used for each step independent of on-chip scan compression logic used.

Keywords: chip design, design automation, on-chip scan compression, pattern generation, scan insertion

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2009 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 13 July 2009. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-0-7381-5962-1 STD95936
Print: ISBN 978-0-7381-5963-8 STDPD95936

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied “**AS IS.**”

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation, or every ten years for stabilization. When a document is more than five years old and has not been reaffirmed, or more than ten years old and has not been stabilized, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon his or her independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Recommendations to change the status of a stabilized standard should include a rationale as to why a revision or withdrawal is required. Comments and recommendations on standards, and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

This introduction is not part of IEEE Std 1450.6.1-2009, IEEE Standard for Describing On-Chip Scan Compression.

The purpose of this standard is to provide a sufficient description of on-chip scan compression structures, operation, and connectivity such that EDA tools may interoperate for pattern generation and diagnosis. The standard extends the CTL language (IEEE Std 1450.6), which provides such information for designs without on-chip scan compression structures.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association website at <http://ieeexplore.ieee.org/xpl/standards.jsp>, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA website at <http://standards.ieee.org>.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/updates/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/interp/index.html>.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the OCI (Open Compression Interface) Working Group had the following membership:

Bruce Cory (NVIDIA), *Chair*

Dwayne Burek (Magma)
Phil Burlison (Verigy)
Al Crouch (Asset Intertech)
Manish Dandekar (Intel)

Geir Eide (Magma)
Rohit Kapur (Synopsys)
Mark Kassab (Mentor Graphics)

Brion Keller (Cadence)
Kee Sup Kim (Intel)
Steve Oakland (IBM)
Rolf Schlagenhaft (Freescale)

The following members of the entity balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Accellera
Cadence Design

Intel
Mentor Graphics

Nvidia
Synopsys

When the IEEE-SA Standards Board approved this standard on 13 May 2009, it had the following membership:

Robert M. Grow, *Chair* **Thomas Prevost, *Vice Chair*** **Steve M. Mills, *Past Chair*** **Judith Gorman, *Secretary***

John Barr
Karen Bartleson
Victor Berman
Ted Burse
Richard DeBlasio
Andy Drozd

Mark Epstein
Alexander Gelman
Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick

David J. Law
Ted Olsen
Glenn Parsons
Ronald C. Petersen
Narayanan Ramachandran
Jon Walter Rosdahl
Sam Sciacca

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Howard L. Wolfman, TAB Representative
Michael Janezic, NIST Representative
Satish Aggarwal, NRC Representative

Lorraine Patsco
IEEE Standards Program Manager, Document Development

Michael Kipness
IEEE Standards Program Manager, Technical Program Development

Contents

1. Overview.....	1
1.1 Scope	1
1.2 General	1
1.3 Conceptual data flow.....	2
1.4 High-level implementation details	3
1.5 Limitations of this standard.....	5
1.6 Structure of this standard.....	5
2. Normative references.....	6
3. Definitions, acronyms, and abbreviations	6
3.1 Definitions	6
3.2 Acronyms and abbreviations	6
4. New blocks—CompressionStructures	7
4.1 CompressionStructures block.....	7
4.2 CompressionStructures block syntax description	7
4.3 CompressionStructures block example	8
5. STIL block—extensions to IEEE Std 1450-1999, Clause 8.....	9
5.1 STIL syntax	9
5.2 STIL syntax description	9
5.3 STIL syntax example	9
6. Environment block—extensions to IEEE Std 1450.1-2005, Clause 17	10
6.1 Environment block	10
6.2 Environment block syntax description	10
6.3 Environment block example.....	12
7. Semantics updates.....	12
Annex A (informative) Examples.....	14

IEEE Standard for Describing On-Chip Scan Compression

IMPORTANT NOTICE: *This standard is not intended to ensure safety, security, health, or environmental protection in all circumstances. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.*

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

1. Overview

1.1 Scope

This standard defines how the necessary information is passed from scan insertion to pattern generation and from pattern generation to diagnosis such that different tool vendors could be used for each step independent of on-chip scan compression logic used.

1.2 General

Flows for scan-based test tools are broken into three main stages. Test logic insertion, pattern generation, and diagnosis. *Test logic insertion* does insertion and verification of test logic. *Pattern generation* uses the test logic to make test patterns that can be used to verify if the design is fabricated correctly. *Diagnosis* is used to identify the failing location in a specific device. Diagnosis information can also be used to increase future yield and to solve problems that keep a design from going to market.

This standard, the Open Compression Interface (OCI), defines on-chip scan compression structures (OCSCS), which can be used to pass information from test logic insertion to pattern generation, and from pattern generation to diagnosis, such that interoperability of electronic design automation (EDA) tools is possible. These structures

- Only reveal as much hardware implementation as necessary for pattern generation and diagnosis
- Do not limit EDA companies from innovating and developing their own compression solutions

- Support most types of structures for input and output compression

OCSCS are defined here as an extension of IEEE Std 1450.6™-2005 [Core Test Language (CTL)].¹ This standard highlights the revised CTL syntax and then provides several examples that cover common compression structure types. The actual OCI syntax and semantics appear prior to the examples.

1.3 Conceptual data flow

Figure 1 shows the OCI conceptual data flow from test logic insertion to pattern generation and from pattern generation to diagnosis. Both the OCI flow (assuming a different vendor for each stage) and the current flow (same vendor for each stage) are shown. Since same-vendor tools have their own way of linking different stages of the flow together, OCI is only needed for changing stages when any previous step was completed by a different vendor. If different vendors were used, then all the data OCI would normally add in the previous stages is needed.

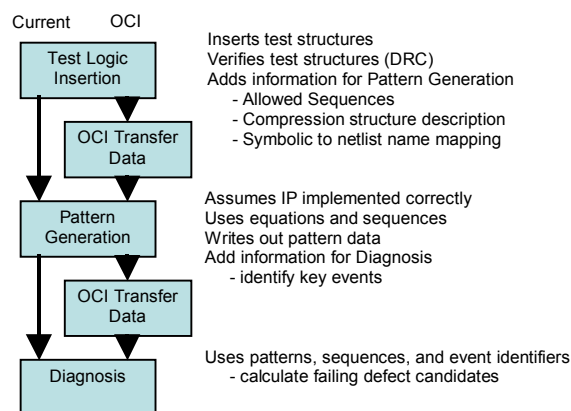


Figure 1—Conceptual OCI flow

The *test logic insertion stage* has the best understanding of the test structure being implemented. Test logic insertion tools are developed based on specific compression structures and have many special design rule checks implemented so customers have as good a chance as possible to generate working, highly effective patterns the first time. The time and knowledge necessary to develop all of the design rule checks is significant. Due to this, the OCI flow presumes the pattern generation tool does not need to perform test structure verification. In the case where a design house is integrating a core with in-house design logic, the test structure in the core is presumed to be verified by the core provider and all core-level OCI information needed for pattern generation shall be in the OCI-compliant CTL file the core provider gives the design house. The design house integrating the core is responsible for generating and verifying the final OCI information passed to pattern generation.

The OCI-specific information passed from test logic insertion to pattern generation includes pattern sequence restrictions, a description of compression hardware, netlist mapping points, and vendor-specific data to ensure vendor flows are as easy to use as possible when using OCI. For a more detailed description of how CTL is leveraged to pass information from test logic insertion to pattern generation, see 1.4.

The *pattern generation stage* can be very time consuming. The automatic test pattern generator (ATPG) determines which logic values are needed on scan cells to detect the most faults by each test pattern. After a pattern is generated, it is then simulated to determine how many faulty locations have been detected. This

¹For information on references, see Clause 2.

sequence is repeated until as many as possible faulty locations have been detected. For chips without on-chip scan compression, the value loaded and unloaded from each scan cell can be mapped directly to a unique value loaded into the chip and to a unique value unloaded from the chip. Only sequence definitions that describe how to initialize, load, unload, and apply tests are needed by pattern generation for this case.

On-chip scan compression needs more information. Putting a logic value in a scan cell requires a value to be loaded into the chip at a previous point in time and likely forces other scan cells to be put to certain logic values. Checking that a value is in a scan cell may require values in masking logic or on other scan cells. Also, on-chip scan compression hardware may restrict the length of a sequence or require a relationship between sequences because of how the hardware is implemented. To do pattern generation efficiently, OCI passes pattern generation sequences, sequence limitations/relationships, and symbolic descriptions of the compression hardware created by the test logic insertion stage.

The output test patterns generated by pattern generation need to be diagnosable using both EDA diagnosis, that can isolate to a gate or net, and on the tester, where isolation is possible due to the failing scan cell or scan chain. To enable this capability, the pattern generation stage shall identify key events in the pattern data and pass this information to diagnosis. The information is passed through OCI by updating the sequences and/or by adding information to the test pattern data. For a more detailed description of how CTL is leveraged to pass information from pattern generation to diagnosis, see 1.4.

The *diagnosis stage* uses the key event information added by pattern generation to map each automatic test equipment (ATE) failure back to a list of defect candidates that might have caused that failure. The diagnosis stage does not add any new information to the OCI file.

1.4 High-level implementation details

OCI data is passed from stage to stage to allow tool interoperability. The format of the OCI data passed leverages CTL (IEEE Std 1450.6-2005) as a foundation with some new syntax and semantics added where needed. Figure 2 shows the information in an OCI file after test logic insertion.

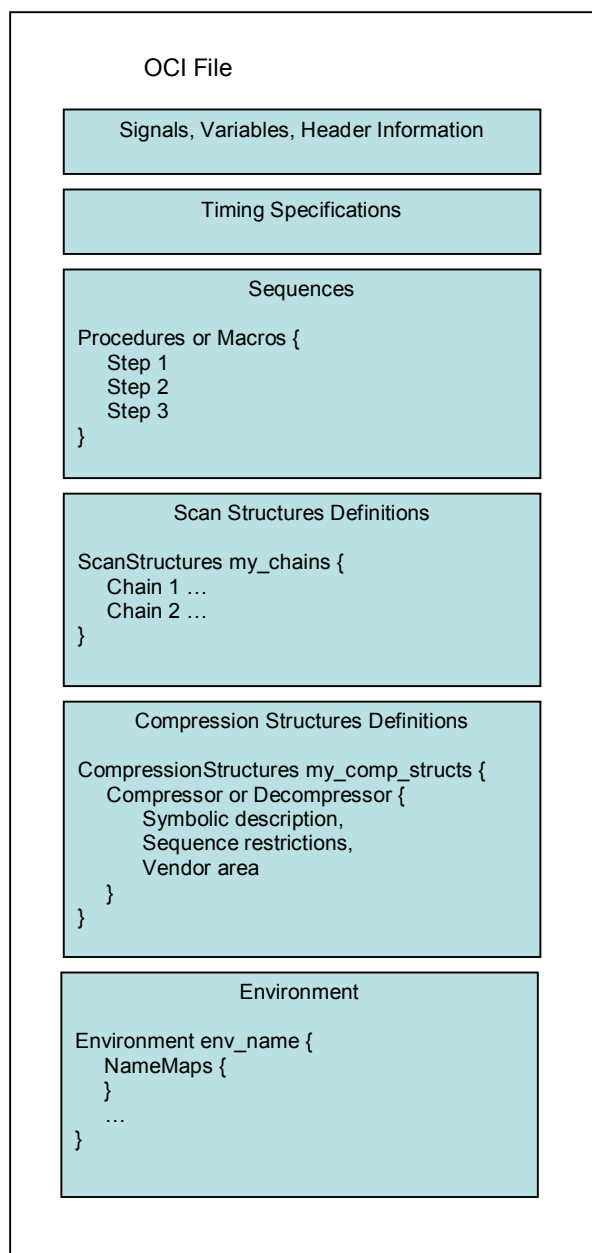


Figure 2—OCI file structure after logic insertion

The goal here is to provide a syntax such that tools support a uniform interface and no special implementation is needed for the various compression technologies. This is achieved by using sequences to handle the complexities of applying the pattern data and compression equations to handle the data relationships between scan cells and the inputs and outputs of the design.

The sequences are procedures and macros provided by the compression technology implementor, while the pattern data provides the data to the sequences. The data relationship of the scan data is provided in equations in the compression structure block. Pattern generation uses the equations to translate the scan data to the scan-in and scan-outs and provide the patterns that invoke the existing macros and procedures with the data passed to them.

The **Environment** syntax (see Clause 6) can be used to define the compression structures for different modes, allowing for a mix and match of multiple vendor solutions as needed.

1.5 Limitations of this standard

The limitations of this standard are as follows:

- a) OCI does not support definition of compression structures if the internal scan chains can dynamically change length from pattern-to-pattern.
- b) OCI does not support interoperability of test logic insertion.

1.6 Structure of this standard

This standard is to be used with IEEE Std 1450™-1999, IEEE Std 1450.1™-2005, IEEE Std 1450.2™-2002, and IEEE Std 1450.6-2005. The conventions established and defined in IEEE Std 1450-1999 are used in this standard and are included verbatim below.

Many clauses in this document add additional constructs to existing clauses in IEEE Std 1450-1999, IEEE Std 1450.1, and IEEE Std 1450.6-2005 and are so identified in the title. Most constructs defined in this standard are limited to the Environment and Test Mode structures that were extended by IEEE Std 1450.6-2005.

The following is a copy of the conventions as defined in IEEE Std 1450-1999 and followed herein.

Different fonts are used as follows:

- SMALL CAP TEXT is used to indicate user data.
- `courier text` is used to indicate code examples.

In the syntax definitions

- a) SMALL CAP TEXT is used to indicate user data.
- b) **bold text** is used to indicate keywords.
- c) *italic text* is used to reference metatypes.
- d) () indicates optional syntax that may be used zero or one time.
- e) ()+ indicates syntax that may be used one or more times
- f) ()* indicates optional syntax that may be used zero or more times.
- g) <> indicates multiple choice arguments or syntax.
- h) Defaults, where appropriate, are underlined in the syntax.

In the syntax explanations, the verb “shall” is used to indicate mandatory requirements. The meaning of a mandatory requirement varies for different readers of the standard:

- To developers of tools that process OCI (readers), “shall” denotes a requirement that the standard imposes. The resulting implementation is required to enforce this requirement and issue an error if the requirement is not met by the input.
- To developers of OCI files (writers), “shall” denotes mandatory characteristics of the language. The resulting output shall conform to these characteristics.
- To the users of OCI, “shall” denotes mandatory characteristics of the language. Users may depend on these characteristics for interpretation of the CTL source.

The language definition clauses contain statements that use the phrase “it is an error”. This phrase indicates improperly defined CTL information.

2. Normative references

OCI extends syntax defined by IEEE Std 1450-1999, IEEE Std 1450.1-2005, and IEEE Std 1450.6-2005. Exceptions, if any, to these standards are defined in this standard. The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 1450-1999, IEEE Standard Test Interface Language (STIL) for Digital Test Vectors.^{2, 3}

IEEE Std 1450.1-2005, IEEE Standard for Extensions to Standard Test Interface Language (STIL) (IEEE Std 1450-1999) for Semiconductor Design Environments.

IEEE Std 1450.2-2002, IEEE Standard for Extensions to Standard Test Interface Language (STIL) for DC Level Specification.

IEEE Std 1450.6-2005, IEEE Standard for Standard Test Interface Language (STIL) for Digital Test Vector Data—Core Test Language (CTL).

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this standard, the following terms and definitions apply. Since OCI is the union of the syntax defined in four documents, the terminology defined in the other documents (IEEE Std 1450-1999, IEEE Std. 1450.1-2005, IEEE Std. 1450.2-2002, and IEEE Std 1450.6-2006) is applicable in this document. IEEE 100, *The IEEE Standard Dictionary of Electrical and Electronics Terms, Seventh Edition*, should be referenced for terms not defined in this standard.

3.1.1 compressor: Test logic that compacts internal scan-chain data (before it gets to output signal pins).

3.1.2 decompressor: Test logic that expands input signal pin data (before it gets to internal scan-chains).

3.2 Acronyms and abbreviations

1450_0 IEEE Std 1450-1999

ATE automatic test equipment

ATPG automatic test pattern generator

CTL Core Test Language (aggregate syntax of IEEE Std 1450.6-2005, IEEE Std 1450.1-2005, IEEE Std 1450.2-2002, and IEEE Std 1450-1999)

DFT design for test (it represents circuit modifications such as scan chains that are created for test)

EDA electronic design automation

LFSR linear feedback shift register

²IEEE publications are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA (<http://standards.ieee.org/>).

³The IEEE standards or products referred to in this standard are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

MISR	multiple input shift register
OCI	Open Compression Interface
OCSCS	on-chip scan compression structures
SoC	System on Chip
STIL	Standard Test Interface Language (combination of IEEE Std 1450-1999, IEEE Std 1450.1-2005, and IEEE Std 1450.2-2002)

4. New blocks—CompressionStructures

This clause defines a new block **CompressionStructures** that is allowed at the top level of a CTL file and within **CoreType** blocks.

4.1 CompressionStructures block

The **CompressionStructures** block is required for chips with on-chip scan compression logic when a different tool vendor is used for test logic insertion, pattern generation, or diagnosis. A CTL file, including a **CompressionStructures** block, standardizes how on-chip scan compression information is passed to enable different vendor tool interoperability.

The **CompressionStructures** block is primarily used to describe the on-chip scan compression compressor and decompressor logic, including how they connect to the internal scan chains and signal pins. A combination of the **CompressionStructures**, **ScanStructures**, **CTLMode**, **MacroDefs**, and **Procedures** blocks shall describe the behavior of the compressor and decompressor logic from the internal scan chains to signal pins. Optionally, the **CompressionStructures** block can also be used to identify vendors and include vendor-specific information.

```

(CompressionStructures (COMPRESSION_STRUCTURE_NAME) {           (1)
  (<Decompressor | Compressor> COMPRESSION_BLOCK_NAME {         (2)
    (Vendor vendor_name <;                                       (3)
      | { (vendor_value1 = vendor_value2 ; )+ }>)
    (SIGNAL_OR_VARIABLE_NAME logic_expression ; )*              (4)
    (Wire WIRE_NAME logic_expression ; )*                        (5)
    (State state_name {Data logic_expression ; Clock logic_expression ;
                      (Reset logic_expression ; ) (Set logic_expression ; )
                      (FuncX logic_expression ; ) } )*          (6)
  })+
})*

(CoreType CORE_TYPE_NAME {
  ....
  (CompressionStructures (COMPRESSION_STRUCTURE_NAME) {         (7)
    compression_structure_definitions
  })*
})*
```

4.2 CompressionStructures block syntax description

(1) **CompressionStructures** COMPRESSION_STRUCTURE_NAME: This statement begins the definition of a block that defines all, or part of, an on-chip scan compression structure. All compression structures shall be defined in this block. The COMPRESSION_STRUCTURE_NAME shall be unique. If this name is not specified, this structure is global across all modes (the default case).

All shift registers or scan chains only used within compression logic can be defined via **ScanStructures** statements (see 20.1 of *IEEE Std 1450-1999*) or as **States** in Decompressor and Compressor blocks.

CompressionStructures specified via **DomainReferences** in a CTL block are interoperable.

(2) **Decompressor | Compressor** **COMPRESSION_BLOCK_NAME**: This statement separates the definition of the input decompressor and output compressor portion of the on-chip scan compression structure. Masking logic can be defined in either block. The **COMPRESSION_BLOCK_NAME** shall be unique in its domain name scoping level (see 6.9 of *IEEE Std 1450-1999* and 14.3 of *IEEE Std 1450.1-2005*). The **Decompressor** and **Compressor** blocks shall not be empty. Compressor block outputs shall not feed into decompressor block inputs during shifting.

(3) **Vendor**: This block specifies the vendor of the on-chip scan compression hardware, where *vendor_name* is a string (see 9.2 of *IEEE Std 1450.6-2005*).

The original vendor may also choose to include a list of expressions in this block (where *vendor_value1* and *vendor_value2* are strings). This list is not guaranteed to be usable by anyone other than the originating vendor.

(4) **SIGNAL_OR_VARIABLE_NAME** *logic_expression*: This statement specifies the hardware relationship between a signal or **Variable** and the logic feeding it (where *logic_expression* is a single quoted expression of **Signals**, **Wires**, **States**, **Variables**, scan cells, and signal groups operating on each other), by using the following set of Verilog operators: MUX (? :), OR (|), XOR (^), AND (&), or INVERT (~).

(5) **Wire** **WIRE_NAME** *logic_expression*: This statement specifies the hardware relationship between a **Wire** and the logic feeding it, by using the following set of Verilog operators: MUX (? :), OR (|), XOR (^), AND (&), or INVERT (~). The **Wire** keyword defines the **WIRE_NAME** signal as a Pseudo whose scope is only valid within the specified **Decompressor** or **Compressor** block.

(6) **State** *state_name*: This statement defines positive edge flip-flops used in the on-chip scan compression hardware. The *state_name* shall be unique within **CompressionStructures** blocks. If a state element is defined in both the **ScanStructures** and **CompressionStructures** blocks, and both are referenced in the same CTL mode **DomainReference** block, the **ScanStructures** cell name shall be the same as the **CompressionStructures** state name. *state_names* inherit all naming capabilities and rules of **ScanCell** names.

Any **Data** for the defined state element specifies generic data input. The **Clock**, **Reset**, **Set**, and **FuncX** for the defined state element are (presumed) active high. When **Set** and **Reset** are both active, any resulting behavior is undefined. When **FuncX** is active, the state element is set to an undefined state.

(7) Added **CompressionStructures** syntax to **CoreType** blocks to enable hierarchical definition of compression logic. All symbolic *logic_expression* and **State** names can be scoped using the :: and : operators, as defined in 5.1 of *IEEE Std 1450.6-2005*.

4.3 CompressionStructures block example

This is a **CompressionStructures** example for a chip with a combinational compressor and decompressor. There are four input signals (*in0* – *in3*), two output signals (*out0* and *out1*), and three internal scan chains whose scan inputs are the internal nets *chip.si1*, *chip.si2*, and *chip.si3* and whose scan outputs are the internal nets *chip.so1*, *chip.so2*, and *chip.so3*. More complex examples are provided in Annex A.

```
CompressionStructures my_compression {
  Decompressor my_decompressor {
```

```

    Vendor Synopsys {
    Decompressor_architecture = mux;
    }
    chip.si1 `in0?in2:in1';
    chip.si2 `in0?in1:in2';
    chip.si3 `in0?in3:in1';
    }
    Compressor my_compressor {
    Vendor Synopsys {
    Compressor_architecture = xor;
    }
    out0 `chip.so1 ^ chip.so2';
    out1 `chip.so2 ^ chip.so3';
    }
}

```

5. STIL block—extensions to IEEE Std 1450-1999, Clause 8

The STIL statement, as defined in IEEE Std 1450-1999, identifies the primary version of IEEE Std 1450-1999 information contained in a STIL file and the presence of one or more standard Extension constructs.

All other constructs and restrictions for Clause 8 of IEEE Std 1450-1999 are in effect here.

5.1 STIL syntax

```

STIL IEEE_1450_0_IDENTIFIER {
    ( EXT_NAME EXT_VERSION; )+
} // end STIL

```

(1)

5.2 STIL syntax description

(1) EXT_NAME: The specific name of the Extension. This syntax of this standard is identified by the name **Compression**. While the Compression concept and mechanisms include IEEE 1450.6, the associated EXT_NAME for IEEE 1450.6 [namely CTL (and Design)] should be used to indicate use of syntax defined by IEEE Std 1450.6-2005. Refer to IEEE Std 1450.1-2005 and IEEE Std 1450.6-2005 for details.

EXT_VERSION: The primary version of an EXT_NAME. This version of the standard is identified by the value **2006**.

5.3 STIL syntax example

```

STIL 1.0 {
    Design 2005;
    CTL 2005;
    Compression 2006;
}

```

6. Environment block—extensions to IEEE Std 1450.1-2005, Clause 17

This section shows the extensions OCI defines for usage within **Environment** blocks.

6.1 Environment block

```

(Environment (ENV_NAME) {
  ...
  (CTLMode (CTLMODE_NAME) {
    (PatternInformation {
      (<Pattern | PatternBurst> (pat_or_burst_name) {
        (Identifiers (PATTERN_OR_BURST_ENUM)* {
          (EventType identifier_event_enum {
            ...
          })*
        })*
      })*
      (<Procedure | Macro> PROCEDURE_OR_MACRO_NAME {
        (Purpose (procedure_or_macro_enum) +;)
        ...
      })*
    })
    (Internal {
      (sig_ref_expr {
        ...
        (DataType (data_type_enum) +) {
          ...
          (DataRateForProtocol <Average | Maximum | Minimum> INTEGER;)*
          (ConstrainedDuringCapture constrain_value;)*
        })*
      })+
    })
    (External {
      (sig_ref_expr {
        ...
        (PadType <PrePad | PostPad | NoPad>)*
        (PadValue default_state;)*
      })+
    })
    (DomainReferences (CORE_INSTANCE_NAME) * {
      (CompressionStructures (COMPRESSION_STRUCT_NAME) +;)*
      ...
    })*
    ...
  })*
})

```

(1)

(2)

(3)

(4)

(5)

(6)

(7)

(8)

6.2 Environment block syntax description

(1) Added new *identifier_event_enum* options (see 16.1 of IEEE Std 1450.6-2005) called **TestReset** and **PreviousTestEnd**. These values shall be specified for logic tested with **CompressionStructures**. The *labels*