

IEEE Standard for Memory Modeling in Core Test Language (CTL)

IEEE Computer Society

Sponsored by the
Test Technology Standards Committee

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

IEEE Std 1450.6.2™-2014

IEEE Standard for Memory Modeling in Core Test Language (CTL)

Sponsor

Test Technology Standards Committee
of the
IEEE Computer Society

Approved 27 March 2014

IEEE-SA Standards Board

Abstract: Reuse of test data and test structures developed for individual cores (designs) when integrated into larger integrated circuits is required for system-on-chip (SoC) tests. This standard defines language constructs sufficient to represent the context of a memory core and of the integration of that memory core into an SoC. This facilitates the development and reuse of test and repair mechanisms for memories. This standard also defines constructs that represent the test structures internal to the memory core for reuse in the creation of the tests for the logic outside the memory core. Semantic rules are defined for the language to facilitate interoperability between different entities (the memory core provider, the system integrator, and the automation tool developer) involved in the creation of an SoC. The capabilities are an extension of IEEE Std 1450.6™-2005. As a result of this extension, CTL's limitations of handling memories are addressed.

Keywords: core test language (CTL), IEEE 1450.6.2™, memory built-in self-test (memory BIST), memory modeling, memory repair, standard test interface language (STIL), system-on-chip (SoC)

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

Copyright © 2014 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 13 June 2014. Printed in the United States of America.

Liberty is a trademark of Synopsys, Inc.
Verilog is a registered trademark of Cadence Design Systems, Inc.

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

PDF: ISBN 978-0-7381-8972-7 STD98570
Print: ISBN 978-0-7381-8973-4 STDPD98570

*IEEE prohibits discrimination, harassment, and bullying.
For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.
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.*

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (“IEEE-SA”) Standards Board. IEEE (“the Institute”) develops its standards through a consensus development process, approved by the American National Standards Institute (“ANSI”), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, 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.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. 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.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own 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.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of 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 of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive 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 comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

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

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document 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

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted 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 these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, 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.

Updating of IEEE Standards documents

Users of IEEE Standards documents 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.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, 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 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-SA Website at <http://ieeexplore.ieee.org/xpl/standards.jsp> or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at <http://standards.ieee.org>.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

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 by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. 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 IEEE standard was completed, the Core Test Language Working Group had the following membership:

Saman Adham, *Chair*
Slimane Boutobza, *Vice Chair*

Albert Au
Tim Ayres
Karen Darbinyan

Prashant Dubey
Geir Eide

Rajat Mehrotra
Vidya Neerkundar
Gary Waggoner

Past members include the following:

Nicco Bhabu
Jaynata Bhadar

Dwayne Burek
Rohit Kapur

Srinivas Mandyam-Kumar
Chandramouli Ramamurti

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

Saman Adham
Juan Carreon
Keith Chow
Adam Cron
Heiko Ehrenberg
Randall Groves
Peter Harrod

Kazumi Hatayama
Werner Hoelzl
Rohit Kapur
Anzou Ken-Ichi
Adam Ley
Benoit Nadeau-Dostie
Jim O'Reilly
Ulrich Pohl

Paul Reuter
Mike Ricchetti
Thomas Starai
Walter Struppler
Srinivasa Vemuru
Oren Yuen
Daidi Zhong

When the IEEE-SA Standards Board approved this standard on 27 March 2014, it had the following membership:

John Kulick, *Chair*
Jon Walter Rosdahl, *Vice Chair*
Richard H. Hulett, *Past Chair*
Konstantinos Karachalios, *Secretary*

Peter Balma
Farooq Bari
Ted Burse
Clint Chaplin
Stephen Dukes
Jean-Philippe Faure
Gary Hoffman

Michael Janezic
Jeffrey Katz
Joseph L. Koepfinger*
David J. Law
Hung Ling
Oleg Logvinov
Ted Olsen
Glenn Parsons

Ron Petersen
Adrian Stephens
Peter Sutherland
Yatin Trivedi
Phil Winston
Don Wright
Yu Yuan

*Member Emeritus

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

Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Julie Alessi
IEEE-SA Content Publishing

Kathryn Bennett
IEEE-SA Technical Community Programs

Introduction

This introduction is not part of IEEE Std 1450.6.2™-2014, IEEE Standard for Memory Modeling in Core Test Language (CTL).

The core test language (CTL) defined in IEEE Std 1450.6™-2005 is a modeling language created to enable design reuse for system-on-chip flows (or SoC flows). The base CTL standard does not address all memory-specific characteristics and structural information required to create memory test and repair patterns. This extension of CTL adds support for modeling memories.

Contents

1. Overview	1
1.1 General	1
1.2 Memory integration flow	2
1.3 Scope	2
1.4 Purpose	2
1.5 Limitations of this standard	3
1.6 Structure of this standard	3
2. Normative references	3
3. Definitions, acronyms, and abbreviations	4
3.1 Definitions	4
3.2 Acronyms and abbreviations	6
4. Extensions to IEEE Std 1450.6™-2005	7
4.1 CTLMode - Internal block—extension of Data Type attributes	7
4.2 Addition to CTLMode Relation block	10
4.3 Port and Target for Purpose statement—extensions to IEEE Std 1450.6™-2005, Clause 16	11
4.4 Compatibility block—extension to IEEE Std 1450.6™-2005, Clause 16	12
4.5 MemoryStructures block—extension to IEEE Std 1450.6™-2005, Clause 6	16
4.6 FileReference block—extension to IEEE Std 1450.6™-2005, Clause 8	17
5. MemoryRepair block	18
5.1 Syntax	18
5.2 MemoryRepair block syntax description	20
5.3 MemoryRepair Examples	22
6. MemoryPhysicalOrganization block	26
6.1 Syntax	26
6.2 MemoryPhysicalOrganization block syntax descriptions	27
6.3 MemoryPhysicalOrganization examples	29
7. Memory CTL Orientation and Capabilities Tutorial	34
7.1 Example of 3-port (2R/1W) memory with repair	34
7.2 Example of Memory with Pipelining and Bit-Line Twisting	45
7.3 Example of Memory with Bank Addressing	49
7.4 Example of Memory with Error Correction Code	54
7.5 Example of memory with embedded comparators	58

IEEE Standard for Memory Modeling in Core Test Language (CTL)

IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

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 General

Memory core test language (memory CTL) is an extension of the core test language (CTL, or IEEE Std 1450.6™-2005) that allows the description of the test-related aspects of memory blocks. CTL is a language created for a system-on-chip flow (or SoC flow), where a design created by one group is reused as a sub-design of a design created by another group. Memory CTL extends this capability to the special requirements of memory blocks embedded in a larger design of logic circuitry. For further information on CTL, consult IEEE Std 1450.6™-2005.¹

Test of memory blocks embedded into an SoC requires access to these memory blocks, or to logic circuitry capable of testing the memory blocks. Memory CTL does not describe the tests, access through the SoC, or the testing circuitry. Rather, it describes the memory blocks themselves so that the SoC designer can provide the test and test mechanisms. After this provision, the main CTL standard may be used to describe the memory blocks in the context of the SoC. memory CTL is intended as extension of CTL for the purpose of facilitating the development of high coverage memory tests.

This standard defines language constructs sufficient to represent the test and repair-relevant aspects of memory blocks, including their capabilities to facilitate testing outside of the blocks.

¹ Information on references can be found in Clause 2.

Memory CTL is a language designed to be the transfer mechanism of test-related design knowledge between a memory provider and a system integrator to allow for interoperability between the producer and the consumer of the information.

As an extension of CTL, memory CTL leverages existing standard representations. The CTL language is the union of the syntax defined in IEEE Std 1450TM-1999, IEEE Std 1450.1TM-2005, IEEE Std 1450.2TM-2002, and IEEE Std 1450.6TM-2005. The reader is assumed to have knowledge of these standards, and their associated content is not repeated or explained in this standard.

Memory CTL has the following characteristics to support memory test:

- a) Memory CTL can describe a wide variety of memory configurations.
- b) Memory CTL can describe various logical-to-physical organizations.
- c) Memory CTL can describe redundancy mechanisms built into the memory block to facilitate repair.
- d) Memory CTL can describe test logic already built into the memory block.

1.2 Memory integration flow

The memory integration flow is the process in which a memory block, which is instantiated inside a larger block of logic circuitry, is isolated from the other logic for the purpose of test while remaining embedded in the logic function for mission-mode purposes. The memory block may contain features that will assist in its own test, or the test of the logic circuitry around it, when it is hooked up and stimulated properly. Or it may contain only those functions essential for its mission-mode purpose. In either case, the memory CTL description contains the information needed to construct a set of high coverage tests for the memory so that connections and/or additional circuitry may be added to the logic design to facilitate this test.

After the tests and additional circuitry are added, the memory test structures and memory test patterns can be described using IEEE Std 1450.6TM-2005. Further SoC integration and pattern porting steps do not require specific descriptions of the memories once they are encapsulated in the SoC block. See IEEE Std 1450.6TM-2005 for details on SoC integration.

1.3 Scope

SoC test requires reuse of test data and test structures developed for individual cores (designs) when integrated into larger integrated circuits. This standard defines language constructs sufficient to represent the context of a memory core and of the integration of that memory core into an SoC. This facilitates the development and reuse of test and repair mechanisms for memories. This standard also defines constructs that represent the test structures internal to the memory core for reuse in the creation of the tests for the logic outside the memory core. Semantic rules are defined for the language to facilitate interoperability between different entities (the memory core provider, the system integrator, and the automation tool developer) involved in the creation of an SoC. The capabilities are an extension of IEEE Std 1450.6TM-2005. As a result of this extension, CTL's limitations of handling memories are addressed.

1.4 Purpose

The purpose of this standard is to develop an extension to the CTL language that provides a sufficient description of a memory core to support the development and reuse of test and repair mechanisms for that memory after integration into SoC environment and enable creation of test patterns for the logic on the SoC external to the memory.

1.5 Limitations of this standard

As the development of the language for complete functionality is a huge task, the first version of this standard is limited in its functionality. Memory CTL, as described in this standard, has limited support for the following areas:

- a) Flash memories
- b) Content-addressable memories
- c) One-time programmable memories, including e-fuses
- d) Scrambling of pages and banks

All of these areas are considered important and should be addressed in future extensions of this language.

1.6 Structure of this standard

This standard is to be used with IEEE Std 1450.6™-2005, IEEE Std 1450™-1999, IEEE Std 1450.1™-2005, and IEEE Std 1450.2™-2002.

Many clauses in this document add additional constructs to existing clauses in IEEE Std 1450.6™-2005, IEEE Std 1450™-1999, IEEE Std 1450.1™-2005, and IEEE Std 1450.2™-2002 and are so identified in the title. Most constructs defined in this standard are limited to the Environment block, which is defined by IEEE Std 1450.1™-2005.

The following is a copy of the conventions as defined in IEEE Std 1450™-1999 and followed by this standard and the others, cited above, with which it is to be used.

Different fonts are used as follows:

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

In the syntax definitions:

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

2. Normative references

Memory CTL encompasses syntax defined in IEEE Std 1450™-1999, IEEE Std 1450.1™-2005, IEEE Std 1450.2™-2002, IEEE Std 1450.6™-2005, and this standard. The language extends syntax defined by IEEE Std 1450™-1999, IEEE Std 1450.1™-2005, IEEE Std 1450.2™-2002, 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 1450TM-1999, IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data.^{2, 3}

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

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

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

IEEE Std 1364TM-2005, IEEE Standard for Verilog® Hardware Description Language.⁴

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. As memory CTL is the union of the syntax defined in five documents, the terminology defined in the other documents (IEEE Std 1450TM-1999, IEEE Std 1450.1TM-2005, IEEE Std 1450.2TM-2002 and IEEE Std 1450.6TM-2005) is applicable in this standard. For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.⁵

bit line: A continuous connection medium (wire) that connects multiple memory cells through the switch (access device) in each cell for read or write. A bit line is normally enabled by decoding the column address of the memory. Bit lines are normally terminated by sense amplifiers. Bit lines are the data transmission mechanism.

bit line pair: A bit line pair is formed from the bit line and the bit line complement for a single memory column.

memory array: A composition of memory cells by a two-dimensional organization.

memory bank: A sub-space inside the memory address space, as shown in the figure below. Each bank is composed of pages and each page can be seen as a sub-array with its own row and column addresses.

memory cell: The basic entity in a memory comprising one or more memory elements and one or more switches (or access devices) to connect the memory element(s) to be read or written.

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

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

⁴ Verilog is a registered trademark of Cadence Design Systems, Inc.

⁵ *IEEE Standards Dictionary Online* subscription is available at:
http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

memory column: A collection of memory cells with a common bit line. A column may consist of a bit line and a bit line complement.

memory core: The part of a memory macro that consists of a memory array, address decoders, sense amplifiers, and read/write logic.

memory element: An electronic circuit that has the ability to retain data (mostly binary but can be ternary or more) for a certain time so that the data can be read reliably within that time. Data retention can be achieved using feedback loops, capacitive elements, magneto-resistive, or any other means.

memory page: A sub-space inside the memory address space, as shown in the figure below. Each page is composed of word lines that share common column multiplexors and sense amplifiers.

memory row: A collection of memory cells with a common word line.

memory word: A collection of bits that are read or written at the same time.

sense amplifier: An electronic circuit that is capable of amplifying small voltage or current variation and translating that into large voltage variation.

word line: A continuous connection medium (wire) that connects multiple memory cells to activate the switch (access device) in each cell for read or write. A word line is normally enabled by decoding the row address of the memory.

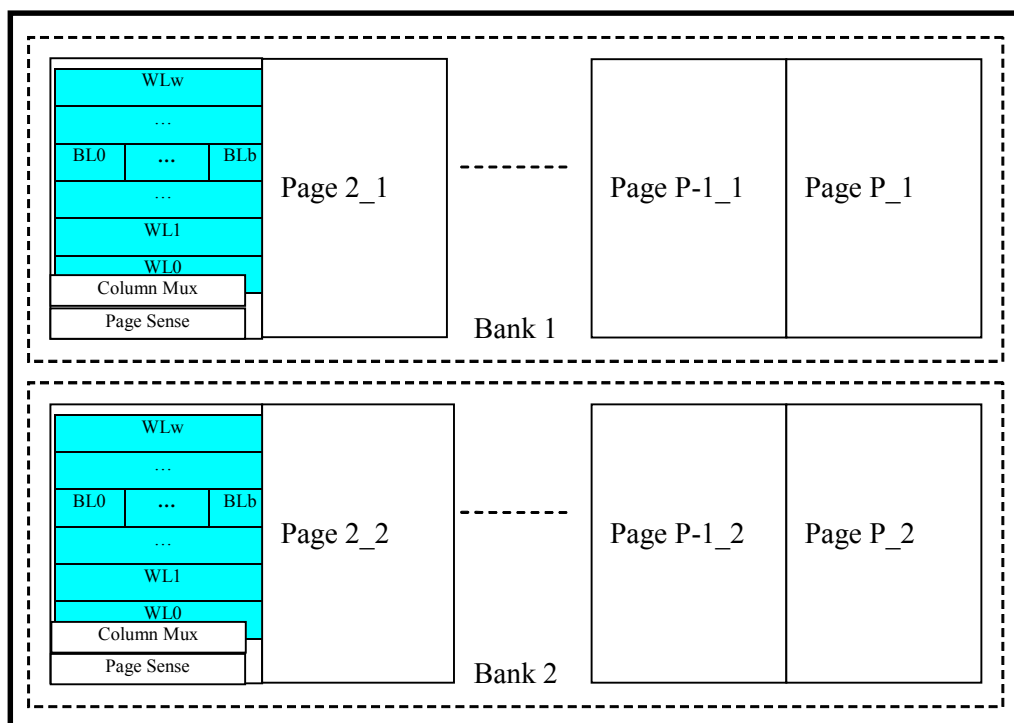


Figure 1—Memory core in presence of pages and banks

3.2 Acronyms and abbreviations

1450	IEEE Std 1450 TM -1999
1450.1	IEEE Std 1450.1 TM -2005
1450.2	IEEE Std 1450.2 TM -2002
1450.6	IEEE Std 1450.6 TM -2005
ATE	automated test equipment
ATPG	automatic test pattern generator
BIST	built-in self-test
CAM	content addressable memory
CTL	Core Test Language (aggregate syntax of 1450.6, 1450.1, 1450.2, and 1450)
DFT	design for test (represents circuit modifications such as scan chains that are created for test)
DRAM	dynamic random access memory
ECC	error correcting code (may also refer to circuitry implementing the code)
IC	integrated circuit
RAM	random access memory
ROM	read-only memory
SoC	system on chip
SRAM	static random access memory
STIL	Standard Test Interface Language (combination of 1450, 1450.1, and 1450.2)
TAM	test access mechanism, used to connect to the (wrapped) cores being tested (refer to 1500)
UDL	user-defined logic (a moniker for any logic that may surround an embedded core)
WFC	WaveFormChar from 1450

4. Extensions to IEEE Std 1450.6™-2005

4.1 CTLMode - Internal block—extension of Data Type attributes

The **CTLMode - Internal** block is described in Clause 10 of IEEE Std 1450.6™-2005. The **DataType** statement defines attributes of all input/output signals of a general core and the connections of the signals to internal parts of the core (scan cells, other cores, other signals). This subclause extends the internal block definition of CTL to include additional attributes to describe all memory pins that may affect memory test to enable seamless integration of test, to provide all information needed to incorporate the test and repair features of the memory core into a larger design, and to provide the necessary test application interface to the memory core such that a set of CTL patterns that are provided with the memory core (and which apply and observe values at the memory core primary input/output signals) can be applied.

The extended **DataType** attributes are added to the *testcontrol_subtype_enum* preceded by the **TestControl** keyword, and to the *testdata_subtype_enum* preceded by the **TestData** keyword.

4.1.1 Syntax

```
memory_testcontrol_subtype_enum =  
  < testcontrol_subtype_enum  
  | AsyncWriteThru  
  | BISTClock  
  | BISTEnable  
  | CAMCompareEnable  
  | CAMReset  
  | ColumnAddressSelect  
  | ECCCorrectionEnable  
  | ECCEncodingEnable  
  | ECCOutEnable  
  | ExpectDataCompareEnable  
  | GroupWriteEnable  
  | MemoryBypass  
  | MemoryClock  
  | MemoryDebugEnabled  
  | MemoryInit  
  | MemoryReset  
  | MemoryScanTest  
  | MemorySelect  
  | PipelineEnable  
  | PortSelect  
  | PowerModeEnable  
  | ReadEnable  
  | RedundancyAccessIndicator  
  | RedundancyUse  
  | Refresh  
  | RefreshClock  
  | RefreshClockSelect  
  | RepairClock  
  | RepairDone  
  | RepairEnable  
  | RepairReset  
  | RowAddressSelect  
  | ScanBypassClock
```

| **ScanClock**
| **SyncWriteThru**
| **TimingMarginSelect** >

memory_testdata_subtype_enum =
 < *testdata_subtype_enum*
 | **BISTSpecific**
 | **CAMComparandInput**
 | **CAMComparandMask**
 | **CAMMatch**
 | **CAMMatchAddress**
 | **CAMMatchOutput**
 | **CAMMultiHit**
 | **DataReadyIndicator**
 | **ECCCorrectedIndicator**
 | **ECCData**
 | **ECCMissIndicator**
 | **ExpectData**
 | **FIFOEmptyIndicator**
 | **FIFOFullIndicator**
 | **MemoryDebugSpecific**
 | **ObservePoint**
 | **OscOut**
 | **RepairData** >

4.1.2 DataType extension syntax descriptions

- a) **AsyncWriteThru:** A signal that enables asynchronous write through functionality of the memory. When active, any data that is written into the memory is available on the data output pins of the memory in the same write cycle.
- b) **BISTClock:** A signal dedicated to clocking the BIST logic.
- c) **BISTEnable:** A signal that enables the BIST mode testing of the memory or activates an embedded memory BIST.
- d) **CAMCompareEnable:** A signal that enables the compare operation in the CAM.
- e) **CAMReset:** A signal used to reset the CAM contents.
- f) **ColumnAddressSelect:** A signal to control the latching of the column address from the address bus. Often abbreviated as CAS.
- g) **ECCCorrectionEnable:** A signal to enable the ECC correction of the memory data output.
- h) **ECCEncodingEnable:** A signal used to enable ECC encoding of memory data input to be stored in the ECC part of the memory array.
- i) **ECCOutEnable:** A signal to control the output of the memory associated with ECC portion.
- j) **ExpectDataCompareEnable:** A signal that enables the comparison of the output data with the expected data.
- k) **GroupWriteEnable:** A signal that controls the write operation to one or more bits in the data path. The range of data bits in the group is defined using the memory specific WriteEnableMap property.
- l) **MemoryBypass:** A signal to enable the memory for scan testing. When activated, the normal memory is bypassed while the scan-related logic is activated.

- m) **MemoryClock:** A signal that clocks memory data into memory elements. In case of a multiport memory or a register file, there can be multiple memory clocks.
- n) **MemoryDebugEnabled:** A signal which is used to activate the memory debug mode.
- o) **MemoryInit:** A signal which initializes the memory state machine, sequential logic, for test, repair, scan, interface, or miscellaneous registers.
- p) **MemoryReset:** A signal used to reset the memory contents.
- q) **MemoryScanTest:** A signal which configures the embedded test logic inside the memory and configures the memory in the test mode to enable scan testing.
- r) **MemorySelect:** A signal to select the memory. When deactivated, memory content is retained, memory periphery is active, but all the dynamic operations such as read or write to a location are disabled.
- s) **PipelineEnable:** A signal used to enable the pipeline at the input or output stage of the memory. When deactivated, the data is sent to the memory output without a pipelining.
- t) **PortSelect:** A signal used to select a port of a multiport memory or register file. When deactivated, the read, write, or decode for the port are disabled.
- u) **PowerModeEnable:** A signal used to configure or decode power modes inside the memory.
- v) **ReadEnable:** A signal used to enable or disable the read operation on a particular port, often used on multiport memories.
- w) **RedundancyAccessIndicator:** A signal to indicate that a redundant location was hit during the memory access.
- x) **RedundancyUse:** A signal to globally enable the repair logic. When disabled, typically during test, no redundancy is used in the memory.
- y) **Refresh:** A signal requesting refresh of the DRAM data. This is a periodic signal and hence the periods need to be specified for a DRAM-related test.
- z) **RefreshClock:** A clock signal used to refresh dynamic memories.
- aa) **RefreshClockSelect:** A static signal used to select between the external RefreshClock or the clock from an internal oscillator.
- bb) **RepairClock:** A signal used to clock the repair data registers.
- cc) **RepairDone:** A signal which becomes active when the repair has been completed inside the memory and memory is ready for use.
- dd) **RepairEnable:** A static signal that indicates whether the memory repair function is enabled or not. This signal is associated with a single repair resource and is overridden when a RedundancyUse signal is disabled.
- ee) **RepairReset:** A signal used to reset the repair data registers.
- ff) **RowAddressSelect:** A signal to control the latching of the row address from the address bus. Often abbreviated as RAS.
- gg) **ScanBypassClock:** A signal that clocks the sequential elements in the bypass logic of the memory.
- hh) **ScanClock:** A signal used to clock functional registers inside a memory for scan shifting or scan test.
- ii) **SyncWriteThru:** A signal that enables synchronous write through functionality of the memory. When active, any data that is written into the memory is available on the data output pins of the memory after a latency of one clock cycle.
- jj) **TimingMarginSelect:** A signal used to select the timing margin when testing the memory.

- kk) **BISTSpecific**: An input signal used for memory data, address, or control during BIST operation. This signal should be given an additional `DataType` to match the signal that it overrides during BIST operation.
- ll) **CAMComparandInput**: CAM input data to be compared.
- mm) **CAMComparandMask**: Mask vector under which the compare operation is done. A specific value on the Mask vector bit causes the compare operation to ignore the related CAM compare input data.
- nn) **CAMMatch**: Single bit global match indicator in a CAM.
- oo) **CAMMatchAddress**: CAM match address vector output.
- pp) **CAMMatchOutput**: CAM “match” vector output. The vector has a specific value on bit positions where a match occurred between the CAM input and data stored in the CAM.
- qq) **CAMMultiHit**: Multiple bit global match indicator in a CAM.
- rr) **DataReadyIndicator**: A signal which indicates that memory data is ready.
- ss) **ECCCorrectedIndicator**: A signal indicating that an error was found and ECC corrected it.
- tt) **ECCData**: A signal that is part of the error correcting code functionality.
- uu) **ECCMissIndicator**: A signal indicating that an uncorrectable error occurred.
- vv) **ExpectData**: Input signal that provides the expected data pattern to be compared with the output data.
- ww) **FIFOEmptyIndicator**: A signal that indicates whether a FIFO-type memory is empty.
- xx) **FIFOFullIndicator**: A signal that indicates whether a FIFO-type memory is full.
- yy) **MemoryDebugSpecific**: A signal used only in debug mode, not in functional operation.
- zz) **ObservePoint**: A signal used to observe a test point inside the memory.
- aaa) **OscOut**: A signal used to observe a memory’s internal self-timed circuit.
- bbb) **RepairData**: A signal that carries the repair data for a specific memory repair resource.

4.2 Addition to CTLMode Relation block

The **CTLMode - Relation** block is described in Clause 13 of IEEE Std 1450.6™-2005. Two additional statements are added to this block. One allows multiple write enable bits to be associated with the memory data inputs whose writing they control. The other allows bits of a BIST specific data or control bus to be distributed across a wider bus of a functional signal.

4.2.1 Syntax

```
Environment { CTLMode (CTLMODE_NAME) {  
    Relation {  
        ( WriteEnableMap signal sigref_expr; )*  
        ( BISTSpecificMap signal sigref_expr; )*  
    } // end Relation  
}} // end Environment, CTLMode
```

4.2.2 Relation block addition syntax description

WriteEnableMap: In the presence of multi-bit write enables, this keyword defines the mapping between the write enable bits and the data bits controlled by those write enables. This keyword can appear only within the Relation block to define the relationship or mapping between the memory write enable signals and the memory data signals. The first *signal* refers to one of the write enable bits, which must be of type GroupWriteEnable or MemoryWrite, and the second *sigref_expr* refers to the memory data input bit or bits whose write it controls. Multiple WriteEnableMap statements are required whenever GroupWriteEnable signals are defined. When there are no WriteEnableMap statements, the MemoryWrite signal is considered to be global across the data word.

A special case applies when the width of the write enable bus is equal to the width of the data bus. In this case, one WriteEnableMap statement can be used to associate the buses, which is considered to be equivalent to associating the bits one by one. When the write enable bus is specified as a whole, bus notation is used.

BISTSpecificMap: This keyword defines the mapping between the BIST specific signal bits and the data or control bits that they override during BIST operation. The first *signal* refers to a signal of type BISTSpecific, and the second *sigref_expr* refers to the memory data or control input bit or bits which it overrides. The BISTSpecific signal should be given an additional DataType to match the signals that it overrides.

4.3 Port and Target for Purpose statement—extensions to IEEE Std 1450.6™-2005, Clause 16

The **Purpose** statement may be applied to **Procedure** blocks inside the **PatternInformation** block. The memory operation purposes are **MemoryPrecharge**, **MemoryRead**, **MemoryReadModifyWrite**, **MemoryRefresh**, **MemoryWrite**, **MemoryBypassMode**, **MemoryBitMasking**, **MemoryRepair**, **MemoryScanTest**, **MemoryStandby**, and **MemoryWriteRead**. The **User** USER_DEFINED purpose may also be used for memory purposes, for example **User MemoryBypass**.

The memory operation purposes may take a sub-statement to indicate the **Port** number used. This **Port** number corresponds to the enumerated **Port** number in the **Relation** block.

The memory operation purposes may take a sub-statement to indicate the **Target** of the current test.

```
memory_enum =  
  < MemoryPrecharge  
  | MemoryRead  
  | MemoryReadModifyWrite  
  | MemoryRefresh  
  | MemoryWrite  
  | MemoryBypassMode  
  | MemoryBitMasking  
  | MemoryRepair  
  | MemoryScanTest  
  | MemoryStandby  
  | MemoryWriteRead  
  | User USER_DEFINED >  
memory_target_enum =  
  < MemoryTest  
  | LogicTest  
  | Mission  
  | User USER_DEFINED >
```

```
Environment { CTLMode (CTLMODE_NAME) { PatternInformation { Procedure {  
    (Purpose memory_enum ({ (Port INTEGER ;) (Target (memory_target_enum)*;) })))  
}}}} // end Environment, CTLMode, PatternInformation, Procedure
```

4.3.1 Memory operation purpose syntax descriptions

The following memory operations are defined in IEEE Std 1450.6™-2005: **MemoryPrecharge**, **MemoryRead**, **MemoryReadModifyWrite**, **MemoryRefresh**, and **MemoryWrite**. The new operations and the targets are defined here.

MemoryBypassMode: A sequence that allows for values to propagate directly from input to output of a memory without being written into the memory.

MemoryBitMasking: A sequence that allows for values to be written into a subset of the word width of a memory.

MemoryRepair: A sequence that allows for values to be applied to the repair control circuitry of a memory.

MemoryScanTest: A sequence that allows for values to be read from or written into the scan chain of a memory.

MemoryStandby: A sequence that allows for a memory to enter a low-power mode.

MemoryWriteRead: A sequence that allows for values to be written into and then read from a memory.

MemoryTest: The operation is part of a sequence that targets the memory core.

LogicTest: The operation is part of a sequence that targets logic outside of the memory core.

Mission: The operation is part of a sequence that targets functional operation of the circuit.

4.4 Compatibility block—extension to IEEE Std 1450.6™-2005, Clause 16

The **Compatibility** block is a new top-level block of the **PatternInformation** block already defined in IEEE Std 1450.6™-2005, Clause 16. It refers to procedures or macros defined in **Procedures** or **MacroDefs** blocks defined elsewhere in the model. The **Compatibility** block does not modify these procedures or macros, but it clarifies the details of the results when they are used for memory testing.

Its purpose is to describe the effects of executing multiple procedures or macros on the same design component, or the effects of executing a procedure or macro with some of the inputs at unknown states. The **Compatibility** block is only required to express limitations on the choices of procedures and macros, since by default all operations are assumed to have the results specified in the main block definitions.

There may be as many **Compatibility** blocks as required to define the different situations and their effects.

4.4.1 Syntax

```
compatibility_enum =  
    < Simultaneous
```

```

| CycleOffset INTEGER
| Order
| Self >
address_difference_enum =
< DiffMoreThan INTEGER
| DiffLessThan INTEGER
| ColumnDiffMoreThan INTEGER
| ColumnDiffLessThan INTEGER
| RowDiffMoreThan INTEGER
| RowDiffLessThan INTEGER >
result_location_enum =
< Memory
| Port INTEGER >
result_enum =
< OldData
| NewData
| XMemory
| XWord
| XDifferences >
severity_enum =
< Allowed
| Warning
| Error >

Environment { CTLMode (CTLMODE_NAME) { PatternInformation {
  (Compatibility compatibility_enum {
    (Macro MACRO_NAME ( { Address address_difference_enum ; } ) ;)*
    (Procedure PROC_NAME ( { Address address_difference_enum ; } ) ;)*
    (AllSignalsUnknown SIGNAL ( SIGNAL)* ;)
    (AnySignalsUnknown SIGNAL ( SIGNAL)* ;)
    (Result result_location_enum result_enum ;)*
    (Severity severity_enum ;)
  })* // end Compatibility
}} // end Environment, CTLMode, PatternInformation

```

4.4.2 Compatibility block syntax descriptions

Compatibility *compatibility_enum*: This statement begins the block of statements that contains the compatibility information for macros and procedures in the current configuration of the design. The *compatibility_enum* refers to the offset in number of Vector (**V**) statements relative to the first-defined macro or procedure. **Simultaneous** means that subsequent macros or procedures start with no time offset. **CycleOffset** INTEGER means that subsequent macros or procedures start after exactly INTEGER **Vector** statements. **Order** means that the first macro or procedure completes before subsequent ones are applied. **Self** means that only one macro or procedure is involved and its X-handling behavior is being described.

Macro MACRO_NAME | **Procedure** PROC_NAME: These statements refer to macros or procedures of the current mode. The MACRO_NAME or PROC_NAME of this statement is expected to be a valid construct in the current CTL. At least one **Macro** or **Procedure** statement is required, and the first one in the block must not have an **Address** sub-statement. Subsequent statements refer to the first one.

Address *address_difference_enum*: This sub-statement qualifies the statement that it is appended to so that the **Result** or **Error** only applies if this condition is met. The INTEGER is the difference between the address used for the current **Macro** or **Procedure** compared to that of the first one listed in the current **Compatibility** block and applies to both positive and negative differences in address. If **Row** or **Column** is specified, then the row or column address is used, otherwise the logical address is used.