

IEEE Standard for Extensions to Standard Test Interface Language (STIL) (IEEE Std 1450™-1999) for Test Flow Specification

IEEE Computer Society

Sponsored by the
Test Technology Standards Committee

IEEE Standard for Extensions to Standard Test Interface Language (STIL) (IEEE Std 1450-1999) for Test Flow Specification

Sponsor

**Test Technology Standards Committee
of the
IEEE Computer Society**

Approved 6 December 2017

IEEE-SA Standards Board

Abstract: IEEE Std 1450™-1999, which specifies the Standard Test Interface Language (STIL), is extended by this standard to provide an interface between test generation tools and test equipment with regard to the specification of the flow of execution of test program components. It defines structures so that test flows, sub-flows, and binning may be described in a manner that facilitates automated generation, modification, and/or manual maintenance and, although not yet a complete run-time test language, execution on automated test equipment (ATE). It also defines an interface between tester configurations (described by IEEE Std 1450-1999 and IEEE Std 1450.2™-2002) and test program components. It also defines a hierarchy of flows, sub-flows, and test components as well as structures for defining flow-related variables and processing expressions involving those variables. It provides structures that support automatic test program generation (ATPRG) and translation and that support running it natively as an ATE programming language. As an adjunct, IEEE Std 1450.3™-2007 may be used by ATPRG for tester rule checking.

Keywords: ATPG, ATPRG, automatic test program generator or generation, binning, C-AE, computer-aided engineering, device under test, DUT, IC test, IEEE 1450.4™, integrated circuit test, test flow, test program description, test program language, TPG

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

Copyright © 2018 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 9 February 2018. 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 978-1-5044-4643-3 STD22970
Print: ISBN 978-1-5044-4644-0 STDPD22970

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 Notices and Disclaimers Concerning IEEE Standards Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

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. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups 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 Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementors and users 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.

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. Implementors 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 Xplore at <http://ieeexplore.ieee.org/> 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 arrangements 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 STIL Test Flow Working Group had the following membership:

Jim O'Reilly, Chair
Ernst J. Wahl, Vice Chair

Gerald Chan*
Kevin Coggins
Julia DiChiaro
Ric Dokken*
Carol Dowding
Dave Dowding
Oleg Erlich
Daniel Fan
Jim Felte
J. Scott Franzen*
Mitsuhito Fujii
Carey Garrenton*
Brian Johnson

Alan Jones
Rohit Kapur
Ajay Khoche
Josie Lewis
Yuhai Ma
Gregory Maston*
Tom Micek
Jim Mosley
Gary Murray
Chris Nelson
Eric Nguyen
Yasunori Okamoto

Don Organ
Bruce Parnas
Paul Reuter
Bob Roberts
Oscar Rodriguez
Jose Santiago
Markus Seuring
Douglas Sprague
Spass Stojanowski
Tony Taylor
S. B. Tatum
Steve Tilson
Allen Teates

(* indicates active membership at the time of draft submission)

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

Paul Berndt
Bill Brown
Juan Carreon
Gerald Chan
Keith Chow
John Cosley
Alfred Crouch
Ric Dokken
David Dowding
Heiko Ehrenberg

Oleg Erlich
J. Scott Franzen
William Fritzsche
Randall Groves
Jon Hagan
Peter Hertz
Wolfgang Hoezl
Nobuhiko Ikeuchi
Gregory Maston
Stephen McGinty
Jeffrey Moore

Charles Ngethe
Jim O'Reilly
Paul Reuter
Osman Sakr
Douglas Sprague
Walter Struppeler
Ernst J. Wahl
Yoshihiro Watanabe
Gregg Wilder
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 6 December 2017, it had the following membership:

Jean-Philippe Faure, John D. Kulick, Chair
Gary Hoffman, Vice Chair
John D. Kulick, Past Chair
Konstantinos Karachalios, Secretary

Thomas Adams
Masayuki Ariyoshi
Ted Atsc
Stephen Dukes
Doug Edwards
J. Travis Griffith
Michael Janezic

Thomas Kochy
Joseph L. Koepfinger*
Kevin Lu
Daleep Mohla
Damir Novosel
Ronald C. Petersen
Annette D. Reilly

Robby Robson
Dorothy Stanley
Adrian Stephens
Mehmet Ulema
Phil Wennblom
Howard Wolfman
Yu Yuan

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1450.4-2017, IEEE Standard for Extensions to Standard Test Interface Language (STIL) (IEEE Std 1450-1999) for Test Flow Specification.

This document is part of a set of IEEE 1450 standards, which cover the Standard Test Interface Language (STIL).

More specifically, this standard (STIL.4) extends IEEE Std 1450TM-1999 (STIL.0) to provide an interface between test generation tools and test equipment with regard to the specification of the flow of execution of test program components. It defines

- Structures so that test flows, sub-flows, and binning may be described in a manner that facilitates automated generation, modification, and/or manual maintenance and, although not yet a complete run-time test language, execution on automated test equipment (ATE).
- An interface between tester configurations [described by STIL.0 and IEEE Std 1450.2TM-2002 (STIL.2)] and test program components.
- A hierarchy of flows, sub-flows, and test components.
- Structures for defining flow-related variables and processing expressions involving those variables.
- Structures that support automatic test program generation (ATPRG) and translation and that support running it natively as an ATE programming language. As an adjunct, IEEE Std 1450.3TM-2007 (STIL.3) may be used by ATPRG for tester rules checking.

Contents

1. Overview	1
1.1 General	1
1.2 Scope	3
1.3 Purpose	3
2. Normative references.....	3
3. Definitions, abbreviations, and acronyms	4
3.1 Definitions	4
3.2 Acronyms and abbreviations	6
4. Preface	7
4.1 General	7
4.2 Word usage.....	7
4.3 Conventions.....	7
4.4 Semantics.....	8
5. Tutorial	8
5.1 General	8
5.2 Flow test program example	8
5.3 FlowExtended test program example	11
6. Extensions to STIL.0 Clause 6 (STIL syntax description)	13
6.1 General	13
6.2 Additional reserved words.....	13
6.3 Additions to STIL.0 Table 3 (SI units).....	15
6.4 Extensions to STIL.0 6.6 (token length).....	15
6.5 Extensions to STIL.0 6.8 (user-defined name characteristics)	16
6.6 Extensions to STIL.0 6.12 (number characteristics).....	16
6.7 Extensions to STIL.0 6.16 (STIL name spaces and name resolution)	16
6.8 Expressions.....	17
6.9 Functions	22
6.10 Enum.....	24
6.11 Parameter, MethodParameter, and FlowVariable types.....	25
7. Extensions to STIL.0 Clause 8 (STIL statement).....	26
7.1 General	26
7.2 STIL syntax	28
7.3 STIL example	28
8. Extensions to STIL.0 Clause 14 (Signals block) (FlowExtended)	28
8.1 General	28
8.2 Signals block syntax and examples	28
9. Extensions to STIL.0 Clause 15 (SignalGroups block) (FlowExtended)	39
10. Extensions to STIL.0 Clause 16 (PatternExec block) (FlowExtended).....	40
10.1 General	40
10.2 PatternExec block syntax.....	40
11. Extensions to STIL.0 Clause 17 (PatternBurst block) (FlowExtended)	40
11.1 General	40

11.2 Extensions to STIL.0 17.1 (PatternBurst block syntax).....	40
12. Extensions to STIL.0 Clause 18 (Timing and WaveformTable block) (FlowExtended).....	41
12.1 General	41
12.2 Timing and WaveformTable syntax	42
13. Extensions to STIL.0 Clause 19 (Spec and Selector blocks).....	42
13.1 General	42
13.2 Spec block syntax	43
14. Extensions to STIL.2 Clause 10 (DCLevels block) (FlowExtended).....	44
14.1 General	44
14.2 DCLevels block syntax.....	44
15. Extensions to STIL.2 Clause 12 (DCSequence) (FlowExtended)	45
15.1 General	45
15.2 DCSequence block syntax	45
15.3 DCSequence block example	46
16. Include enhancements	47
16.1 IncludeOnce.....	47
16.2 DomainInclude	47
17. FlowVariables	49
17.1 General	49
17.2 FlowVariables syntax	49
17.3 FlowVariables examples.....	51
17.4 FlowVariable access	53
17.5 FlowVariable types.....	54
17.6 FlowVariable attributes	60
17.7 FlowVariable operators and member functions.....	63
17.8 FlowVariable array operations.....	65
18. Device to tester interface	66
19. SignalMap	67
19.1 General	67
19.2 SignalMap syntax	68
19.3 SignalMap examples	70
20. Device (FlowExtended)	75
20.1 General	75
20.2 STIL.2 DC levels	82
20.3 Chip	82
20.4 Package	83
20.5 Channel map	84
20.6 Multi-site/MPW testing	86
20.7 Device block examples	86
21. Binning	94
21.1 General	94
21.2 Binning element reference	94
22. SoftBinDefs	95
22.1 SoftBinDefs syntax.....	95

22.2 SoftBinDefs examples	96
22.3 Bins.....	97
22.4 Bin None (FlowExtended).....	99
22.5 Bin axes	100
22.6 countSince functions (FlowExtended).....	100
23. HardBinDefs.....	101
23.1 HardBinDefs syntax.....	101
23.2 HardBinDefs examples.....	102
23.3 Bins.....	102
24. BinMap.....	103
24.1 General	103
24.2 BinMap syntax.....	103
24.3 BinMap example.....	104
25. Flow conceptual model.....	104
26. Flow conceptual model (FlowExtended).....	107
26.1 General	107
26.2 Flow-related types	110
26.3 Inheritance	110
26.4 Instantiation and execution	112
27. TestBase definition (FlowExtended).....	112
27.1 TestBase syntax	112
27.2 TestBase example	116
27.3 Parameter initialization and assignment	117
27.4 Parameter types.....	118
27.5 Parameter attributes	123
27.6 Parameter operators and member functions.....	123
27.7 Parameter array operations	124
27.8 Spec variable access	124
28. TestType definition (FlowExtended).....	126
28.1 General	126
28.2 TestType syntax.....	126
28.3 TestType example.....	128
29. Test.....	129
29.1 General	129
29.2 Test syntax.....	129
29.3 Test example.....	131
30. FlowNode	133
30.1 General	133
30.2 FlowNode syntax	134
30.3 FlowNode examples	136
31. FlowType definition (FlowExtended)	137
31.1 FlowType syntax	137
32. Flow.....	137
32.1 General	137
32.2 Flow syntax.....	138

32.3 Flow examples.....	139
33. Actions and flow control	140
34. TestProgram	142
34.1 General	142
34.2 TestProgram syntax	143
34.3 TestProgram examples.....	144
34.4 Entry points.....	145
34.5 Bin map	145
35. Standard definitions.....	146
35.1 Standard enumerated types.....	146
35.2 Standard global variables (FlowExtended).....	148
35.3 Flow control defaults (FlowExtended)	149
35.4 Standard No-op and None (FlowExtended).....	154
35.5 Standard PatternExec test (FlowExtended)	154
35.6 Standard functional test (FlowExtended).....	155
35.7 Standard flow (FlowExtended).....	156
Annex A (informative) Event sequence	158
A.1 General.....	158
A.2 Parsing and loading.....	158
A.3 Execution	158
Annex B (informative) Top-level block sequence (FlowExtended).....	159
B.1 General.....	159
B.2 Skeleton and dependencies	159
Annex C (informative) Usage examples (FlowExtended).....	161
C.1 Coding examples.....	161
Annex D (informative) Switching from Flow to FlowExtended.....	172

List of Figures

Figure 1—Diagram: STIL flow contents and application	1
Figure 2—Diagram: ATPRG STIL data flow	2
Figure 3—Example: conventions	8
Figure 4—Diagram: STIL flow.....	9
Figure 5—Example: STIL.4 syntax overview	10
Figure 6—Example: "./Patterns/Pat1.stil" include file	11
Figure 7—Example: FlowExtended test program	13
Figure 8—Flow expression assignment and evaluation	18
Figure 9—FlowExtended expression assignment and evaluation	18
Figure 10—Example: SignalGroup functions	22
Figure 11—Example: mathematical functions	23
Figure 12—Example: context-sensitive function <i>executed</i>	24
Figure 13—Example: Enum FlowVariables	25
Figure 14—Diagram: LVDS center tap.....	33
Figure 15—Example: mixed signal Signals block	35
Figure 16—Diagram: inverter chip	36
Figure 17—Example: inverter signals block with pad numbers and coordinates.....	36
Figure 18—Example: inverter signals block, no pad numbers or coordinates	37

Figure 19—Diagram: programmable buffers	38
Figure 20—Example: programmable buffers	39
Figure 21—Example: domainInclude statement	49
Figure 22—FlowVariables example.....	51
Figure 23—Example: scalar variable initialization	52
Figure 24—Example: scalar variable initialization	53
Figure 25—Example: array initialization	53
Figure 26—Example: array initialization, all elements set to the same value	53
Figure 27—Example: multi-dimensional array, per element initialization	53
Figure 28—Example: array element access and assignment.....	53
Figure 29—Example: limits function "check".....	57
Figure 30—Example: FlowVariables block limits definitions	58
Figure 31—Example: FlowVariables block VecLocation definitions.....	59
Figure 32—Example: VecLocation parameter initializations	59
Figure 33—Example: FlowVariable window definitions.....	60
Figure 34—Example: real FlowVariable definitions.....	60
Figure 35—Example: Variables shared between Pattern and Flow	62
Figure 36—Example: SpecVariable field assignment.....	65
Figure 37—Example: array size.....	66
Figure 38—Diagram: single-site SignalMap with Signal names and device pins assigned to tester resources	71
Figure 39—Diagram: multi-site SignalMap with Signal names and device pins assignments to tester resources	72
Figure 40—Diagram: multi-site SignalMap with diagonal site layout specified, showing assignment of sites to grid positions in 4x4 grid.....	74
Figure 41—Diagram: multi-site SignalMap with counterclockwise (CCW) site layout specified, showing assignment of sites to grid positions in 2x2 grid	74
Figure 42—Diagram: Device block overview.....	75
Figure 43—Diagram: relay terminals, normally open positions	77
Figure 44—Diagram: component terminals	78
Figure 45—Example: loadboard components	78
Figure 46—Example: device site layout.....	80
Figure 47—Diagram: device site layout.....	80
Figure 48—Example: Signals, SignalGroups, chip, and package definitions	86
Figure 49—Diagram: single-site wafer test.....	87
Figure 50—Example: Device block for single-site wafer test.....	88
Figure 51—Diagram: single-site package test.....	89
Figure 52—Example: Device block for single-site package test.....	90
Figure 53—Diagram: dual chip package, dual site testing.....	91
Figure 54—Example: Device block for dual chip package, dual site testing	92
Figure 55—Diagram: pass group with two bin axes	96
Figure 56—Example: soft bin definitions—simple, common usage.....	97
Figure 57—Example: soft bin definitions—bin axes, autoincrementing bin numbers.....	97
Figure 58—Example: hard bin definitions—simple, common usage.....	102
Figure 59—Example: hard Bin definitions—autoincrementing Bin numbers	102
Figure 60—Example: BinMap using unnamed SoftBinDefs, HardBinDefs	104
Figure 61—Example: BinMap using named SoftBinDefs, HardBinDefs	104
Figure 62—Diagram: STIL.4 conceptual model	105
Figure 63—Diagram: conceptual model of flow	106
Figure 64—Diagram: conceptual model of test.....	106
Figure 65—Diagram: conceptual model for flow node.....	107
Figure 66—Diagram: STIL.4 conceptual model (FlowExtended)	108
Figure 67—Diagram: conceptual model for test and flow (FlowExtended).....	109
Figure 68—Diagram: conceptual model for flow node.....	110
Figure 69—Example: inheritance with overrides	111
Figure 70—Example: parameter initialization	117

Figure 71—Example: global, top-level, and local FlowVariables.....	123
Figure 72—Example: TestType calling subflow using inline instantiation of other TestTypes and implicit standard FlowNode	128
Figure 73—Example: Test block without TestType.....	132
Figure 74—Example: Test statement using defined TestType (FlowExtended).....	133
Figure 75—Example: FlowNode with ExitPorts.....	136
Figure 76—Example: equivalent FlowNode specification forms (FlowExtended).....	137
Figure 77—Example: Flow in TestProgram block (using Flow 2017 constructs)	139
Figure 78—Example: Flow in TestProgram block (using FlowExtended 2017 constructs)	140
Figure 79—Example: TestProgram using Flow constructs	144
Figure 80—Example: TestProgram using FlowExtended constructs	144
Figure 81—Example: standard global variable definitions	149
Figure 82—Example: minimum content standard TestBase definition	151
Figure 83—Example: FlowNode/Test interaction.....	153
Figure 84—Example: standard functional test definition.....	155
Figure C.1—Diagram: And gate with programmable output levels.....	166
Figure C.2—Example: small production test program.....	171

List of Tables

Table 1—Additional global STIL.4 reserved words	14
Table 2—Additional STIL.4 reserved words—Flow	14
Table 3—Additional STIL.4 reserved words—FlowExtended	15
Table 4—Additions to STIL.0 Table 3.....	15
Table 5—Namespaces.....	17
Table 6—Utility functions.....	22
Table 7— STIL.4 clauses by capability (Flow or FlowExtended)	27
Table 8—Signal type/subtype combinations	30
Table 9—Example: combinatorial units.....	43
Table 10—Real types	55
Table 11—FlowVariable types.....	55
Table 12—Variable attributes	61
Table 13—Operator precedence and associativity	63
Table 14—FlowVariable member functions	64
Table 15— SignalMap/Device block comparison.....	67
Table 16—Component-dependent connect statement positional significance	78
Table 17—Bin None standard attributes and data access functions	100
Table 18—Parameter directionality semantics.....	114
Table 19—STIL block parameter types	119
Table 20—Parameter attributes	123
Table 21—Actions and their legal locations	140

IEEE Standard for Extensions to Standard Test Interface Language (STIL) (IEEE Std 1450-1999) for Test Flow Specification

1. Overview

1.1 General

Standard Test Interface Language (STIL) is a standard language that provides an interface between digital test generation tools and test equipment. This standard, referred to as STIL.4, extends IEEE Std 1450™-1999 (STIL.0) to define test flows, enable STIL to tester-language translation, and provide hooks for automatic test program generation (ATPRG).¹

Test flows direct the execution and sequence of tests. STIL.4 defines TestProgram, Flow, FlowNode, Test, Bin, and FlowVariable blocks to support the definition of test flows. The STIL.4 TestProgram block invokes the test sequence that involves other STIL.4 blocks and references constructs from other STIL standards to create a complete flow. Test operations are defined down to the TestMethod or TestType/Test constructs, which identify invocation but not execution of specific test operations. Figure 1 diagrams the interaction of other STIL standards with STIL.4; when present, a STIL.4 TestProgram identifies the top of the STIL hierarchy.

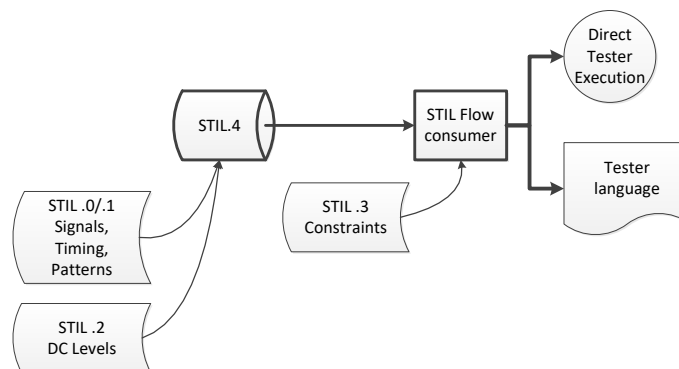


Figure 1— Diagram: STIL flow contents and application

¹ Information on normative references can be found in Clause 2.

STIL.4 supports multiple contexts of use as indicated in Figure 1. Some contexts leverage the ability to use predefined tester interfaces, and the definition of the flow can be specific to that context, in which case the STIL constructs are often directed for this specific context. Other contexts, such as ATPRG usage, require comprehensive and concise semantics in order to translate between tester environments. Not specific to context, STIL.4 identifies two levels of language use, identified with the STIL statement extensions Flow or FlowExtended.

Figure 2 shows a data flow envisioned for ATPRG using STIL. The goal is to, as comprehensively as possible, use STIL as a conduit for automatically generating test programs and retargeting them, i.e., moving them from one tester and/or test environment to another. Retargeting requires special considerations that are not addressed by this standard. Testers X and Y run STIL as the native language. Tester Z runs a proprietary language. Arrows from testers X and Y to ATPRG support incremental test program development.

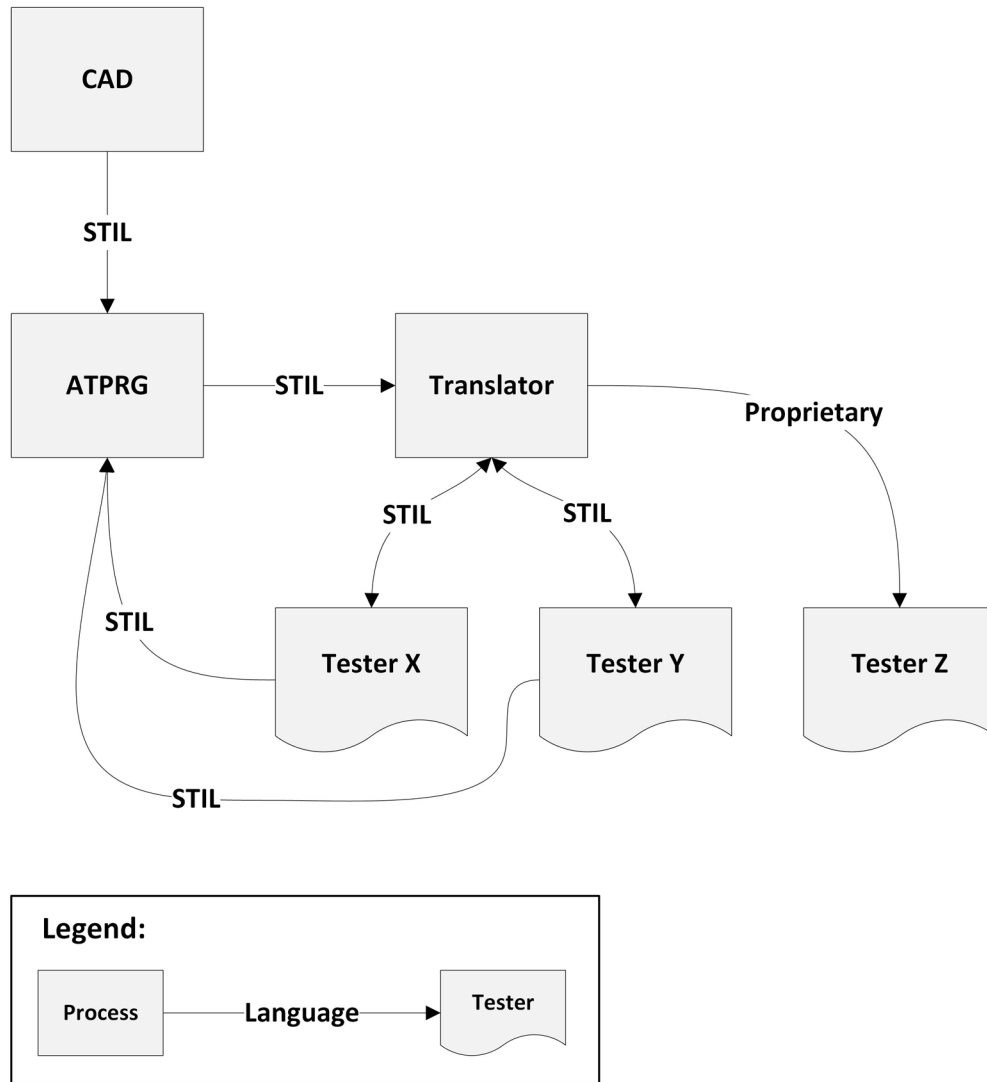


Figure 2—Diagram: ATPRG STIL data flow

1.2 Scope

This standard specifies extensions to STIL.0 that define the description of certain test flow and binning components of an integrated circuit (IC) test program in a test-hardware-independent manner. These extensions provide language constructs and semantics necessary to describe both the test program flow and the sequencing data needed to compose a test program to run on an automated test equipment (ATE) platform. The language constructs defined include structures for specifying the following:

- Order of execution of test program components
- Hierarchical test flow structures to facilitate automated modification or maintenance
- Common interfaces between the test flow environment and test program components
- Test flow variables to facilitate concurrent and serial test flow interactions
- Binning or categorization of tested ICs

The following aspects integral to test execution are specifically not addressed by this standard:

- The standardization of the interface between the prober or handler and tester is beyond the scope of STIL.4. STIL.4 requires that appropriate `AsynchronousEvent` signals shall be issued to the `TestProgram` triggering the corresponding entry-points.
- Input/output operations and exception handling.
- The definition of `TestMethods` is beyond the scope of this standard.

1.3 Purpose

STIL is the standard for the interchange of digital test data from the test generation environment (where a great deal of design information is used to generate device tests) to the test and manufacturing environment. The initial STIL standard (IEEE Std 1450-1999) addresses the essential digital test description information (i.e., signals, timing, vectors, and parameter specifications). Other aspects needed for testing devices are provided in extension activities such as this standard, which addresses test flow extensions to STIL.

The flow and binning constructs in this extension allow for developing a test program description in a common language; this common description can either be used as input to a test program generator that translates the description into the native language of specific IC ATE systems or be run directly on IC ATE systems that use IEEE 1450.4 as their native language.

2. Normative references

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 754™-2008, Standard for Floating-Point Arithmetic. ^{2,3}

IEEE Std 1450™-1999, IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data. ⁴

² The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

³ IEEE publications are available from The Institute of Electrical and Electronics Engineers (<http://standards.ieee.org/>).

⁴ This standard combined with IEEE Std 1450.2 and IEEE Std 1450.4 can be used to describe the minimum information required to generate a test program, i.e., timing, levels, patterns, and flow.