BRITISH STANDARD

BS IEC 62525:2007

Standard Test Interface Language (STIL) for Digital Test Vector Data

ICS 25.040.01; 35.060

BSI British Standards

This British Standard is the UK implementation of IEC 62525:2007.

The UK participation in its preparation was entrusted to Technical Committee GEL/93, Design automation.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 December 2007

ISBN 978 0 580 59313 0

Amendments issued since publication

Amd. No.	Date	Comments

IEC 62525

Edition 1.0 2007-11

IEEE 1450™

INTERNATIONAL STANDARD

Standard Test Interface Language (STIL) for Digital Test Vector Data

IEEE Introduction 7

1. Overview.....8

1.1 Scope.....10
1.2 Purpose.....11

2. References.....11

3. Definitions, acronyms, and abbreviations.....11

3.1 Definitions.....11
3.2 Acronyms and abbreviations.....14

4. Structure of this standard 14

5. STIL orientation and capabilities tutorial (informative).....15

5.1 Hello Tester.....15
5.2 Basic LS245 20
5.3 STIL timing expressions/"Spec" information..... 24
5.4 Structural test (scan) 29
5.5 Advanced scan 33
5.6 IEEE Std 1149.1-1990 scan 39
5.7 Multiple data elements per test cycle..... 44
5.8 Pattern reuse/direct access test..... 48
5.9 Event data/non-cyclized STIL information 52

6. STIL syntax description..... 62

6.1 Case sensitivity 62
6.2 Whitespace..... 62
6.3 Reserved words..... 62
6.4 Reserved characters 64
6.5 Comments 65
6.6 Token length 65
6.7 Character strings 65
6.8 User-defined name characteristics 66
6.9 Domain names 66
6.10 Signal and group name characteristics..... 67
6.11 Timing name constructs..... 67
6.12 Number characteristics..... 67
6.13 Timing expressions and units (time_expr)..... 68
6.14 Signal expressions (sigref_expr)..... 70
6.15 WaveformChar characteristics..... 71
6.16 STIL name spaces and name resolution..... 72

7. Statement structure and organization of STIL information 74

7.1 Top-level statements and required ordering 66
7.2 Optional top-level statements 68
7.3 STIL files 68

8. STIL statement 77

8.1 STIL syntax 77
8.2 STIL example 77

9. Header block 78

9.1 Header block syntax 78
9.2 Header example 78

10. Include statement 78

10.1 Include statement syntax 79
10.2 Include example 79
10.3 File path resolution with absolute path notation 79
10.4 File path resolution with relative path notation 79

11. UserKeywords statement 80

11.1 UserKeywords statement syntax 80
11.2 UserKeywords example 80

12. UserFunctions statement 80

12.1 UserFunctions statement syntax 81
12.2 UserFunctions example 81

13. Ann statement 81

13.1 Annotations statement syntax 81
13.2 Annotations example 81

14. Signals block 81

14.1 Signals block syntax 82
14.2 Signals block example 84

15. SignalGroups block 84

15.1 SignalGroups block syntax 84
15.2 SignalGroups block example 85
15.3 Default attribute values 85
15.4 Translation of based data into WaveformChar characters 86

16. PatternExec block 87

16.1 PatternExec block syntax 88
16.2 PatternExec block example 88

17. PatternBurst block 88

17.1 PatternBurst block syntax 89
17.2 PatternBurst block example 90

18. Timing block and WaveformTable block 90

18.1 Timing and WaveformTable syntax 91
18.2 Waveform event definitions 94
18.3 Timing and WaveformTable example 96
18.4 Rules for timed event ordering and waveform creation 97
18.5 Rules for waveform inheritance 100

19. Spec and Selector blocks 101

19.1 Spec and Selector block syntax 101
19.2 Spec and Selector block example 103

20. ScanStructures block 104

20.1 ScanStructures block syntax 105
20.2 ScanStructures block example 106

21. STIL Pattern data 107

21.1 Cyclized data 107
21.2 Multiple-bit cyclized data 108
21.3 Non-cyclized data 109
21.4 Scan data 109
21.5 Pattern labels 110

22. STIL Pattern statements 110

22.1 Vector (V) statement 110
22.2 WaveformTable (W) statement 111
22.3 Condition (C) statement 111
22.4 Call statement 112
22.5 Macro statement 112
22.6 Loop statement 113
22.7 MatchLoop statement 113
22.8 Goto statement 114
22.9 BreakPoint statements 114
22.10 IDDQTestPoint statement 114
22.11 Stop statement 115
22.12 ScanChain statement 115

23. Pattern block 115

23.1 Pattern block syntax 115
23.2 Pattern initialization 116
23.3 Pattern examples 116

24. Procedures and MacroDefs blocks 116

24.1 Procedures block 117
24.2 Procedures example 118
24.3 MacroDefs block 118
24.4 Scan testing 118
24.5 Procedure and Macro Data substitution 119

Annex A (informative) Glossary 123
Annex B (informative) STIL data model 124
Annex C (informative) GNU GZIP reference 129
Annex D (informative) Binary STIL data format 130
Annex E (informative) LS245 design description 134
Annex F (informative) STIL FAQs and language design decisions 136
Annex G (informative) List of participants 140

IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data

Sponsor

Test Technology Standards Committee of the IEEE Computer Society

Approved 18 March 1999

IEEE-SA Standards Board

Abstract: Standard Test Interface Language (STIL) provides an interface between digital test generation tools and test equipment. A test description language is defined that: (a) facilitates the transfer of digital test vector data from CAE to ATE environments; (b) specifies pattern, format, and timing information sufficient to define the application of digital test vectors to a DUT; and (c) supports the volume of test vector data generated from structured tests.
Keywords: automatic test pattern generator (ATPG), built-in self-test (BIST), computer-aided engineering (CAE), cyclize, device under test (DUT), digital test vectors, event, functional vectors, pattern, scan vectors, signal, structural vectors, timed event, waveform, waveshape

IEEE Introduction

Standard Test Interface Language (STIL) was initially developed by an ad-hoc consortium of test equipment vendors, computer-aided engineering (CAE) and computer-aided design (CAD) vendors, and integrated circuit (IC) manufacturers, to address the lack of a common solution for transferring digital test data from the generation environment to the test equipment.

The need for a common interchange format for large volumes of digital test data was identified as an overriding priority for the work; as such, the scope of the work was constrained to those aspects of the test environment that contribute significantly to the volume issue, or are necessary to support the comprehension of that data. Binary representations of data were a key consideration in these efforts, resulting in a proposal to incorporate the compression of files as part of this standard.

Limiting the scope of any standards project is a difficult thing to do, especially for a room full of engineers. However, issues that did not impact the scope as identified were dropped from consideration in this version of the standard. Subclause 1.1 covers, specifically, the capabilities that are not intended to be part of this first standard.

Early work in this consortium consisted of identifying the requirements necessary to address this problem and reviewing existing options and languages in the industry. All options proposed fell short of addressing the requirements, and the consortium started to define a new language. This work was executed with heavy leverage from some existing languages and environments, and STIL owes much to the groundwork established by these other languages.

STANDARD TEST INTERFACE LANGUAGE (STIL) FOR DIGITAL TEST VECTOR DATA

1. Overview

Standard Test Interface Language (STIL) is a standard language that provides an interface between digital test generation tools and test equipment. STIL may be directly generated as an output language of a test generation tool, or it may be used as an intermediate format for subsequent processing. Figure 1 shows STIL usage in a "pipe" format. This is meant solely as a visual analogy to emphasize the high-volume/high-throughput requirements. It is not meant to represent physical structures or implementation requirements.

STIL is a representation of information needed to define digital test operations in manufacturing tests. STIL is not intended to define how the tester implements that information. While the purpose of STIL is to pass test data into the test environment, the overall STIL language is inherently more flexible than any particular tester. Constructs may be used in a STIL file that exceed the capability of a particular tester. In some circumstances, a translator for a particular type of test equipment may be capable of restructuring the data to support that capability on the tester; in other circumstances, separate tools may operate on that data to provide that restructuring. In all circumstances, it is desirable to provide the capability to check the data against the constraints of a tester. This capability is referred to as Tester Rules Checking and is the domain of tools that operate on STIL data. As such, Tester Rules Checking operations are outside the scope of this standard.

Figure 2 shows how STIL fits into the data flow between computer-aided engineering (CAE)/simulation and the test environment. In this figure, STIL is shown as both the input and output of "STIL Manipulation Tools." STIL represents patterns as a series of cyclized waveforms that are executed sequentially. The waveform representation can be as simple as a "print-on-change" set of events, or a complex set of parameterized events. Hence, tools may be required to manipulate the data according to the requirements of a particular class of device, simulation, or tester. The output of that manipulation is still represented in STIL.

Another issue presented in Figure 2 is the need for data from the tester to be transmitted back to the CAE/simulation environment for the purpose of correlating simulation data to tester data. Although this is recognized as an important aspect of testing digital devices, it does not represent the data volume that the patterns themselves do, and is not specifically supported in this version of the standard.

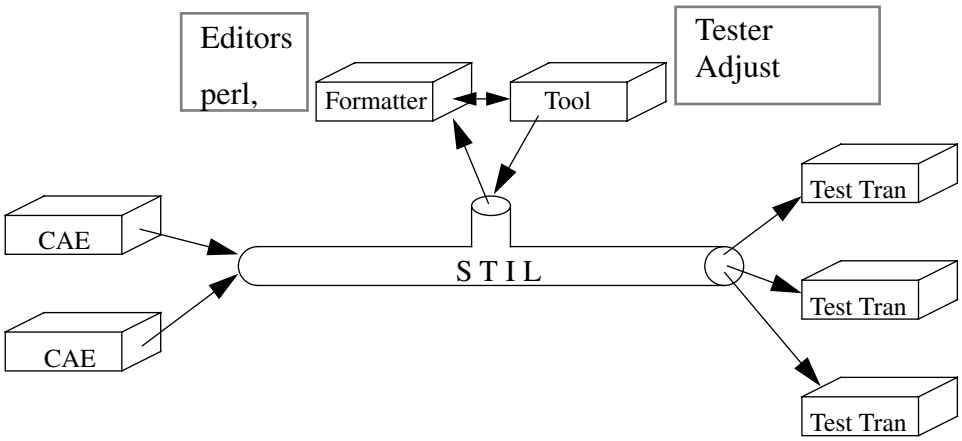

Figure 1—A conduit for transporting data from CAE to ATE

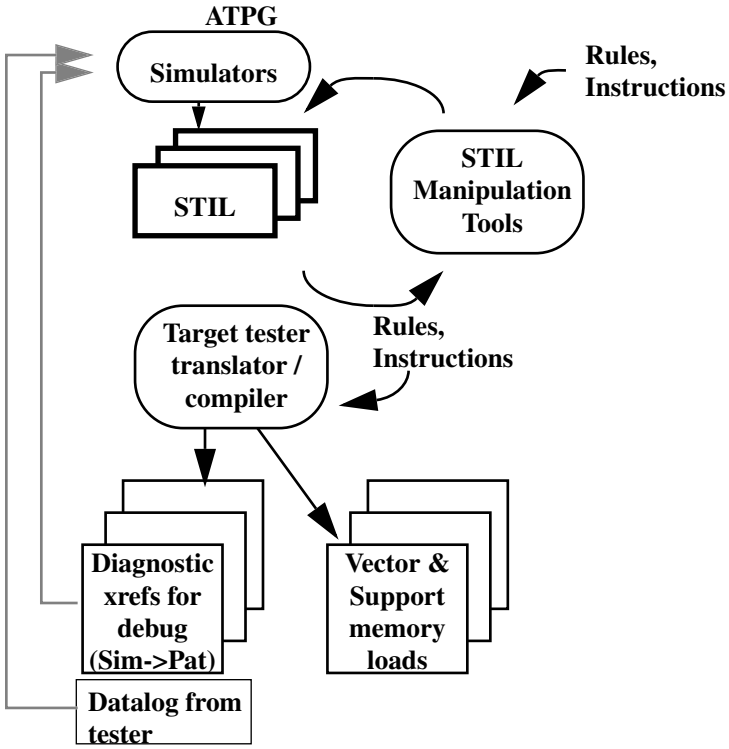

Figure 2—STIL usage model

1.1 Scope

This standard defines a test description language that:

a) Facilitates the transfer of large volumes of digital test vector data from CAE environments to automated test equipment (ATE) environments;

b) Specifies pattern, format, and timing information sufficient to define the application of digital test vectors to a device under test (DUT);

c) Supports the volume of test vector data generated from structured tests such as scan/automatic test pattern generation (ATPG), integral test techniques such as built-in self test (BIST), and functional test specifications for IC designs and their assemblies, in a format optimized for application in ATE environments.

In setting the scope for any standard, some issues are defined to not be pertinent to the initial project. The following is a partial list of issues that were dropped from the scope of this initial project:

— Levels: A key aspect of a digital test program is the ability to establish voltage and current parameters (levels) for signals under test. Level handling is not explicitly defined in the current standard, as this information is both compact (not presenting a transportation issue) and commonly established independently of digital test data, requiring different support mechanisms outside the current scope of this standard. Termination values may affect levels.

— Diagnostic/fault-tracing information: The goal of this standard is to optimally present data that needs to be moved onto ATE. While diagnostic data, fault identification data, and macro/design element correspondence data can fall into this category (and is often fairly large), this standard is also focused on integrated circuit and assemblies test, and most debug/failure analysis occurs separately from the ATE for these structures. Note that return of failure information (for off-ATE analysis) is also not part of the standard as currently defined.

— Datalogging mechanisms, formatting, and control usually are not defined as part of this current standard.

— Parametric tests are not defined as an integral part of this standard, except for optional pattern labels that identify potential locations for parametric tests, such as I_{DDQ} tests or alternating current (AC) timing tests.

— Program flow: Test sequencing and ordering are not defined as part of the current standard except as necessary to define collections of digital patterns meant to execute as a unit.

— Binning constructs are not part of the current standard.

— Analog or mixed-signal test: While this is an area of concern for many participants, at this point transfer of analog test data does not contribute to the same transportation issue seen with digital data.

— Algorithmic pattern constructs (such as sequences commonly used for memory test) are not currently defined as part of the standard.

— Parallel test/multisite test constructs are not an integral part of the current environment.

— User input and user control/options are not part of the current standard.

— Characterization tools, such as shmoo plots, are not defined as part of the current standard.