IEEE Standards

IEEE Std 1450.2™-2002

1450.2™

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

IEEE Computer Society

Sponsored by the
Test Technology Standards Committee

IEEE

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

18 March 2003

Print: SH95069
PDF: SS95069

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

Sponsor

Test Technology Standards Committee
of the
IEEE Computer Society

Approved 11 December 2002

IEEE-SA Standards Board

Abstract: This standard extends IEEE Std 1450-1999 (STIL) to support the definition of DC levels. STIL language constructs are defined to specify the DC conditions necessary to execute digital vectors on automated test equipment (ATE). STIL language extensions include structures for: (a) specifying the DC conditions for a device under test; (b) specifying DC conditions either globally, by pattern burst, by pattern, or by vector; (c) specifying alternate DC levels; and (d) selecting DC levels and alternate levels within a period, much the same as timed format events.
Keywords: automated test equipment (ATE), comparator, DC levels, device power supply (DPS), device under test (DUT), driver, driver termination, dynamic load, functional test, parametric measurement unit (PMU), power sequence, slew rate, voltage clamp

---

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

Print: ISBN 0-7381-3503-8 SH95069
PDF: ISBN 0-7381-3504-6 SS95069

Introduction

[This introduction is not part of IEEE Std 1450.2-2002, IEEE Standard for Extensions to Standard Test Interface Language (STIL) (IEEE Std 1450^{TM}-1999) for DC Level Specification.]

Standard Test Interface Language (STIL) (IEEE Std 1450-1999) was developed and approved with an intentionally constrained scope. While DC levels were explicitly excluded from that scope, it was apparent that DC levels were an area of interest and importance to the STIL user community. The P1450.2 Working Group was formed to address the extension of DC levels to the STIL standard.

Three main topics were identified as priorities for the work. These include per-pin reference levels for signal pins (e.g., VIH, VIL, VOH, VOL), device power supply levels (voltage and current), and power sequencing to the device under test. During the course of development, two other important topics were addressed. These included the capability for switching levels within a period, and for switching levels between vectors in a pattern.

Participants

Contents

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

1. Overview

This standard extends IEEE Std 1450-1999[1] (STIL) to support the definition of DC levels. The DC levels information consists of the per-pin reference levels, the device power supply (DPS) levels, and the sequencing of these levels for powering up the device, powering down the device, or changing levels of the device. The DC level definitions may be defined as static states that are established prior to execution of a pattern. They also may be selected within a pattern.

Figure 1 is a model of the test environment for a device under test (DUT) on an automatic test equipment (ATE) tester. Figure 2 is a model of the per-pin DC resources of an ATE tester. Figure 3 is a model of the differential DC resources of an ATE tester. The statements and blocks defined in this standard are defined relative to these models. Some functions represented by these models may not be available on some ATE systems. The DCSequence commands Apply and Connect load values into the hardware registers, e.g., VIL and VIH, and connect the tester resource, e.g., the driver, to the DUT, respectively.

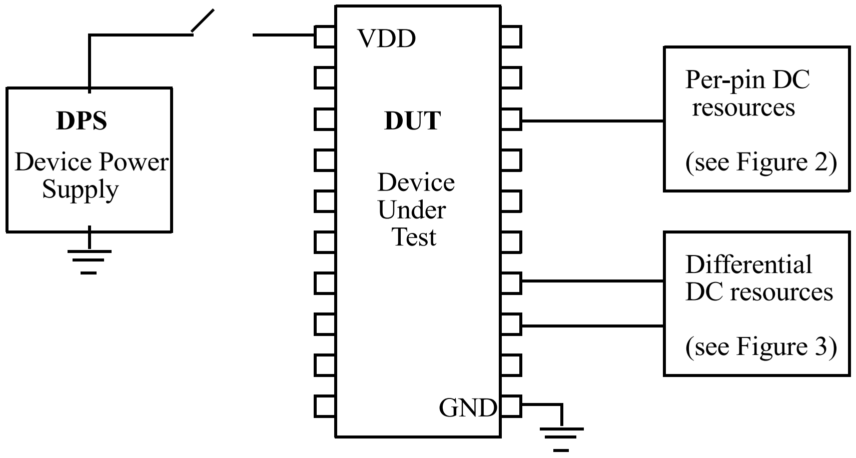

Figure 1—STIL model of DUT test environment on ATE tester

[1]Information on references can be found in Clause 2.

1.1 Scope

This standard defines the following:

a) Defines structures in STIL for specifying the DC conditions for a DUT. Examples of the DC conditions for device power supplies are DPS setup, power sequencing to the device, and power supply limiting/clamping. Examples of the DC conditions for commonly used signal references are VIL, VIH, VOL, VOH, IOL, IOH, VREF, VClampLow, and VClampHi.
b) Defines structures in STIL such that the DC conditions may be specified either globally, by pattern burst, by pattern, or by vector.
c) Defines structures in STIL to allow specification of alternate DC levels. Examples of commonly used alternate levels are VIHH, VIPP, and VILL.
d) Defines structures in STIL such that the DC levels and alternate levels can be selected within a period, much the same as timed format events.

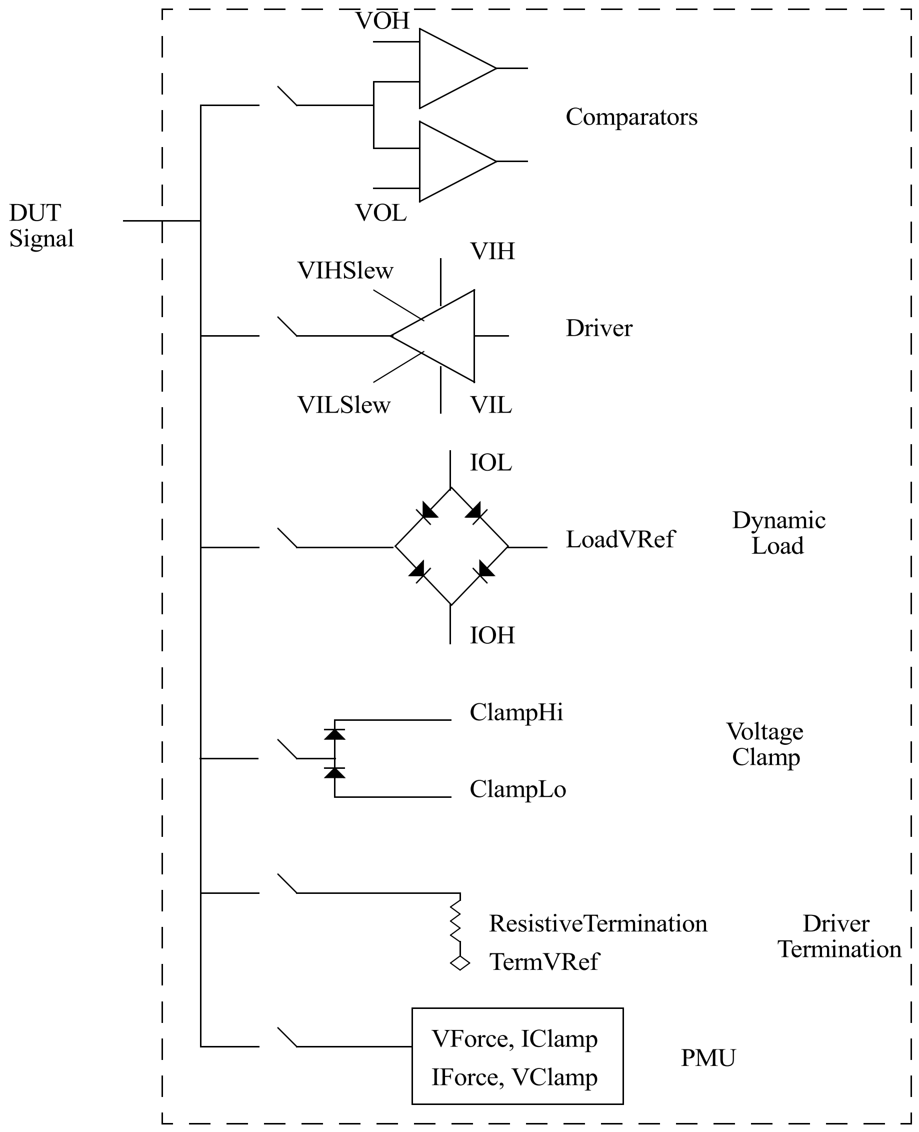

Figure 2—STIL model of per-pin DC resources of ATE tester

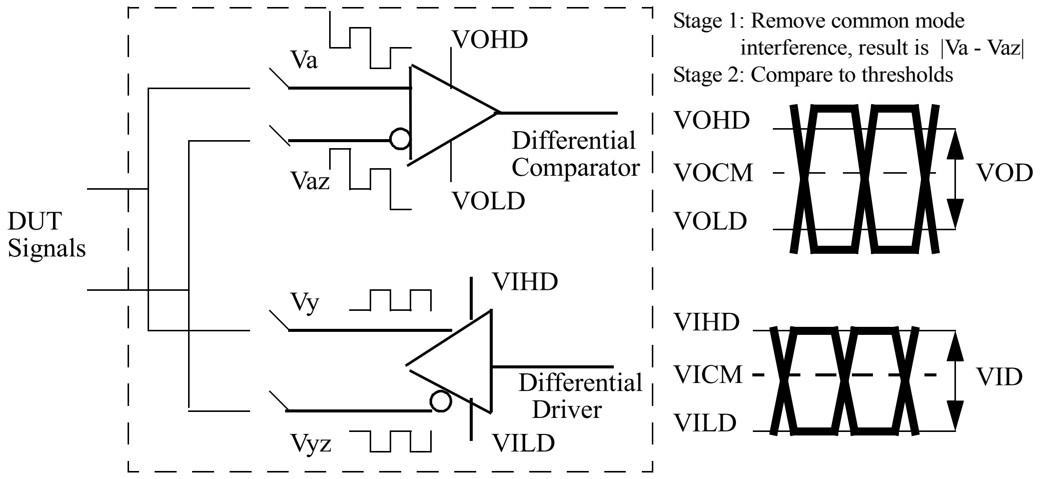

Figure 3—STIL model of differential DC resources of ATE tester

1.2 Purpose

This effort will define constructs in STIL to specify the DC conditions necessary to execute the digital vectors on ATE. This will complement the IEEE Std 1450-1999 definition, which defines structures for specification of timing and format information but does not define the DC conditions under which this information should be applied.

2. References

This standard shall be used in conjunction with the following standard. If the following standard is superseded by an approved revision, the revision shall apply.

IEEE Std 1450-1999, IEEE Standard Test Interface Language (STIL) for Digital Test Vectors.[2, 3]

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this standard, the following terms and definitions apply. IEEE 100™ [B1][4] should be referenced for terms not defined in this standard.

3.1.1 alternate input high voltage (VIHH): An alternate input high voltage forced by the ATE driver to the DUT.

[2]The IEEE standards referred to in Clause 2 are trademarks owned by the Institute of Electrical and Electronics Engineers, Inc.

[3]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

[4]The numbers in brackets correspond to those of the bibliography in Annex B.

3.1.2 alternate input high voltage (VIPP): An alternate input high voltage forced by the ATE driver to the DUT. This is typically used to generate programming pulses for electrically programmable memories and is synonymous with the commonly used term VPP in the context of programming voltage.

3.1.3 alternate input low voltage (VILL): An alternate input low voltage forced by the ATE driver to the DUT.

3.1.4 automatic test equipment (ATE): A system that provides a test capability for the automatic testing of one or more devices under test (DUT).

3.1.5 clamp current (IClamp): The maximum current allowed for a DPS or PMU.

3.1.6 clamp voltage (VClamp): The maximum voltage allowed for a DPS or PMU.

3.1.7 comparator: A component of an ATE that senses voltage response from the DUT during functional testing.

3.1.8 device power supply (DPS): A component of an ATE that provides DC voltage and/or current to the DUT power supply pins.

3.1.9 device under test (DUT): An electronic component, typically an integrated circuit, to be tested by an ATE.

3.1.10 differential comparator: A component of an ATE that senses voltage difference between two DUT outputs during functional testing.

3.1.11 differential driver: A component of an ATE that provides voltage difference to two DUT inputs during functional testing.

3.1.12 differential input high voltage (VIHD): The differential input high voltage forced by the ATE driver to the DUT.

3.1.13 differential input low voltage (VILD): The differential input low voltage forced by the ATE driver to the DUT.

3.1.14 differential input voltage (VID): The differential input voltage forced by the ATE driver to the DUT. VID = |VIHD – VILD|.

NOTE—See Figure 3.

3.1.15 differential output high voltage (VOHD): The differential output high reference voltage used by the ATE comparator to define minimum output high voltage from the DUT.

3.1.16 differential output low voltage (VOLD): The differential output low reference voltage used by the ATE comparator to define maximum output low voltage from the DUT.

3.1.17 differential output voltage (VOD): The differential output voltage received by the ATE comparator from the DUT. VOD = |VOHD – VOLD|.

NOTE—See Figure 3.

3.1.18 driver: A component of an ATE that provides voltage stimulus to the DUT during functional testing.

3.1.19 driver termination: A component of an ATE that provides impedance matching between the ATE driver and the DUT during functional testing.

3.1.20 dynamic load: A component of an ATE that provides current loading to the DUT during functional testing.

3.1.21 force current (IForce): The forcing current provided by a DPS or PMU to the DUT.

3.1.22 force voltage (VForce): The forcing voltage provided by a DPS or PMU to the DUT.

3.1.23 functional test: The process of applying functional vectors to the DUT and verifying the expected response from the DUT using an ATE. Refer to 3.1.10 of IEEE Std 1450-1999, for definition of functional vector.

3.1.24 high clamp voltage (ClampHi): The voltage level set for the high clamp circuit.

3.1.25 input common mode voltage (VICM): The input common mode voltage used to reference the differential input voltages forced by the ATE driver to the DUT. VICM = (VIHD + VILD)/2.

NOTE—See Figure 3.

3.1.26 input high voltage (VIH): The input high voltage forced by the ATE driver to the DUT.

3.1.27 input low voltage (VIL): The input low voltage forced by the ATE driver to the DUT.

3.1.28 LoadVRef: The commutating voltage of the ATE dynamic load. When no current is flowing, this is the voltage applied to the DUT.

3.1.29 low clamp voltage (ClampLo): The voltage level set for the low clamp circuit.

3.1.30 negative terminal input voltage (Vyz): The negative terminal differential input voltage from the ATE driver to the DUT. Vyz is the complement of Vy.

NOTE—See Figure 3.

3.1.31 negative terminal output voltage (Vaz): The negative terminal differential output voltage from the DUT. Vaz is the complement of Va.

NOTE—See Figure 3.

3.1.32 output common mode voltage (VOCM): The output common mode voltage used to reference the differential output voltages received by the ATE comparator from the DUT. VOCM = (VOHD + VOLD)/2.

NOTE—See Figure 3.

3.1.33 output high current (IOH): The output high current provided by the ATE dynamic load to the DUT.

3.1.34 output high voltage (VOH): The output high reference voltage used by the ATE comparator to define minimum output high voltage from the DUT.

3.1.35 output low current (IOL): The output low current provided by the ATE dynamic load to the DUT.

3.1.36 output low voltage (VOL): The output low reference voltage used by the ATE comparator to define maximum output low voltage from the DUT.

3.1.37 parametric measurement unit (PMU): A component of an ATE that forces DC voltage or current to the DUT, and measures DC current or voltage from the DUT.

3.1.38 positive terminal input voltage (Vy): The positive terminal differential input voltage from the ATE driver to the DUT.

NOTE—See Figure 3.

3.1.39 positive terminal output voltage (Va): The positive terminal differential output voltage from the DUT.

NOTE—See Figure 3.

3.1.40 ResistiveTermination: A specific resistance value for terminating the ATE driver.

3.1.41 slew rate for input high voltage (VIHSlew): The slew rate for the ATE driver when transitioning to the input high voltage, specified in units of volts per time.

3.1.42 slew rate for input low voltage (VILSlew): The slew rate for the ATE driver when transitioning to the input low voltage, specified in units of volts per time.

3.1.43 TermVRef: The voltage to which the ATE driver is terminated using ResistiveTermination.

3.1.44 voltage clamp: A protection circuit used to limit the voltage swing seen by the DUT and/or ATE.

3.2 Acronyms and abbreviations

ATE	automatic test equipment
ClampHi	high clamp voltage
ClampLo	low clamp voltage
DPS	device power supply
DUT	device under test
GND	device ground pin
IClamp	clamp current for DPS or PMU
IForce	forcing current for DPS or PMU
IOH	output high current
IOL	output low current
LoadVRef	commutating voltage of dynamic load
PMU	parametric measurement unit
ResistiveTermination	resistance value for terminating ATE driver
TermVRef	termination voltage for ResitiveTermination
Va	positive terminal differential output voltage
Vaz	negative terminal differential output voltage
VClamp	clamp voltage for DPS or PMU
VDD	device power pin
VForce	forcing voltage for DPS or PMU
VICM	input common mode voltage
VID	differential input voltage
VIH	input high voltage
VIHD	differential input high voltage
VIHH	alternate input high voltage
VIHSlew	slew rate for input high voltage
VIL	input low voltage

VILD	differential input low voltage
VILL	alternate input low voltage
VILSlew	slew rate for input low voltage
VIPP	alternate input high voltage
VOCM	output common mode voltage
VOD	differential output voltage
VOH	output high reference voltage
VOHD	differential output high reference voltage
VOL	output low reference voltage
VOLD	differential output low reference voltage
VPP	programming voltage for electrically programmable memories
Vy	positive terminal differential input voltage
Vyz	negative terminal differential input voltage

4. Structure of this standard

This standard is structured as an adjunct to IEEE Std 1450-1999. Please refer to IEEE Std 1450-1999 for information with regard to the conventions used in this standard. In most cases, the clauses in this standard are to add new constructs to existing clauses in IEEE Std 1450-1999 and are so identified in the title. The DCLevels, DCSets, and DCSequence blocks are new constructs that are introduced in this standard. All clauses in this standard are normative. Example code is provided within each clause. More complete examples of code are provided in informative Annex A.

5. Extensions to Clause 6, STIL syntax description

All constructs and restrictions for IEEE Std 1450-1999, Clause 6, are in effect here, with the following additions:

— Additional STIL reserved words specific within the context of this standard
— A new type of expression *dc_expr* as defined in 5.2.

5.1 Additional reserved words

Table 1 lists all STIL reserved words defined by this standard. Subsequent clauses in this standard identify the use and context of each of these additional reserved words.

5.2 DC expressions and units (dc_expr)

DC expressions follow the characteristics defined for timing expressions in 6.13 of IEEE Std 1450-1999. DC expressions are enclosed in single quotes and contain the same entities as timing expressions. Expressions that compute to units of volts, amps, or ohms may be used in a dc expression. In addition, complex expressions that are expressed as a ratio of two values (e.g., a slew rate of ’1V/1ns’) may be used in a dc expression.

There are three contexts where DC expressions may occur. The first context is in the Spec block as part of a spec variable definition. The second context is in the DCLevels block, to define a value associated with a DCLevels statement. The third context is in the DCSequence block, to define a value associated with a DCSequence operation.

Table 1—Additional STIL reserved words

Apply
Clamp, ClampHi, ClampLo, Comparator, Connect
DCLevels, DCSequence, DCSets, Disconnect, Driver
EndOfProgram
ForceHi, ForceLo
IClamp, IForce, InheritDCLevels, InitHi, InitialSetup, InitLo, IOH, IOL
Load, LoadVRef
PMU, PowerLower, PowerRaise
Ramp, ResistiveTermination
Termination, TermVRef
User
VClamp, VForce, VICM, VID, VIH, VIHD, VIHSlew, VIL, VILD, VILSlew, VOCM, VOD, VOH, VOHD, VOL, VOLD

5.3 Additions to STIL name spaces and name resolution (IEEE Std 1450-1999, 6.16)

DCLevels, DCSets, and DCSequence blocks augment the STIL name space as defined in Table 2. This table is incremental to IEEE Std 1450-1999, Table 6; all definitions present in that table remain unchanged.

Table 2—Additional STIL name space

STIL block	Type of name	Domain restrictions
DCLevels	DCLevels domain names	Supports a single unnamed global block and domain name blocks. Domain names shall be unique across all DCLevels blocks.
DCSets	DCSets domain names	Supports a single unnamed global block and domain name blocks. Domain names shall be unique across all DCSets blocks.
DCSequence	DCSequence domain names	Supports a single unnamed global block and domain name blocks. Domain names shall be unique across all DCSequence blocks.

6. Statement structure and organization of STIL information

This standard defines three additional top-level STIL blocks: DCLevels (Clause 10), DCSets (Clause 11), and DCSequence (Clause 12). These blocks relate to the top-level blocks of IEEE Std 1450-1999 as defined in 6.1 and 6.2.

6.1 Top-level statements and required ordering

The DCLevels and DCSets blocks, if present, have a required ordering with respect to other STIL blocks defined in IEEE Std 1450-1999. This ordering is shown in Table 3. This table is incremental to Table 7 in IEEE Std 1450-1999 and indicates the required position of these two blocks with respect to definitions present in IEEE Std 1450-1999. All other definitions and requirements specified in Table 7 of IEEE Std 1450-1999 remain unchanged.

Table 3—STIL top-level statements and ordering requirements

Statement	Purpose
Timing (from IEEE Std 1450-1999)	As defined in IEEE Std 1450-1999.
DCLevels	Defines the DC levels to be applied to signal pins and power supply pins for each PatternExec. DC expressions in this block may reference variables defined in Spec blocks, but DC variables are not resolved until the PatternExec. DCLevels blocks shall precede any DCSets or Pattern-Exec blocks that reference them.
DCSets	Identifies a set of DCLevels that may be referenced in a Pattern block. DCSets blocks shall precede any PatternExec blocks that reference them.
Selector (from IEEE Std 1450-1999)	As defined in IEEE Std 1450-1999.

6.2 Optional top-level statements

DCSequence blocks, if present, do not have a **defined** order with respect to other STIL sections as defined in IEEE Std 1450-1999. DCSequence blocks may appear any place a top-level STIL statement is allowed. This is delineated in Table 4, which is provided to be complete with Table 8 in IEEE Std 1450-1999.

Table 4—Optional top-level statements

Statement	Purpose
DCSequence	Defines the timed sequence in which power is to be applied. DCSequence blocks, if referencing other STIL data, shall be defined after the blocks that define the STIL data.

7. Extensions to Clause 8, STIL statement

The STIL statement identifies the primary version of STIL (IEEE Std 1450-1999) information contained in an STIL file and the presence of one or more standard extension constructs. The primary version of STIL is defined in IEEE Std 1450-1999.

The extension to the STIL statement allows for a block containing extension identifiers that allow for additional constructs in the STIL file. There may be multiple extension statements present, to identify the presence of multiple extension environments. The extension name and the extension statements are defined in the individual documents for those standards.

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

7.1 STIL syntax

```
STIL IEEE_1450_0_IDENTIFIER {
     ( EXT_NAME   EXT_VERSION; )+
}
```

STIL: A statement at the beginning of each STIL file.

IEEE_1450_0_IDENTIFIER: The primary version of STIL, as identified by IEEE Std 1450-1999.

EXT_NAME: The specific name of the extension. This standard is identified by the name **DCLevels**.

EXT_VERSION: The primary version of an EXT_NAME. This standard is identified by the value **2002**.

7.2 STIL example

```
STIL 1.0 {
     DCLevels 2002;
}
```

8. Extensions to Clause 19, Spec and Selector blocks

All capabilities and constructs defined in IEEE Std 1450-1999, Clause 19, are maintained for this standard, with the addition that Spec blocks are also used to define the value of the variables and expressions that are used within dc expressions.

9. Extensions to Clause 16, PatternExec block

The PatternExec block is extended to include two optional reference statements for specifying DC levels. The DCLevels block defines the DC levels for all signals referenced in the PatternExec. The DCSets block is a collection of DCLevels references which appear in the Pattern blocks associated with the PatternExec. The DCSets block is used to resolve references to DCLevels blocks in the Pattern blocks.

9.1 PatternExec block syntax

```
PatternExec (PAT_EXEC_NAME) {
 (DCLevels (DC_LEVELS_NAME);)
 (DCSets (DC_SETS_NAME);)
}
```

PatternExec: This block defines all of the top-level information for a pattern execution. Refer to IEEE Std 1450-1999 for full definition of this block.

DCLevels: This specifies the DCLevels block that contains the DC level information to use for this PatternExec. There may be multiple DCLevels blocks in an STIL file, but only one DCLevels block is associated with any given PatternExec. Any DCLevels block referenced in a PatternExec shall specify DC parameters for all signals referenced in that PatternExec except signals of type Pseudo, which are optionally specified as discussed in 9.3.

DC_LEVELS_NAME: This is an optional reference to a named DCLevels block. If no name is present, then this PatternExec shall use the unnamed DCLevels block.

DCSets: This specifies the DCSets block that is to be used to resolve all DCLevels references that appear in the Pattern blocks associated with this PatternExec. A DCSets reference statement in the PatternExec shall be present to reference any DCLevels in a Pattern. See 9.3 for additional information about the interaction of DCSets and DCLevels.

DC_SETS_NAME: This is an optional reference to a named DCLevels block. If no name is present, then this PatternExec shall use the unnamed DCLevels block.

9.2 PatternExec block example

```
PatternExec func {
  Category dc;
  Selector dc_setup;
  Timing slow;
  DCLevels slow_dc;
  DCSets all_sets;
  PatternBurst dc_patts;
}
```

9.3 DCLevels and DCSets usage in PatternExec and Pattern blocks

A DCLevels block referenced in a PatternExec with the DCLevels statement shall define DC parameters as necessary for all pertinent Signals referenced in the PatternExec. All constructs present in this block shall be applied to Signals before each Pattern referenced in this environment starts execution. When present, the DCLevels statement shall be applied even when DCSets information is also present. If no DCLevels statement is present in the PatternExec, and the DCSets statement is present in the PatternExec, then a DCLevels statement shall be present in each Pattern referenced by this PatternExec, before the first Vector statement of each Pattern. The DCLevels block referenced by this first DCLevels statement shall define DC parameters as necessary for all pertinent Signals. Subsequent DCLevels blocks referenced by DCLevels statements in a Pattern need to define only values for DC parameters that are different from the previous value established. See Annex A for a usage example of DCLevels and DCSets.

Pertinent Signals are defined to be all the In, Out, InOut, and Supply signals defined in the Signals block. When DCLevels information is provided, at least one DCLevels block shall define values as necessary across at least all of these Signals. Definitions are required even for Signals that are not referenced in a specific Pattern to define the levels information for the DefaultState value of those Signals. The effect of unspecified DCLevels values is implementation defined.

10. DCLevels block

The DCLevels block defines the DC levels to be applied to signal pins and power supply pins for each test pattern. The DCLevels block defines specific data for each DC parameter for a set of signals. Each statement in the DCLevels block specifies the characteristics for one DC parameter. Each DC parameter shall be specified at most once for a specific signal, in a single DCLevels block. Refer to Figure 2 and Figure 3 for symbolic explanation of the DC parameters and their relationship to tester resources.

A related operation to DCLevels is the measurement of DC parametrics of the DUT signals using the parametric measurement unit (PMU) or of the DUT supply pins using measure capabilities of the DPS. The DCLevels block defines the forcing conditions of the PMU or DPS, but does not address how the measurement is to be performed.

The InheritDCLevels statement allows for defining the DC characteristics across a subset of all signals, combining these definitions into a single DCLevels block that is complete for all signals referenced in a specific PatternExec. The InheritDCLevels also supports incremental definition of DCLevels for a particular signal, and allows for redefining a DC parameter by overriding an inherited value with a local definition.

10.1 DCLevels block syntax

```
 (DCLevels (DC_LEVELS_NAME) {        // DCLevels block
 (InheritDCLevels DC_LEVELS_REF;)*
 (SignalGroups GROUPS_DOMAIN;)*
 (sigref_expr {                      // Signals of type In, Out, InOut, Supply
   (VIH (dc_expr)+;)
   (VIL (dc_expr)+;)
   (VICM dc_expr;)
   (VID dc_expr;)
   (VIHD dc_expr;)
   (VILD dc_expr;)
   (ForceHi;)
   (ForceLo;)
   (InitHi;)
   (InitLo;)
   (VIHSlew dc_expr;)
   (VILSlew dc_expr;)
   (VOH (dc_expr)+;)
   (VOL (dc_expr)+;)
   (VOCM dc_expr;)
   (VOD dc_expr;)
   (VOHD dc_expr;)
   (VOLD dc_expr;)
   (IOH dc_expr;)
   (IOL dc_expr;)
   (LoadVRef dc_expr;)
   (ClampHi dc_expr;)
   (ClampLo dc_expr;)
   (ResistiveTermination dc_expr;)
   (TermVRef dc_expr;)
   (VForce dc_expr;)
   (IClamp dc_expr;)
   (IForce dc_expr;)
   (VClamp dc_expr;)
 })*
})*
```

DCLevels: This is the start of a DCLevels block, which identifies DC parameters for a set of Signals of type In, Out, InOut, and/or Supply. DC parameters for signals of type Pseudo may be specified but are not required.

DC_LEVELS_NAME: This is the name of this DCLevels block. If no name is present, then this block is global and is used when a PatternExec references a DCLevels without a name.

InheritDCLevels DC_LEVELS_REF: This statement is optional and is used to define data inheritance from another DCLevels block. DC_LEVELS_REF is a reference to the name of a previously defined DCLevels block. All parameters specified for a particular signal in an inherited DCLevels block are applied to that signal in this block; a local definition of any DC parameter overrides any inherited definition of that parameter. If there are multiple InheritDCLevels statements, the last DCLevels definitions for each signal are the initial definitions for each signal in this block. Local DCLevels definitions override the inherited definitions; the last inherited DCLevels definitions are applied to any signal without local DCLevels defined.

SignalGroups GROUPS_DOMAIN: This optional statement indicates the name of a SignalGroups block that is to be used in resolving signal group references in this DCLevels block. Referenced SignalGroups in the DCLevels block will typically be the same as those used in the Timing block and/or the PatternBurst block and require the same SignalGroup reference statements in this block to resolve *sigref_expr*. If the application of DCLevels requires different grouping of signals, then unique SignalGroups blocks are referenced with this statement.

sigref_expr: This is signal expression as defined in 6.14 of IEEE Std 1450-1999.

DC parameters for each *sigref_expr* are specified using the following entities:

VIH: This defines the drive high voltage to be applied to Signals in the specified *sigref_expr*. In most situations, a single dc_expr is present. Multiple dc_expr statements are present to support changing values during one Vector, as explained in Clause 13.

VIL: This defines the drive low voltage to be applied to Signals in the specified *sigref_expr*. In most situations, a single dc_expr is present. Multiple dc_expr statements are present to support changing values during one Vector, as explained in Clause 13.

VICM: This defines the input common mode voltage for differential Signals in the specified *sigref_expr*. VICM = (VIHD + VILD)/2 (see Figure 3).

VID: This defines the differential input voltage for Signals in the specified *sigref_expr*. VID = |VIHD – VILD| (see Figure 3).

VIHD: This defines the drive high voltage to be applied to differential Signals in the specified *sigref_expr* (see Figure 3).

VILD: This defines the drive low voltage to be applied to differential Signals in the specified *sigref_expr* (see Figure 3).

Input voltages for differential Signals may be specified either by defining VICM and VID, or by defining VIHD and VILD.

dc_expr: This is a DC expression as defined in 5.2.

ForceHi: This specifies that the VIH level shall be applied as a static drive high voltage during pattern execution, overriding the pattern data.

ForceLo: This specifies that the VIL level shall be applied as a static drive low voltage during pattern execution, overriding the pattern data.

It is an error to define both ForceHi and ForceLo on the same signal. If ForceHi is present for a signal, then a VIH statement shall be present for that signal; if ForceLo is present for a signal, then a VIL statement shall be present.

InitHi: This specifies that the VIH level shall be applied as an initial drive high voltage before pattern execution begins.

InitLo: This specifies that the VIL level shall be applied as an initial drive low voltage before pattern execution begins.

It is an error to define both InitHi and InitLo on the same signal. If InitHi is present for a signal, then a VIH statement shall be present for that signal; if InitLo is present for a signal, then a VIL statement shall be present. The presence of an InitHi or InitLo statement does not replace STIL requirements for the presence of data on that signal.

VIHSlew: This defines input slew rate for drive high voltages, specified in units of volts per time (e.g., ’1V/1ns’).

VILSlew: This defines input slew rate for drive low voltages, specified in units of volts per time (e.g., ’1V/1ns’).

VOH: This defines the compare high voltage to be applied to Signals in the specified *sigref_expr*. In most situations, a single dc_expr is present. Multiple dc_expr statements are present to support changing values during one Vector, as explained in Clause 13.

VOL: This defines the compare low voltage to be applied to Signals in the specified *sigref_expr*. In most situations, a single dc_expr is present. Multiple dc_expr statements are present to support changing values during one Vector, as explained in Clause 13.

VOCM: This defines the output common mode voltage for differential Signals in the specified *sigref_expr*. VOCM = (VOHD + VOLD)/2 (see Figure 3).

VOD: This defines the differential output voltage for Signals in the specified *sigref_expr*. VOD = |VOHD – VOLD| (see Figure 3).

VOHD: This defines the compare high voltage for differential Signals in the specified *sigref_expr* (see Figure 3).

VOLD: This defines the compare low voltage for differential Signals in the specified *sigref_expr* (see Figure 3).

Output voltages for differential Signals may be specified either by defining VOCM and VOD, or by defining VOHD and VOLD. The differential comparator functions is a two-stage process. The first stage removes common mode interference from the DUT signals, resulting in |Va – Vaz| (see Figure 3). The second stage compares the result to the output thresholds.

IOH: This defines the output high load current to be applied to Signals in the specified *sigref_expr*. IOH is typically a negative current value.

IOL: This defines the output low load current to be applied to Signals in the specified *sigref_expr*. IOL is typically a positive current value.

LoadVRef: This defines the load threshold or commutating voltage. When the Termination statement in 14.1 and 17.1 of IEEE Std 1450-1999 is used, it defines how LoadVRef should be set in order for the Vectors in the associated PatternExec to execute successfully. A Termination attribute of TerminateHigh requires that LoadVRef be set to VOH for float-state measures. A Termination attribute of TerminateLow requires that LoadVRef be set to VOL for float-state measures. A Termination attribute of TerminateOff or TerminateUnknown requires that LoadVRef be set to the float-state voltage level.

ClampHi: This defines the voltage level set for the high clamp circuit.

ClampLo: This defines the voltage level set for the low-clamp circuit.

ResistiveTermination: This defines a specific resistance value for terminating the tester driver (e.g., 50 Ohm).

TermVRef: This defines the termination voltage for ResistiveTermination.

ResistiveTermination and TermVRef are typically used to provide impedance matching to the DUT. This capability is typically implemented on ATE by using the internal impedance of the tester driver. This is distinct from usage of the Termination statement in 14.1 and 17.1 of IEEE Std 1450-1999 (see LoadVRef).

VForce: This defines the forcing voltage for a DPS (for Signals of type Supply), or for Signals of type In, Out, Inout when using a PMU in place of the tester driver/receiver.

IClamp: This defines the maximum current allowed for a DPS (for Signals of type Supply) or for Signals of type In, Out, Inout when using a PMU in place of the tester driver/receiver.

IForce: This defines the forcing current for a DPS (for Signals of type Supply) or for Signals of type In, Out, Inout when using a PMU in place of the tester driver/receiver.

VClamp: This defines the maximum voltage allowed for a DPS (for Signals of type Supply) or for Signals of type In, Out, Inout when using a PMU in place of the tester driver/receiver.

Typically, VForce and IClamp are used together to force voltage, and IForce and VClamp are used together to force current.

10.2 DCLevels block example

```
DCLevels dc_func {
  VDD {
    VForce ’vdd’;
    IClamp ’iddmax’;
  }
  VDDA {
    VForce ’vdda’;
    IClamp ’iddamax’;
  }
  ins {
    VIH ’vih1’;
    VIL ’vil1’;
    VIHSlew ’1V/1ns’;
    VILSlew ’1V/1ns’;
  }
  outs {
    VOH ’voh1’;
    VOL ’vol1’;
    IOH ’ioh1’;
    IOL ’iol1’;
    LoadVRef ’vth1’;
  }
  inouts {
    VIH ’vih2’;
    VIL ’vil2’;
```

```
      VOH 'voh2';
      VOL 'vol2';
      IOH 'ioh2';
      IOL 'iol2';
      LoadVref 'vth2';
    }
  }
```

10.3 InheritDCLevels Processing

The InheritDCLevels statement allows statements defined in one DCLevels block for a set of signals to be "inherited" in another block. The InheritDCLevels statement is defined only for the outermost level of the DCLevels block; only complete DCLevels blocks are inherited. Sections or segments of DCLevels blocks cannot be individually inherited (as is supported in IEEE Std 1450-1999 with the multiple Waveform Inherit options).

When information is inherited, each statement for each signal defined will be applied in this new block, unless that statement for a signal is redefined in this new block. Inherited information becomes part of the information contained in a DCLevels block, and that information is inherited by a successive block. This allows for a hierarchical organization of DCLevels information, where multiple DCLevels blocks may inherit successively from previous blocks, as shown in 10.4.

10.4 InheritDCLevels example

```
DCLevels ate {
   sig1 {
     VOH '1V';
     VOL '-0.3V';
   }
}

DCLevels loadboard {
  InheritDCLevels ate;
  sig1 {
     LoadVRef '0.2V';
     ResistiveTermination '50Ohm';
     TermVRef '-2V';
  }
}
DCLevels dut {
  InheritDCLevels loadboard;
  sig1 {
     LoadVRef '0.4V';
     VOH '1.2V';
  }

}
```

In the preceding example, sig1 is defined with only a VOH and VOL level for the DCLevels block named ate. The DCLevels block loadboard inherits these definitions and adds LoadVRef, ResistiveTermination, and TermVRef values into the definition for sig1. So sig1 in DCLevels block loadboard contains the following definitions: VOH='1V', VOL='–0.3V', LoadVRef='0.2V', ResistiveTermination='50Ohm', and TermVRef='–2V'.

The DCLevels block dut inherits all definitions from the loadboard block, including those definitions that were inherited in that block itself. It also redefines the inherited value of LoadVRef and VOH. So in this block, sig1 contains VOH='1.2V', VOL='–0.3V', LoadVRef='0.4V', ResistiveTermination='50Ohm', and TermVRef='–2V'.

11. DCSets block

The DCSets block identifies a set of DCLevels blocks that may be referenced in a Pattern block to change DCLevels between Vector statements. All DCLevels referenced in a Pattern associated with the PatternExec shall be identified in a single DCSets block. A DCSets reference statement in the PatternExec is required in order to reference any DCLevels in a Pattern. If no DCSets statement is present in the PatternExec then no reference to DCLevels is allowed in the associated Patterns. Refer to 9.3 for an explanation of requirements for defining DC parameters across signals in each DCLevels block identified in the DCSets block.

11.1 DCSets block syntax

```
(DCSets (DC_SET_NAME) {
(DCLevels (DC_LEVELS_REF);)*
})*
```

DCSets: Start of a DCSets block, which identifies a set of DCLevels blocks that may be referenced in a Pattern block associated with the PatternExec.

DC_SET_NAME: Name of a DCSets block. If no name is present, then it is a global block to be used for all PatternExec statements that do not specifically identify a DCSets block name.

DCLevels: Reference to a DCLevels block, which may be referenced in a Pattern block associated with the PatternExec.

DC_LEVELS_REF: Name of a previously defined DCLevels block. If no name is present, then it is a reference to the global DCLevels block.

11.2 DCSets statement example

```
DCSets set1 {
 DCLevels switch1;
 DCLevels switch2;
}
```

12. DCSequence block

The DCSequence block defines the power sequences to be used when DCLevels are applied to the device. The DCSequence block names each power sequence and defines the timed sequence in which power is to be applied. The DCSequence block is not referenced in the PatternExec block. Rather, it is assumed that the DCSequence is referenced externally, either in the device test program or by the tester operating system. This is because different power sequences may be used for the same PatternExec depending on where in the test flow the PatternExec occurs.

12.1 DCSequence block syntax

```
(DCSequence (<InitialSetup | PowerRaise | PowerLower | EndOfProgram |
                  User USER_DEFINED>) {
(SignalGroups GROUPS_DOMAIN;)*
(time_expr sigref_expr {          // Each sigref_expr is sequenced together
  (Connect (<Supply | PMU | Driver | Comparator | Load | Termination | Clamp>)*;)*
                                  // Physical connection (e.g., relay)
  (Disconnect (<Supply | PMU | Driver | Comparator | Load | Termination | Clamp>)*;)*
  (Apply (dc_expr);)*             // optional voltage or current value
  (Ramp time_expr (dc_expr);)*    // ramp duration, optional voltage target
 })*
})*
```

DCSequence: Start of the DCSequence block, which defines the timed sequence in which power is to be applied. Each DCSequence block may have a predefined or user-defined name, as described in the following paragraph. If no name is present, then it is a global DCSequence block.

There are four predefined power sequence names: **InitialSetup**, **PowerRaise**, **PowerLower**, and **EndOfProgram**. These predefined names are not required to be used. When present, they define specific power sequence functions as explained in the following sentences. The contents of each of these is user defined. **InitialSetup** defines the initial power sequence used at the beginning of the test program. **PowerRaise** defines the power sequence used when the device power supplies are increasing in absolute value of voltage. **PowerLower** defines the power sequence used when the device power supplies are decreasing in absolute value of voltage. **EndOfProgram** defines the power sequence used to power down the device at the end of the test program. All other power sequence names are user defined.

User USER_DEFINED: This identifies USER_DEFINED as a user-defined name for this DCSequence.

SignalGroups GROUPS_DOMAIN: This optional statement indicates the name of a SignalGroups block that is to be used in resolving signal group references in this DCSequence block. Referenced SignalGroups in the DCSequence block will typically be the same as those used in the Timing block and/or the PatternBurst block and require the same SignalGroup reference statements in this block to resolve *sigref_expr*. If the application of DCSequence requires different grouping of signals, then unique SignalGroups blocks are referenced with this statement.

time_expr: This is defined in IEEE Std 1450-1999. The first *time_expr* for a given DCSequence is considered to be relative to time zero (the time at which the DCSequence starts). Each following *time_expr* is relative to the start of the preceding one.

sigref_expr: This is defined in IEEE Std 1450-1999. Each *sigref_expr* identifies a set of Signals which will be sequenced together.

Connect: This specifies a physical connection to a tester resource (Supply, PMU, Driver, Comparator, Load, Termination, Clamp) (e.g., through a relay). If no resource is listed, all resources associated with each Signal referenced in this *time_expr* block shall be connected. When multiple Connect statements are used, the physical connection is performed in the order the statements appear.

Disconnect: This specifies that the physical connection to a tester resource (Supply, PMU, Driver, Comparator, Load, Termination, Clamp) (e.g., through a relay) is to be disconnected. If no resource is listed, all resources associated with each Signal referenced in this *time_expr* block shall be disconnected. When multiple Disconnect statements are used, the physical disconnection is performed in the order the statements appear.

Apply: This statement defines the target voltage or current value to apply to the specified Signals at this point in the power sequence. If no *dc_expr* is specified, the programmed voltage or current from the DCLevels block currently in effect (as determined by the device test program or tester operating system) shall be used.

Ramp: This statement shall only be used with Signals of type Supply. This statement defines the time duration used to arrive at the target voltage or current. The *time_expr* specifies the total time duration. The optional *dc_expr* specifies the target voltage or current. If no *dc_expr* is specified, the programmed voltage or current from the DCLevels block currently in effect (as determined by the device test program or tester operating system) shall be used.

12.2 DCSequence example

```
  DCSequence InitialSetup {                // Beginning of program
          '0s' 'VDD+VDDA' {                // T0
                Apply '0V';
            }
          '1ms' 'VDD+VDDA' {
                Connect Supply;
            }
          '1ms' VDD {
                Apply '100mA';        // Apply line charge current
            }
          '100us' VDD {
                Apply;               // Apply voltage programmed by test
            }
          '1ms' VDDA {
                Apply '50mA';        // Apply line charge current
            }
          '100us' VDDA {
                Apply;               // Apply voltage programmed by test
            }
          '1ms' signals {
                Connect Load;
                Connect Driver Comparator;
            }
   }

   DCSequence PowerRaise {                // Increasing VDD
          '1ms' VDD {
                Ramp '5ms';    // Ramp to programmed voltage
            }
          '1ms' VDDA {
                Ramp '5ms';
            }
          '6ms' signals {
                Apply;      // Apply new values to signals after supplies change
            }
    }

    DCSequence PowerLower {              // Decreasing VDD
          '1ms' signals {
                Apply; // Apply new values to signals before supplies change
            }
          '1ms' VDDA {
                Apply;                // Apply voltage programmed by test
            }
```

```
      '1ms' VDD {
            Apply;
        }
  }

  DCSequence EndOfProgram {             // End of program
      '0s' 'VDD+VDDA+signals' {
            Ramp '5ms' '0V';            // Ramp to 0V
        }
      '6ms' signals {
            Disconnect Driver Comparator;
            Disconnect Load;
        }
      '1ms' 'VDDA+VDD' {
            Disconnect Supply;
        }
  }
```

A timing diagram for the DCSequences in 12.2 is shown in Figure 4.

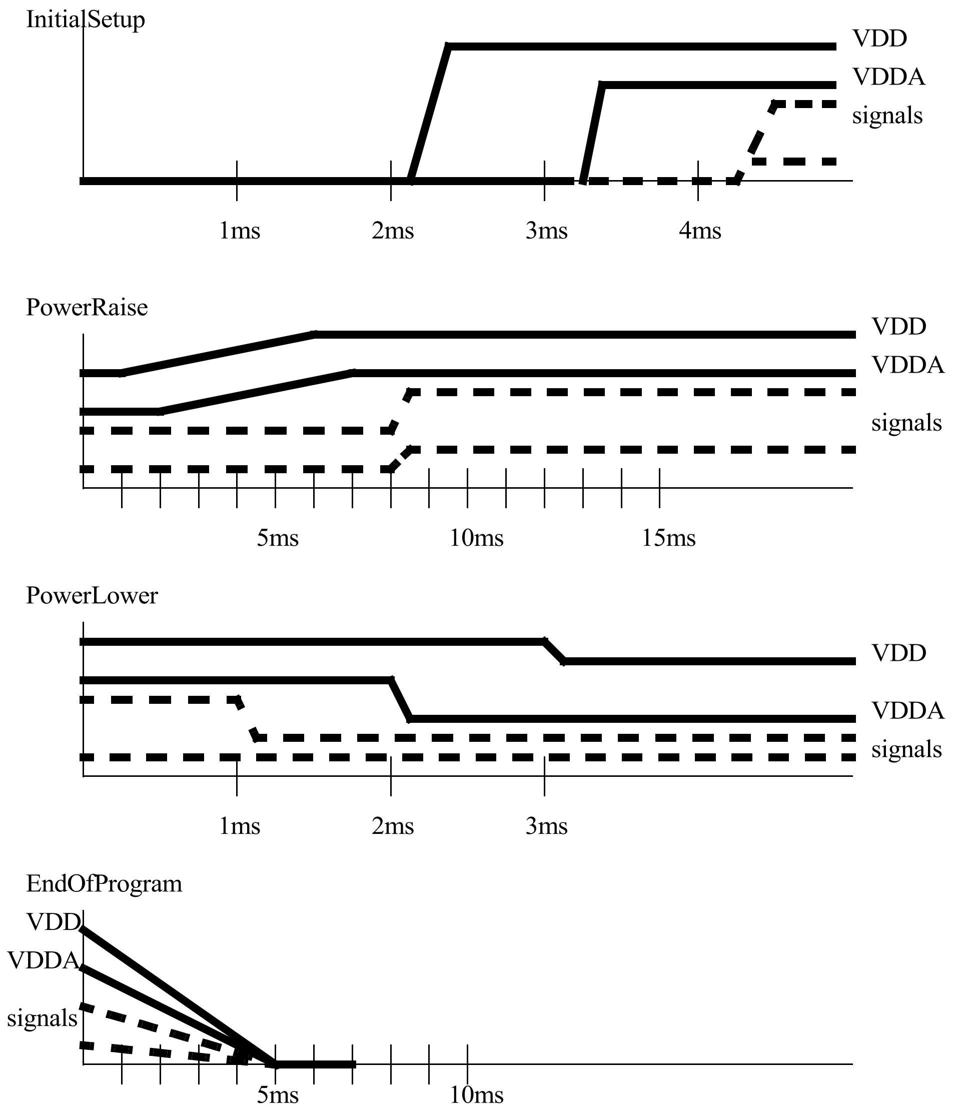

Figure 4—Timing diagram for DCSequence example

13. Extensions to Clause 18, WaveformTable block

This clause defines an extension to the definition of event characters in the WaveformTable block. The purpose of the extension is to support switching DC levels within an STIL Vector.

The extension to the WaveformTable block is limited to an extension of the definition of the event construct defined in 18.1 of IEEE Std 1450-1999.

13.1 Event definition in WaveformTable block

An "event" is an identifier that is used to signal a potential change in the state of the signal. All events have fixed definitions as specified in 18.2 of IEEE Std 1450-1999. The extension to the event definition is the addition of an optional integer suffix (0 .. n), each of which shall specify a different voltage level. The voltage levels shall be defined in the DCLevels, using multiple values in the VIH/VIL/VOH/VOL statements. The order of appearance of the voltage levels in the VIH/VIL/VOH/VOL statements shall map to the corresponding event suffixes (0 .. n). The existing IEEE Std 1450-1999 multiple-bit vector data capability can be used to define the shape of the waveform resulting from the DC level switching.

13.2 Mapping of event integers to DCLevels statements

Table 5 shows STIL events, as defined in Tables 9 and 10 of IEEE Std 1450-1999, which are extended in this standard to support an optional integer value following that event.

Table 5—STIL events referencing DCLevels statements

STIL event	DCLevels statement
`U (or ForceUp)`	`references the current VIH levels statement for this signal`
`D (or ForceDown)`	`references the current VIL levels statement for this signal`

`H (or CompareHigh)`
`h (or CompareHighWindow)`
	`references the current VOH levels statement for this signal`

`L (or CompareLow)`
`l (or CompareLowWindow)`
	`references the current VOL levels statement for this signal`

It is an error for a STIL event not listed in Table 5 to have an integer value following that event.

An event identified in Table 5 that does not have an integer value following that event shall use the first dc_expr statement present in the appropriate levels statement for that event if that levels statement is present for that signal.

The integer value following an STIL event is interpreted to reference a specific dc_expr in the appropriate levels statement for that event. The first dc_expr in the levels statement is referenced by the integer value 0 (or no integer value present), the second dc_expr is referenced by 1, etc. It is an error to specify an integer value that does not have a matching dc_expr specified.

13.3 DC levels switching example

This example demonstrates switching DC levels within a cycle in order to generate the waveform in Figure 5.

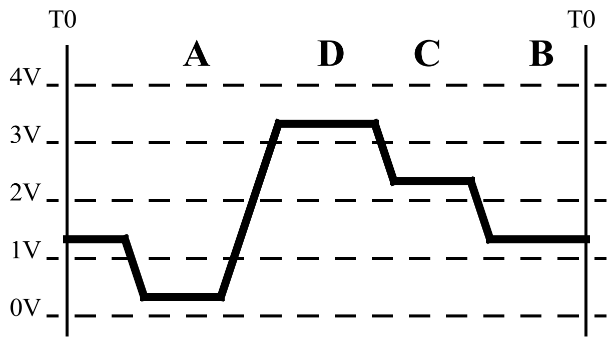

Figure 5—Example waveform generated by switching DC levels within a cycle

The following STIL fragments produce the waveform in Figure 5.

The numbers in the circles (e.g.,①) correspond to the example notes that follow.

```
DCLevels          ②
pins {
    mult {
      VIH '0.2V', '1.4V', '2.2V', '3.6V';  // define multiple drive high levels
    }
}


Timing slow {
  WaveformTable wft1 {
    Period '30ns';
    Waveforms {                         ①
       mult {
               ABCD   {
                                        '0ns'  U1;
                                        '5ns'  U0/U1/U2/U3[0];
                                        '10ns' U0/U1/U2/U3[1];
                                        '15ns' U0/U1/U2/U3[2];
                                        '20ns' U0/U1/U2/U3[3];
                 }
       }
} } }

Pattern one {
  W wft1;
   V  { mult = ADCB;  }         ③
}
```

NOTES

1—In the multiple-bit waveform definition for ABCD, WaveformChar A maps to event U0, WaveformChar B maps to event U1, WaveformChar C maps to event U2, and WaveformChar D maps to event U3.

2—In the VIH definition for group mult, the first dc_expr (0.2V) is referenced by event U0, the second dc_expr (1.4V) is referenced by event U1, the third dc_expr (2.2V) is referenced by event U2, and the fourth dc_expr (3.6V) is referenced by event U3.

3—In the multiple-bit vector ADCB, WaveformChar A specifies that event U0 will be used for bit [0] at time 5ns, thereby referencing VIH level of 0.2V. Similarly, WaveformChar D specifies that event U3 will be used for bit [1] at time 10ns, referencing VIH level 3.6V. Also, WaveformChar C specifies that event U2 will be used for bit [2] at time 15ns, referencing VIH level 2.2V. And, WaveformChar B specifies that event U1 will be used for bit [3] at time 20ns, referencing VIH level 1.4V.

14. Extensions to Clause 22, STIL Pattern statements

14.1 DCLevels statement

The DCLevels statement is used to specify a DCLevels block on a vector-by-vector basis. This provides the capability for switching DCLevels within a Pattern.

The syntax of the DCLevels statement is

DCLevels (LEVELSNAME);

LEVELSNAME is the optional name of a DCLevels block. If LEVELSNAME is not present, the reference is to the global DCLevels block.

14.2 DCLevels statement example

```
Pattern switch_levels {
  W wft1;
  DCLevels switch1;
  V   { inouts = 010101; }
  V   { inouts = 101010; }
  V   { inouts = 0L1HL0; }
  DCLevels switch2;
  V   { inouts = LHLHLH; }
  V   { inouts = HLHLHL; }
  V   { inouts = L10H01; }
}
```

Annex A

(informative)

DCLevels and DCSets usage example

The following example demonstrates DCLevels and DCSets usage in PatternExec and Pattern blocks.

The numbers in the circles (e.g.,①) correspond to the example notes that follow.

```
STIL 1.0 {
      DCLevels 2002;
}

Signals {
     en_  In;
     clk  In;
     d1   In;
     d2   In;
     a0   InOut;
     a1   InOut;
     a2   InOut;
     VDD  Supply;
     VDDS Supply;
}

SignalGroups {
     ins    'en_+d1+d2';
     clks   'clk';
     inouts 'a0+a1+a2';
     all    'ins+clks+inouts';
}

Spec all_specs {
     Category all_cats {
          per1 = '100ns';
          dely1 = '10ns';
          off1 = '25ns';
          off2 = '50ns';
          stb1 = '75ns';
          per2 = '50ns';
          dely2 = '5ns';
          off3 = '12.5ns';
          off4 = '25ns';
          stb2 = '40ns';
          vdd_low = '1.5V';
          vdds_low = '2.7V';
          vdd_hi = '1.8V';
          vdds_hi = '3.3V';
          iddmax = '500mA';
          vih_low = '2.5V';
          vil_low = '0V';
          voh_low = '2.4V';
          vol_low = '0.3V';
          ioh_low = '1mA';
          iol_low = '1.5mA';
          vth_low = '1.4V';
          vih_hi = '3V';
          vil_hi = '0V';
          voh_hi = '2.9V';
          vol_hi = '0.5V';
          ioh_hi = '2mA';
          iol_hi = '4mA';
          vth_hi = '1.7V';
          slew = '1V/1ns';
     }
}

Timing slow {
     WaveformTable ts1 {
          Period 'per1';
          Waveforms {
               ins { LH { 'dely1' D/U; } }
               clks { LC { '0ns' D; 'off1' D/U; 'off2' D; } }
               inouts { LH { 'dely1' D/U; } }
               inouts { 01ZM { 'stb1' L/H/T/X; } }
          }
     }
     WaveformTable ts2 {
          Period 'per2';
          Waveforms {
               ins { LH { 'dely2' D/U; } }
               clks { LC { '0ns' D; 'off3' D/U; 'off4' D; } }
               inouts { LH { 'dely2' D/U; } }
               inouts { 01ZM { 'stb2' L/H/T/X; } }
          }
     }
}

DCLevels low_vdd {
     VDD {                             ①
           VForce
'vdd_low';
           IClamp
'iddmax';
     }
     VDDS {
            VForce 'vdds_low';
            IClamp 'iddmax';
     }
     'ins+clks' {
           VIH 'vih_low';
           VIL 'vil_low';
           VIHSlew 'slew';
           VILSlew 'slew';
     }
     inouts {
           VIH 'vih_low';
           VIL 'vil_low';
           VIHSlew 'slew';
           VILSlew 'slew';
           VOH 'voh_low';
           VOL 'vol_low';
           IOH 'ioh_low';
           IOL 'iol_low';
           LoadVRef 'vth_low';
     }
}

DCLevels hi_vdd {
     VDD {
           VForce 'vdd_hi';
           IClamp 'iddmax';
     }
     VDDS {
            VForce 'vdds_hi';
            IClamp 'iddmax';
     }
     'ins+clks' {
           VIH 'vih_hi';
           VIL 'vil_hi';
           VIHSlew 'slew';
           VILSlew 'slew';
     }
     inouts {
           VIH 'vih_hi';
           VIL 'vil_hi';
           VIHSlew 'slew';
           VILSlew 'slew';
           VOH 'voh_hi';
           VOL 'vol_hi';
           IOH 'ioh_hi';
           IOL 'iol_hi';
           LoadVRef 'vth_hi';
     }
}

DCSets all_sets {
     DCLevels low_vdd;              ②
     DCLevels hi_vdd;
}

DCSequence User DefaultSequence {
     '0s' 'VDD+VDDS+all' {
          Connect;
          Apply;
     }
}

Selector all_typ {
     per1     Typ;
     dely1    Typ;
     off1     Typ;
     off2     Typ;
     stb1     Typ;
     per2     Typ;
     dely2    Typ;
     off3     Typ;
     off4     Typ;
     stb2     Typ;
     vdd_low  Typ;
     vdds_low Typ;
     vdd_hi   Typ;
     vdds_hi  Typ;
     iddmax   Typ;
     vih_low  Typ;
     vil_low  Typ;
     voh_low  Typ;
     vol_low  Typ;
     ioh_low  Typ;
     iol_low  Typ;
     vth_low  Typ;
     vih_hi   Typ;
     vil_hi   Typ;
     voh_hi   Typ;
     vol_hi   Typ;
     ioh_hi   Typ;
     iol_hi   Typ;
     vth_hi   Typ;
     slew     Typ;
}

PatternBurst one_pat {
     PatList {
          nolevels;
     }
}

PatternBurst two_pats {
     PatList {
          onelevel;
          bothlevels;
     }
}

PatternBurst all_pats {
     PatList {
          nolevels;
          onelevel;
          bothlevels;
     }
}

PatternExec first_func {
     Category all_cats;
     Selector all_typ;                ③
     Timing slow;
     DCLevels low_vdd;
     PatternBurst one_pat;
}

PatternExec second_func {
     Category all_cats;
     Selector all_typ;
     Timing slow;
     DCSets all_sets;
     PatternBurst two_pats;
}

PatternExec all_func {
     Category all_cats;
     Selector all_typ;
     Timing slow;
     DCLevels low_vdd;
     DCSets all_sets;
     PatternBurst all_pats;
}

Pattern nolevels {
  W ts1;
  V { all = LLLLMMM }                 ④
  V { all = LLHC001 }
  V { all = HHHLZZZ }
  V { all = LHLC010 }
}

Pattern onelevel {
  W ts1;
  DCLevels hi_vdd;
  V { all = LLLLMMM }
  V { all = LLHC001 }
  V { all = HHHLZZZ }
  V { all = LHLC010 }
  V { all = LHLC111 }
}

Pattern bothlevels {
  W ts1;
  DCLevels low_vdd;
  V { all = LLLLMMM }
  V { all = LLHC001 }
  V { all = HHHLZZZ }
  V { all = LHLC010 }
  W ts2;
  DCLevels hi_vdd;
  V { all = LLLLMMM }
  V { all = LLHC001 }
  V { all = HHHLZZZ }
```

```
   V { all = LHLC010 }
   V { all = LHLC111 }
}
```

NOTES

1—Two DCLevels blocks are defined here, low_vdd and hi_vdd.

2—DCSets all_sets is defined here, referencing both DCLevels blocks.

3—PatternExec first_func references DCLevels low_vdd and PatternBurst one_pat. Since no DCSets block is referenced in the PatternExec, the Pattern contained in the PatternBurst cannot reference any DCLevels.

PatternExec second_func does not reference any DCLevels blocks, but it does reference DCSets all_sets. In this case, the Patterns contained in the PatternBurst two_pats must have a DCLevels reference before the first Vector of each Pattern.

PatternExec all_func references DCLevels low_vdd and DCSets all_sets. The Patterns contained in PatternBurst all_pats can either reference no DCLevels blocks, in which case DCLevels low_vdd is assumed, or can reference any DCLevels blocks in DCSets all_sets.

4—Since Pattern nolevels does not contain any DCLevels references, no DCSets reference is needed in the corresponding PatternExec. In order to use DCLevels information with this Pattern, the corresponding PatternExec must contain a DCLevels reference.

Pattern onelevel references DCLevels hi_vdd. The corresponding PatternExec must contain a DCSets reference which includes DCLevels hi_vdd.

Pattern bothlevels references DCLevels low_vdd and hi_vdd. The corresponding PatternExec must contain a DCSets reference which includes both DCLevels.

Annex B

(informative)

Bibliography

[B1] IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms,* Seventh Edition.