STARC STIL-based Semiconductor Test Action Group (SSTAG)

IEEE Std 1450.0-1999

STIL Usage Guide

Revision 6.20
Oct. 17, 2011

変更履歴

Revision	Date	No	SectionTitle	Modified Item
6.2	Oct. 17, 2011	48	Problem of comments	New addition

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
6	STIL syntax description					-						
	1	55	Case sensitivity	55	1-2	Exist	Problem for case-sensitive user-defined names	STIL is a case-sensitive language. Case-sensitive user-defined names (especially for signal names) are anticipated to cause problems with the following reasons. To distinguish names only by case-sensitiveness is prohibited except for comments in the *RTL Design Style Guide* provided by STARC. Formats such as VHDL and EDIF that don't distinguish upper case and lower case exist. A malfunction may occur when make the EDA tool read the STIL data that comes from the tester.	Page1	Environment		Expressions that do not distinguish only by case-sensitiveness are recommended. (It is recommended not to use the names of which the differences are only case-sensitiveness. That is, Abc, ABC and abc should not be used together.)
	2	55	Whitespace			-						
	3	55	Reserved words			-						
	4	57	Reserved characters			-						
	5	58	Comments	58	4	Exist	Problem of comments	For comments, there are two types of description forms, //line comment and /*block comment*/. It is prescribed that comments in both forms can be described to any blank space and recognized as space (whitespace) , however the concrete treatment has been undefined.	Page138-141	Description	It is interpreted that the two types of comment description, //line comment and /*block comment*/, can be described in any place in STIL file.	Handling these description as space (whitespace) is not particularly prescribed. It depends on each application. It is recommended that //line comment and /*block comment*/ are interpreted as space of one character.
	6	58	Token length			-						
	7	58	Character strings	58	5-10	Exist	Problem of how to handle a concatenated character string	A period is a special character provided as notation to connect two strings. Using this concatenation character notation, strings with over 1024 characters, that number is a maximum token length of STIL, can be defined. However, for Inherit reference functions in the Timing block, a period represents two different things: a hierarchy divider to delimit a hierarchical block, and a concatenation character. Therefore it may cause a problem because it is unclear which interpretation should be overridden.	Page2 - 4	Application	Interpretation as a hierarchy divider should be overridden. That is, a period shall be interpreted as a hierarchy divider in Inherit reference functions in a Timing block and if there is no definition information to be referenced by this function, it is interpreted as a concatenation character.	It is recommended that an application issues the message indicating a period was interpreted as a hierarchy divider or a concatenation character in Inherit reference functions in a Timing block.

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
	8	59	User-defined name characteristics	59	4-5	Exist	Problem of how to use STIL reserved words and characters	For user-defined names with reserved characters, the requirement of double quotes are not clear. Reserved characters include special characters, SI units and prefix characters for SI units etc. (See to Table 2 on page 57 and 58 in IEEE Specification). It is unclear that when user-defined names themselves are reserved characters, or includes reserved characters, they should be enclosed by double quotes or not.	Page5 - 7	Description	When a user-defined name itself is a reserved word or includes special character(s) (Special Characters and Whitespace Characters), it shall be enclosed in double quotes. However reserved characters having multiple characters such as Cel and Ohm do not need to be enclosed by double quotes.	In order to avoid confusion, it is recommended that user-defined names should be enclosed by double quotes. Except Wafeformchar references, It is also recommended not to use single-character-user-defined names or user-defined names themselves are reserved words or reserved characters. Followings are the examples showing if the reserved words themselves or user defined names(Signal Name) can be used or not. The case-sensitive mixed name case and one letter User-defined name case are shown. However, those descriptions are simply for convenience purpose, therefore for the actual STIL description, it is recommended to be described with following the Solutions of "Problem for case-sensitive user-defined names" and "Problem of how to use STIL reserved words and characters" in STIL Usage Guide 1450.0. Examples: (1)User defined name of reserved word itself Signals { A In; a In; Alignment In; Ann In; } // not OK Signals { "A" In; "a" In; "Alignment" In; "Ann" In; } // OK (2)User defined name including reserved word Signals { AA In; aa IN; Alignment In; XAnn In; } // OK Signals { "AA" In; "aa" IN; "XAlignment" In; "XAnn" In; } // OK （These expressions are recommended.) (3)User defined name of Special Character itself Signals { ; In; * In; } // not OK Signals { ";" In; "*" In; } // OK (4)User defined name including Special Character Signals { name; In; name* In; } // not OK Signals { "name;" In; "name*" In; } // OK (5)User defined name for Engineering Prefix itself Signals { E In; P In; T In; } // not OK Signals { "E" In; "P" In; "T" In; } // OK (6)User defined name including Engineering Prefix Signals { EE In; PP In; TT In; } // OK Signals { "EE" In; "PP" In; "TT" In; } // OK （These expressions are recommended.) (7)User defined name for SI Unit itself of single character Signals { A In; V In; F In; s In; } // not OK Signals { "A" In; "V" In; "F" In; "s" In; } // OK (8)User defined name including SI Unit itself of single character Signals { AA In; VV In; FF In; ss In; } // OK Signals { "AA" In; "VV" In; "FF" In; "ss" In; } // OK （These expressions are recommended.) (9)User defined name for SI Unit itself of multiple character(Cel, Hz, Ohm) Signals { Cel In; Hz In; Ohm In; } // OK Signals { "Cel" In; "Hz" In; "Ohm" In; } // OK (As expressions, these are recommended but these are reserved characters themselves so should not be used anyway.) (10)User defined name including SI Unit of multiple character(Cel, Hz, Ohm) Signals { XCel In; XHz In; XOhm In; } // OK Signals { "XCel" In; "XHz" In; "XOhm" In; } // OK（These expressions are recommended.) (11)User defined name for min, max itself Signals { min In; max In; } // OK Signals { "min" In; "max" In; } // OK (As expressions, these are recommended but these are reserved characters themselves so should not be used anyway.) (12)User defined name including min, max Signals { Xmin In; Xmax In; } // OK Signals { "Xmin" In; "Xmax" In; } // OK (This expressions are recommended)

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
				59	9-37	Exist	Problem of how to handle double-quoted user-defined names	Double quotes are not allowed as part of signal names in the IEEE Specification. The IEEE Clarification describes that both double-quoted signal name and unquoted signal name may be expressed in the Signals block. If double quotes can not be used as part of signal names, double-quoted signal name and unquoted signal name indicate the same signal name, resulting in duplicated definition of the same signal name. Therefore, double-quoted and unquoted signal names must be distinguished in a single STIL file. Though signal names are not enclosed in double quotes in the netlist file, in many cases signal names are enclosed in double quotes in the STIL file generated by the ATPG tool. When unquoted signal names as well as double-quoted signal names are included in the STIL file, an application can not determine which of signals to use.	Page8 - 12	Application	The application which handles STIL expressions can process the same user-defined name which is double-quoted and unquoted (i.e., ”Xyz” and XYZ used for the same type name such as signal name, signal group name and domain name) as an error for duplicated definition of the same user-defined name. This is because double-quoted and unquoted user-defined names might pose a high risk for usage errors through readability or misunderstanding of oral instructions even though the application can differentiate between them. The application shall process user-defined names internally in a unified format which is either one of unquoted format or double-quoted format.	It is recommended to differentiate user-defined names by adding numbers and characters, and not to differentiate them by double-quoted and unquoted names. It is recommended that the application generating the STIL file outputs double-quoted user-defined names. For bus signal, “BUS”[0] expression is recommended rather than “BUS[0]” expression with double quotes enclosing the whole bus expression. “BUS”[0..7] can be interpreted as a bus signal expression, whereas “BUS[0..7]” can not be determined as a bus signal expression since it may be interpreted as a character string containing special characters.
				59	34	Exist	Problem for newline and tab characters included in user-defined names	For user-defined names in examples in a IEEE Specification, expressions using blank characters exist, but expressions using tab and newline characters do not exist. A tab character, a newline character and a blank character can not be differentiated in a hard copy respectively. So in the case that the STIL file containing tab, newline or blank characters in user-defined names is transferred, it need to be clarify how to describe user-defined names using tab, newline or blank characters.	Page13 - 14	Environment	When a user-defined name containing whitespace (blank, tab or newline character) is described, it is difficult for the STIL creator or user to check visually how blank, tab and newline character are used in expressions. If the STIL expression is modified, it is not recognized, which increases the likelihood of problems. Therefore, we should set the minimum requirements to prevent problems occur during a visual check . As a result, user-defined names containing blank may be used, but user-defined names containing tab and newline characters shall not be used.	When the STIL expression contains a user-defined name including tab or newline character, it is recommended that an application handles it as an error of STIL expression by issuing a message to request the correction of the error.
				59	1	-	Problem of User-defined names	The description of User-defined name is defined as follows, 6.8 User-defined name characteristics (P59) There are several categories of user-defined names in STIL: signal and group references, WaveformChar references, WaveformTable references, variable references, UserKeywords, labels, and domain names(*). It is not clear how to handle the description not specified as User-defined names, such as ScanChain name of ScanStructures block, ScanCells name and UserFunction name. (*)domain names : Domain names provide a mechanism to reference the data defined in a named block. When a domain name is present for a SignalGroups, Procedures, MacroDefs, PatternBurst, Timing, Selector, Pattern or ScanStructures block, that domain name shall be specified in a “reference” statement in order to make use of the data in that block. Also in P59, there is a description that the block name of Table 6 (P66) Spec, PatternExec etc. are User-defined name.	Page 119 - 120	Description	The block name, put on the block statement becomes Named Block, is interpreted to be handled as User-defined name. Therefore the ScanChain name in the ScanStructures block is handled as User-defined name. Also, the name put on the UserFunctions, as same as the UserKeywords, is interpreted to be handled as User-defined name. ScanCells name in the ScanStructures block, as same as the Signals name, is interpreted to be handled as User-defined name. Therefore User-defined name is interpreted as the collective designation of names of person creating STIL or of the names tools and designers can put. It is interpreted the same way on the STIL extension spec (IEEE1450.1, 1450.2, 1450.3, 1450.6) where found the same problem. The detail should be refereed to the each STIL Usage Guide.	
	9	59	Domain names			-						

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number[*1]	Exist/Nonexist[*2]	Title	Problem description	Link to problem-related material	Classification[*3]	Solution	Remarks
	10	60	Signal and group name characteristics	60	2	-	Problem of BUS signal name description	BUS description can be described by ascending order and descending order of the Index Number. However, it is not clear how to handle the case described in the same Index Numbers. Example: “BUSA” [0..31] //Index Number is ascending order, ”BUSA”[0], “BUSA”[1], … the same as “BUSA”[31] “BUSA” [31..0] //Index Number is descending order, ”BUSA”[31], “BUSA”[30], …the same as” BUSA”[0] “BUSA” [15..15] // Index Number is the same integer value, it is not clear how to handle this case. ,,, (*)	Page121	Description	For the BUS description, if the same integer value is described (*)in the Index Number, it is interpreted to show the same signal as the BUS signal of that integer value which is “signal name” (integer value).	However, it is recommended not to use the description (*) which the same integer value is in the Index Number. In addition, regarding the description of one Bus signal, if there is a mix of the description(*) which the same integer value in the Index Number and the BUS signal description of that integer value, to avoid any confusion, that is fine to handle it as an error of the application side.

STIL 1450.0 Usage Guide

Feb.7, 2011 (Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
	10	60	Signal and group name characteristics			-						
	11	60	Timing name constructs			-						
	12	60	Number characteristics			-						
	13	61	Timing expressions and units (time_expr)			-						
	14	63	Signal expressions (sigref_expr)			-						
	15	64	WaveformChar characteristics	65	19-20	Exist	Problem of pattern repeated description scope	On the list description of repeated Characters of vector flag \r, it is not clear if \r can be described or not and the interpretation for the case \r is described on the list description is vague. <Format> \rN XXX Example) sigref_expr=\r3 00\r2 11 0 01;	Page122 - 124	Description	The space after the repeated times of \r is handled as a part of delimiter of the count fields which indicates the repeated times of \r. It is not handled as the first white space of \r repeated description and even if \r is described in the repeated list description, there is no contradiction. Therefore it is interpreted to be able to describe \r in the vector flag repeated list description of \r. Example) V { sigref_expr=\r3 00\r2 11 0 01; } is the same as below.	Note1) On the STIL specification, WhiteSpace means space, tab, newline character. Note2) On the repeated list description of \r, it cannot be described to switch the character types by using \h or \d. It is possible to switch the character types from hex description or decimal description by using \w. \h01 11 \r2 A\wLH 0101 00010001 1010 LH 1010 LH 0101
	16	65	STIL name spaces and name resolution			-						
7	Statement structure and organization of STIL					-						
	1	69	Top-level statements and required ordering			-						
	2	70	Optional top-level statements			-						
	3	70	STIL files			-						
8	STIL statement					-						
	1	70	STIL syntax			-						
	2	70	STIL example			-						
9	Header block					-						
	1	71	Header block syntax			-						
	2	71	Header example			-						
10	Include statement			71	1-11	Exist	Problem of whether the Include statement can be nested	In the STIL file referenced with the Include statement, when there is a nesting description of Include statement which references from another STIL file by using an Include statement, it is unknown how to handle it because the validation of the description is not clearly stated in the specification. That is, in STIL file "B" referenced by Include statement from STIL file "A", when there is a nesting description which references STIL file "C" by an Include statement, the validation of using the Include statement is not clearly stated in the specification.	Page15 - 16	Description	The nesting of the Include statement shall be allowed and the number of nesting shall be unlimited.	It is recommended that nesting expressions with the Include statement resulting an endless loop are regarded as an error, and the error message is issued by the application which handles the STIL file. That is, in STIL file (B) referred by the Include statement from STIL file (A), when nesting expressions referring STIL file (A) back by the Include statement exist, it is an endless loop so the application should output an error message.
	1	72	Include statement syntax			-						
	2	72	Include example			-						
	3	72	File path resolution with absolute path notation			-						
	4	72	File path resolution with relative path notation			-						
11	UserKeywords statement					-						
	1	73	UserKeywords statement syntax			-						

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
	2	73	UserKeywords example			-						
12	UserFunctions statement					-						
	1	74	UserFunctions statement syntax			-						
	2	74	UserFunctions example			-						
13	Ann statement					-						
	1	74	Annotations statement syntax			-						
	2	74	Annotations example			-						
14	Signals block					-						
	1	75	Signals block syntax	76	63-67	Exist	Problem of pattern description allowed when the Alignment statement is defined	It is unclear whether both WFCs and numerical values may be described as pattern to the Vector statement when you describe the Alignment statement in defining signals or a signal group in the Signals or SignalGroups block.	Page66 - 67	Description	When you describe the Alignment statement in defining signals or a signal group in the Signals or SignalGroups block, if Hex or Dec numerical values are used for the target Vector statement, the sequence of bit data from MSB or LSB specified by the Alignment statement is assigned to the signals and it becomes a WFC string corresponding to the sequence of bit data. The Alignment statement becomes invalid for the order of WFCs. When you describe WFCs for patterns, the numbers of WFCs and signals have to be the same.	

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
	2	77	Signals block example			-						
15	SignalGroups block			77	1-2	Exist	Problems of how to define the SignalGroups block	Pin group defined in the SignalGroups block can define 0 or more pins as a group but the specification of how to define 0 pin is not clear. Also the interpretation when there was the STIL expression referring a 0 pin group was not clear. For example, when patterns are defined to a 0 pin group in the Patten block or timing is defined to a 0 pin group in the Timing block, the interpretation is unknown.	Page17 - 20	Description	When defining a pin group name, if the number of pin is zero, the following two expressions are possible, but Type 1 shall be used. (Type 1) GroupName = ''; (Type 2) GroupName = ; If there is the STIL expression referring a 0 pin group, it shall be interpreted that there is no expression corresponding to a pin group name as a result of the reference. For example when patterns are defined to a 0 pin group in the Patten block, it is interpreted as no pattern definition to the pin group name. Also when timing is defined to a 0 pin group, it is interpreted as no timing definition to the pin group name.	When defining a pin group with zero pin, Type 1 is allowed in the scope of 1450.0 STIL and Type 1 and 2 are allowed in the scope of 1450.1. In the scope of 1450.0, Type 2 shall not be allowed and to avoid confusion with different STIL versions, Type 1 is recommended. It is recommended to define signals by using signals of type Pseudo as much as possible in order to make missing signals clear in expressions, except when an empty signal group is required. Except when pin group definition using the same name of the pin is required, it is recommended not to use the same name to the group as much as possible.
	1	77	SignalGroups block syntax			-						
	2	78	SignalGroups block example			-						
	3	78	Default attribute values			-						
	4	79	Translation of based data into WaveformChar characters			-						
16	PatternExec block			80	1-15	Exist	Problems for multiple PatternExec blocks	In the STIL file which multiple PatternExec blocks are defined, since there is no rule of the order for processing multiple PatternExec blocks, it is unknown how to handle them.	Page21 - 22	Application	In the STIL file which multiple PatternExec blocks are defined, the multiple PatternExec blocks shall be understood that all the PatternExec blocks will be executed by executing each of the multiple PatternExec blocks in the order of description. Specific handling of these blocks performed by an application shall not be defined, and handling these blocks shall be dependent on individual application.	
	1	81	PatternExec block syntax			-						
	2	81	PatternExec block example			-						
17	PatternBurst block					-						
	1	82	PatternBurst block syntax			-						
	2	82	PatternBurst block example			-						
18	Timing block and WaveformTable block					-						
	1	84	Timing and WaveformTable syntax	86	11-13	Exist	Problem of how to interpret multiple events specified to the same time	It is unknown how to interpret multiple events defined to the same event time for one WFC. Also it is unknown whether clearly specified ForceOff definition is required to switch to the output mode. For example, how to interpret the following definitions is unknown. `[ O { 0ns Z; 0ns X; } ]`	Page23 - 24	Description	In the case of multiple events specified to the same event time, an input and an output events may be specified to the same time respectively. When WFC is defined with multiple events specified to the same event time, it shall be understood that these events are executed in the order of the expression. When multiple input events or output events are defined to the same event time, basically it is interpreted as an error and the application outputs an error message. An input and output events shall be handled to be unaffected mutually. Therefore the exclusive operation that the driver is turned off as turning the output mode on will not be performed. If the driver needs to be turned off on a tester, the event must be clearly defined with ForceOff(Z).	It is up to each application how to continue the process after the output of the error message, such as performing in the order of expressions, or as a result, taking the latter expression. It is recommended to output a message about the performance. It is recommended that an application on a terser issues a warning or error message if production test presents behaviors different from the STIL expression on a device testing expected.

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number[*1]	Exist/Nonexist[*2]	Title	Problem description	Link to problem-related material	Classification[*3]	Solution	Remarks
	2	87	Waveform event definitions	88	Table 11	Exist	Problem for the Marker event expression	It is unknown how to handle the Marker event expression. Since there is no example expression in the IEEE Specification, it is unknown which case is assumed to require this expression.	Page25	Application	Since the Marker event is not for test behavior, it's up to each application how to handle it. For example, a typical application just ignores it.	For the application which does not require to handle the Marker event information, it is recommended to issue a message indicating that the Marker event definition in the STIL file was ignored.

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number *1	Exist/Nonexist *2	Title	Problem description	Link to problem-related material	Classification *3	Solution	Remarks
	3	89	Timing and WaveformTable example			-						
	4	90	Rules for timed event ordering and waveform creation			-						
	5	93	Rules for waveform inheritance			-						
19	Spec and Selector blocks					-						
	1	94	Spec and Selector block syntax	94	10-22	Exist	Problem when Meas is selected in the Selector block	When Meas is selected, the timing value is set as it executes Meas, so the timing value is not clearly set until the actual measurement of the device on the tester is executed. Therefore, for some applications how to handle timing values is important. For example, when a simulation runs using the STIL expressions with Meas selected, the execution timing can not be set.	Page26 - 28	Application	For STIL expressions containing an execution timing set by Meas, the same handling is not necessarily required for applications which can operate and can not operate in accordance with STIL expressions. An individual application shall be responsible for handling an execution timing set by Meas.	It is recommended that application output a message how to handle an execution timing set by Meas.
	2	96	Spec and Selector block example			-						
20	ScanStructures block					-						
	1	98	ScanStructures block syntax	98	8	Exist	Problem of ScanCells' ScanCell Names description	Description of CELLNAME-LIST is defined as follows in STIL1450.0 ScanCells. 20.1 ScanStructures block syntax(P98) ScanCells: Identifies the scan cells comprising the scan chain. CELLNAME-LIST is ScanLength scan cell NAMEs separated by whitespace. Inversion is also specified by interleaving the “!” character between scan cell NAMEs (also includes before the first scan cell and after the last scan cell). Scan cell NAMEs are ordered from the first scan cell to be shifted (input) to the last scan cell to be shifted (output). In the ScanCells statement of the ScanStructures Block, it is not clear whether only description of the FF instance name is required, or the pin name of the FF is also required to the Scan Cell Names description. In addition, it is impossible to identify the instance name description whether it is Verilog format, VHDL format or own format, therefore it depends on tools' implementation to decide the instance name description format. This is causing problems in execution of the parallel loading simulation.	Page113-115	Application		About the information needed for parallel loading simulation, since it can be described in the ScanCellType block and ScanCells block in the STIL1450.1, it is recommendable to describe it in the STIL 1450.1. It is recommended to write FF(cell) instance name in the ScanCells block instead of ScanCells statement, describe FF(cell) pin name in the scan input/output of the ScanCellType. Also, it is recommended to describe the instance name of each FF(cell) with either Verilog format or VHDL format. It depends on the application to adapt which format. The detail should be referred to “Problem of ScanCells definitions” in the STIL Usage Guide 1450.1. Example) STIL 1.0 { Design 2005; } ScanCellType FD1 { CellIn "MASTER" : "SIN1" ; // Define the scan in name SIN1 of the cell type CellOut "SLAVE" : "SOUT1" ; //Define the scan out name SOUT1 } ScanCellType FD2 { CellIn “MASTER” : “SIN2” ; // Define the scan in name SIN2 of the cell type CellOut “SLAVE” : “SOUT2” ; // Define the scan out name SOUT2 } ScanCells { "top.a1" "FD1" ; "top.a2" "FD1" ; "top.a3" "FD2" ;.....; } // “top.a1”,“top.a2” should refer to ScanCellType FD1 and "top.a3” should refer to ScanCellType FD2.
	2	99	ScanStructures block example			-						
21	STIL Pattern data					-						
	1	100	Cyclized data			-						
	2	101	Multiple-bit cyclized data	101	1-22	Exist	Problem with DatabitCount pattern omission	When an empty Vector statement follows after the Vector statement in which DataBitCount is described, it is not clear whether the last state is carried over, or the same description as the Vector statement immediately before is described, into the empty Vector statement.	Page57 - 59	Description	When an empty Vector statement follows after the Vector statement in which DataBitCount is described, it is interpreted that the same description as the Vector statement immediately before is described in the empty Vector statement.	

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number[*1]	Exist/Nonexist[*2]	Title	Problem description	Link to problem-related material	Classification[*3]	Solution	Remarks
	3	102	Non-cyclized data			-						
	4	102	Scan data			-						
	5	103	Pattern labels			-						
22	STIL Pattern statements					-						
	1	103	Vector (V) statement	103	1-9	Exist	Problem with multiple definitions of Pattern	The interpretation is not clear when different multiple patterns are specified to the same pin in one address. Also as for the pattern description of one address to a bidirectional pin, the description is not clear when the pattern (WFC) for input and the pattern (WFC) for output are specified separately. As a result, a simulation error occurs due to the difference in interpretation.	Page29 - 31	Application	In a Vector or a Condition statement, the multiple definitions of pattern for the same pin in one address is not allowed. That is, in a Vector or Condition statement, specifying patterns (WFC) to each pin is allowed at least once. Also for a bidirectional pin, when a pattern description of one address is specified to a pattern for input and a pattern for output separately, it is interpreted as an error so that the application output an error message.	It is up to each application how to continue the process after the output of the error message, such as taking only the latter pattern (WFC) definition. It is recommended to output a message about the performance.
	2	104	WaveformTable (W) statement			-						
	3	104	Condition (C) statement			-						
	4	105	Call statement			-						
	5	105	Macro statement			-						
	6	106	Loop statement	106	1-8	Exist	Problem for timing change and pattern omission in the Loop block	When the pattern expressions are omitted in the first Vector statement within the Loop block, or timing is changed within the Loop block, it is unknown which test patterns are applied.	Page32 - 35	Description	When patterns are omitted in the first Vector statement within the Loop block, that means no patterns are assigned to all signals, the pattern (WFC) described preceding the Loop block shall be taken. It shall be understood that the same test patterns in the Loop block are run by the first and later loop execution. When timing is not defined on the top in the Loop block, and timing is changed in the Loop block, it shall be understood that the timing information used with the first Vector statement is the same as one used in the Vector statement preceding the Loop block. This timing information shall be regarded as timing used in the first test pattern (Vector statement) within the Loop block, and referenced by first and later loop execution.	It is recommended not to omit patterns at the top of the Loop block. At the beginning of the Loop block, it is recommended that not to omit timing specification. Especially when switching the timing in the middle of the pattern of Loop block, it is recommended to specify timing to explicitly specify the timing of the beginning pattern of the Loop block. As is the case with the Loop block, it is recommended to describe patterns and timings definitely without omitting them for the branch destinations in the Goto and Start statements.
	7	106	MatchLoop statement			-						
	8	107	Goto statement			-						
	9	107	BreakPoint statement			-						

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
	10	107	IddqTestPoint statement	107	1-3	Exist	Problem of the measurement order in IDDq tests	In an IDDq test, it is more efficient to start the measurement in order from highest fault detection capability point to lowest. However, in the current IddqTestPoint statement specification, information concerning the measurement order of multiple IDDq test points, such as fault coverage, can not be described. Therefore, in order to create efficient test programs, users need to refer the result file of fault simulation to set the measurement order in the IDDq test from highest fault detection capability point to lowest.	Page60 - 62	Description		It is recommended to enhance the IddqTestPoint statement and use UserKeywords Detection when describing the information concerning the measurement order of IDDq test points.
	11	108	Stop statement			-						
	12	108	ScanChain Statement	108	1-10	Exist	Problem for the ScanChain statement	The IEEE Specification does not define clearly how to make multiple ScanChain blocks active.	Page36 - 38	Description	When the ScanChain statement is not described in the Pattern block (or in the Procedures and MacroDefs blocks), all scan chains shall be interpreted as being active. When the ScanChain statement is described, only the scan chain specified in the ScanChain statement is active. It shall be understood that all the scan chains have to be described one by one in the ScanChain statement in order to make multiple scan chains active. Since the statement to make scan chains inactive is not provided, it shall be understood that the active scan chains become inactive by executing only one shift operation to given scan chains after the ScanChain statement. To make the scan chains active again, the ScanChain statement needs to be described.	When the scan chain to which the shift operation of a set of patterns executed after the ScanChain statement differs from the scan chain which has declared as active by ScanChain statement, and that difference can be identified in the application side, whether to continue the process after outputting an error or warning depends on each application. In order to make multiple scan chains active, users can describe it simply by using ActiveScanChains (Clause 16) and ScanChainGroups (Clause 14) in STIL1450.1, the STIL extended specification to design environment.

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
23			Pattern block			-						
	1	108	Pattern block syntax			-						
	2	109	Pattern initialization			-	Problem with pin definitions in the first Vector statement in the Pattern block	(1) If pins not described in the first Vector statement in the Pattern block are described in the Include file or in the Procedure block, which is used as a separate Pattern block in the middle of the Pattern block, it is unknown whether the DefaultState value should be used for the undescribed pins or this case should be a violation of the specification.	Page63 - 65	Description	If pins not described in the first Vector statement in the Pattern block are described in the Include file or in the Procedure block, which is used as a separate Pattern block in the middle of the Pattern block, it shall be a violation of the specification because this case should be treated the same as pins not used in the first Vector in the Pattern block being used in a later Vector.	It is recommended that the states of all the pins used in a Pattern block should be described in the first Vector statement in that Pattern block.
								(2) If there are pins not described in the first Vector statement in the Procedures block or Include file used as a separate Pattern block at the top of the Pattern block, it is unknown whether the DefaultState value should be used for pins not described at the top of the Procedures block or Include file, or they should be handled as a violation of the specification.	Page63 - 65	Description	If there are pins not described in the first Vector statement in the Procedures block or Include file used as a separate Pattern block at the top of the Pattern block, and the pins are not described through the Pattern blocks, the pin at the top of the Pattern block accepts the DefaultState value. If pins not described in the first Vector statement in the Procedures block or Include file used as a separate Pattern block are described in the middle of the Pattern block, it shall be a violation of the specification.	
	3	109	Pattern example			-						
24			Procedures and MacroDefs blocks	109	Table13	Exist	Problem for the state of pins undefined in the Procedures block	When not all pins are described in Procedures, how to handle the pins not described is unknown.	Page39 - 40	Description	For the pins not described, use the values when defined in the DefaultState statement, and if not defined in the DefaultState statement, the default values of the DefaultState statement as the pins' values. The default value of the DefaultState statement for input shall be Z and the default value of the DefaultState statement for output without clear specification shall be X. DefaultState will be invalid if patterns (WFC) were described just once in the Condition statement or the Vector statement somewhere in the Pattern block.. That is, it is not defined by the DefaultState statement, however the Procedures block is independent so DefaultState is always valid in the first pattern.	As the same as the MacroDefs block and the Pattern block, at the beginning of the Procedures block, it is recommended that the state of all pins used in the Pattern block, except the pins to which the DefaultState statement is valid shall be described in the Condition statement or the Vector statement.
	1	110	Procedures block			-						
	2	111	Procedures example			-						
	3	111	MacroDefs block			-						
	4	111	Scan testing			-						
	5	112	Procedure and Macro Data substitution	112	16-20	Exist	Problem for the # operator for pattern substitution	When substituting for pattern, the # operator allows the substitution to the scanpattern, or to the Vector statement in the MacroDefs block, or the Vector statement in the Procedures block. But the substitution to the latter Vector statement is not clearly specified in the standard.	Page41 - 46	Description	It shall be understood that to # in the Vector statement in the MacroDefs block or # in the Vector statement in the Procedures block, parameter data (patterns) in the Macro statement or in the Call statement is substituted to the # according to the order of the Alignment statement's definition. (cf. In the case of the % operator substitution, to % in the Vector statement in the MacroDefs block , or in the Vector statement in the Procedures block, only one of parameter data (patterns) specified in the Macro statement or the Call statement is substituted to the % according to the Alignment statement definition.) When the parameter data (patterns) described in the Macro statement or the Call statement is defined to a pin or a whole set of the pins in a pin group, the substitution by the Vector statements with the mixing of # and % operators shall not be allowed to the pin or the pin group. When the parameter data (patterns) described in the Macro statement or the Call statement is defined to a pin group pin-wise, the substitution by the Vector statements without the mixing of # and % operators through the Vector statements to any pin in the pin group shall be allowed.	

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
		114		114	1-17	Exist	Problem of pattern substitution	When a pattern is substituted to an argument of Procedure or Macro, it is unclear how to substitute the pattern if it is not sufficient for the argument.	Page125 - 137	Description	In the SignalGroups names substituted at Call, if there is a same name as SignalGroups names in the main body definition, substitute the pattern (argument) on the order of Vector statement of SignalGroups name matching the head side. This operation should be repeated until there is no SignalGroups name. However the substitution pattern which is at the Call of matched SignalGroups name can only be used for substituted Vector statement (# or % Vector statement) of following same name body and if it is remained, should be abandoned, and it never be used for the signal substitution of deploying response. Next, if the substituted main body Vector statement is remained and substituted SignalGroups name at Call is remained, substitute the pattern described at responded signal of deploying SignalGroups on the order of description at Call. If there is no more main body Vector statement for substitution, the substitution operation is finished. It should be noted that if the substituted signals in the remained Signal Groups names are Even after above, if there is a signal remained in the main body substituted Vector statement, and there is a shortage of patterns to substitute at Call, the pattern specified just before the signal substituted Vector statement appeared first in the main body definition, should be substituted to the signal. At the body definition, if there is no pattern (WFC) specified before the substituted Vector statement, and there is a shortage for substitution patterns, Procedure Call will be an error and to Macro, the pattern just before the Call will be succeeded and substituted.	

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number[*1]	Exist/Nonexist[*2]	Title	Problem description	Link to problem-related material	Classification[*3]	Solution	Remarks
Some items of the tester control information, which is required for creating a test program, are missing in the current STIL standard, as shown below. Until these items are included in the STIL standard or Clarification, it is recommended to follow the proposed examples and commonly utilize these items by using UserKeywords, Pragma, or NameMaps. Also if they are included as the STIL standard by IEEE, we shall comply with it. For UserKeywords, please see problem-related materials.												
			Waveform event definitions			Exist	Problem when the driver switching time is more than the input waveform edge timing	When you set the driver switching time more than the input waveform edge and try to describe the waveform within the scope of the current STIL specification, you need to use different WFCs or switch the WaveformTable for two different cases of input waveforms; one, which has switched from output to input, and another, which has stayed in input mode. On a tester supporting driver-enable timing, to perform a test, test patterns may need to be rewritten in some case due to the above constraint .	Page47 - 51	Application		If the input timing is different for an input waveform switching from output to input and for another staying in input mode, it is recommended to describe (see the examples in the related materilas) in accordance with the STIL specifications. However, on a tester supporting driver-enable timing, in order to avoid the complicated pattern data description, even in the following case of application's exception handling, the driver switching timing is made at 50ns as the same as L, H, T, and X when using WFC 0 and 1. So this case can be regarded and processed as the same as the examples in the problem-related materials. That is, 01{'0ns' D/U; } LHTX { '50ns' Z; '80ns' L/H/T/X; } for test patterns defined in the above, pattern data "0, 1" of a cycle which changes from output to input 01{'0ns' D/U; } LHTX { '50ns' Z; '80ns' L/H/T/X; } AB{ '50ns' D/U; } can be regarded and processed as pattern data "A, B" defined in the immediately above. When an application operates as said above, to notate this process explicitly, you may use a drive event UserKeywords, e, which is newly defined in the problem-related materials. When the application supporting exception handling mentioned above outputs STIL data, it can be either in the form of complying with the STIL specifications using a drive event UserKeywords e.
			TestType statement			Exist	Problem of the test type description	No statement to describe test types is provided in the specification.	Page52 - 53	Description		It is recommended to describe test types using User USER_DEFINED of the Purpose statement which can be defined in Pattern or PatternBurst block statement of PatternInformation block in 1450.6. For the detail, please see the proposed Solution of “Problem of pattern attribute description such as test pattern type and test method” in STIL Usage Guide 1450.6.
			InfiniteLoop statement			Exist	Problem of the description of infinite loops	In a tester, the test pattern may need to get into an infinite loop, but no statement is provided.	Page54 - 55	Description		It is recommended to define InfiniteLoop by UserKeywords and describe it in a pattern block when you get the test pattern into an infinite loop at test pattern execution.
16	PatternExec block											
			TestType statement in PatternExec			Exist	Problem of the test type description in the PatternExec block	When executing a test, to describe a statement to define a test type in the PatternExec block is required, but such statement is not provided.	Page56	Description		It is recommended to describe the TestType statement in the PatternExec block.
	ADCUnits block											
						Exist	Problem of DAC units settings of testers in ADC conversion tests	Unable to describe the settings of DAC units in ADC conversion tests.	Page68-70	Description		It is recommended to describe the DAC unit settings of the tester using ADCUnits block defined by UserKeywords when declaring AMD P7 in STILextentions.
	VrefLevels block											

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
						Exist	Problem of condition descriptions for reference power supply in ADC conversion tests	Unable to describe the conditions for reference power supply in ADC conversion tests.	Page71-72	Description		It is recommended to specify the conditions for reference power supply using VrefLevels block defined by UserKeywords when declaring AMD P7 in STILextentions.
	DoutCapUnits block											
						Exist	Problem of description for data loading unit of testers in ADC conversion tests	Unable to describe the data loading unit of testers in ADC conversion tests.	Page73-74	Description		It is recommended to specify the data loading unit using DoutCapUnits block defined by UserKeywords when declaring AMD P7 in STILextensions.
	PatternExec block（DCSets）											
						Exist	Problem of block-name reference in ADC conversion tests	Unable to reference ADCUnits/VrefLevels/DoutCapUnits block names in PatternExec.	Page75-77	Description		Define ADCUnits/VrefLevels/DoutCapUnits block names defined by UserKeywords in DCSets Block to enable referencing in PatternExec.
												Enable referencing ADCUnits/VrefLevels/DoutCapUnits block names defined by UserKeywords directly in Pattern Exec block.
	Pattern statement											
						Exist	Problem of description for ADC conversion operation patterns in ADC conversion tests	Unable to describe the ADC conversion operation patterns in ADC conversion tests.	Page78-79	Description		It is recommended to define StartAnalogVoltage, StopAnalogVoltage and ADCMeasLoop by UserKeywords, and describe the starting and ending points of forcing DAC unit outputs to device analog input using the StartAnalogVoltage and StopAnalogVoltage statements, and describe the repeat of ADC conversion operation using the ADCMeasLoop statement.

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
	DACUnits block											
						Exist	Problem of ADC unit description in DAC conversion tests	Unable to describe the settings for ADC units in DAC conversion tests.	Page80-83	Description		It is recommended to specify the conditions for the ADC units using DACUnits block defined by UserKeywords when declaring AMD P7 in STILextensions.
	VrefLevels block											
						Exist	Problem of condition descriptions for reference power supply in DAC conversion tests	Unable to describe the conditions for reference power supply in DAC conversion tests.	Page84-85	Description		It is recommended to specify the conditions for reference power supply using VrefLeveles block defined by UserKeywords when declaring AMD P7 in STILextensions.
	PatternExec block（DCSets）											
						Exist	Problem of reference of block names of in DAC conversion tests	Unable to reference ADCUnits/VrefLevels block names in PatternExec.	Page86-88	Description		Define the block names of DACUnits/VrefLevels defined by UserKeywords in DCSets block to enable referencing in PatternExec Enable referencing the block names of ADCUnits/VrefLevels defined by UserKeywords directly in PatternExec block
	Pattern statement											
						Exist	Problem of descriptions for DAC conversion operation patterns in DAC conversion tests	Unable to describe the patterns of DAC conversion operation in DAC conversion tests.	Page89-90	Description		It is recommended to define DACMeasPattern, Cap and DaCUnits by UserKeywords, define DACMeasPattern block in Pattern block, and describe the patterns of DAC conversion operations using Cap statement and DACUnits statement.
	PeriodCounter block											
						Exist	Problem of settings for measuring instruments for testers in PLL tests	Unable to specify the settings for measuring instruments for testers in PLL tests.	Page91-92	Description		It is recommended to define PeriodCounterLoop by UserKeywords and use PeriodCounter block in Pattern block in order to describe the settings for measuring instruments for the testers.
	PatternExec block											
						Exist	Problem of reference of measuring instruments' settings of testers in PLL tests	In PatternExec, unable to refer to the settings of measuring instruments of testers in PLL tests.	Page93	Description		It is recommended to enable referencing PeriodCounter block defined by UserKeywords in PatternExec block.
	Pattern statement											
						Exist	Problem of description to specify the loop of Period counter' s measurement patterns in PLL tests	Unable to describe the loop of the measuring patterns of Period counters in PLL tests.	Page94	Description		It is recommended to describe the loop of the measuring patterns of Period counters using PriodCounterLoop statement in Pattern block after defining PeriodCounterLoop by UserKeywords.
	MemoryBist block											
						Exist	Problems of settings for MemoryBist pattenrs	Unable to describe the settings for MemoryBist patterns.	Page95-100	Description		Describe the settings for MemoryBist patterns using MemoryBist block defined by UserKeywords.

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
						Exist		Unable to describe the settings for MemoryBist patterns per MemoryBist circuit and Redundancy(BIRA) circuit.	Page95-100	Description		Use Instance block in MemoryBist block defined by UserKeywords.
						Exist		Unable to describe the settings for MemoryBist patterns per RAM macro.	Page95-100	Description		Use MacroName block in MemoryBist block defined by UserKeywords.
						Exist	Problem of referencing the settings of MemoryBist patterns	Unable to refer to MemoryBist pattern settings in PatternBurst.	Page101	Description		It is recommended to enable referencing MemoryBist block names defined by UserKeywords in PatternBurst block.
						Exist	Problem of locations for BIST patterns in Pattern block	Unable to specify the locations to begin BIST pattern of MemoryBIST pattern or vector locations for output of BIST data in Pattern block.	Page102-103	Description		It is recommended to define BistStart and BistPatternOut by UserKeywords and describe locations to begin BIST using BistStart statement, and vector locations to load BIST data outputs using BistPatternOut statements in Pattern block.
						Exist	Problem of locations for repair patterns in Pattern block	Unable to specify the locations to begin repair pattern of MemoryBist pattern and vector location to load repair data in Pattern block.	Page104-105	Description		It is recommended to define BiraStart and RepairPatternOut by UserKeywords, and describe the locations to begin repairing using BiraStart statement, and vector locations of loading repair data using RepairPatternOut statements in Pattern block.

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number*1	Exist/Nonexist*2	Title	Problem description	Link to problem-related material	Classification*3	Solution	Remarks
22	STIL Pattern Data											
				100		Exist	Problem of release/capture clock definitions which should be distinguished in the transition pattern	STIL doesn't have the statements which indicate its pattern attribute, therefore it cannot be simply distinguished what test pattern it was by looking at only STIL data. For example, it cannot be distinguished whether it was the Transition test pattern or not, or whether such test pattern method was "LoC" or "LoS".	Page106-112	Description		For fault diagnosis tool, it is necessary to recognize what information (attribute) ATPG output-STIL has. To distinguish such, in STIL Usage Guide 1450.0Rev4.00, it was recommended to describe with UserKeywords PatType;, but changed that recommendation, we now recommend to describe using User USER_DEFINED of Purpose statement which can be defined in Pattern or PatternBurst block statement in PatternInformation block. For the detail, please see the proposed Solution of "Problem of pattern attribute description such as test pattern type and test method" in the STIL Usage Guide 1450.6
						Exist		When executing fault diagnosis as Transition pattern, Launch(*)/Capture clock cannot be distinguished. Therefore there is a problem of executing fault diagnosis.	Page106-112	Description		As for Launch/Capture clock, it is necessary to distinguish them. In STIL Usage Guide 1450.0Rev4.00, it was recommended to describe with UserKeywords ClockRelations; to distinguish them, however we revised that recommendation and now we recommend it to define Launch/Capture clock with using UserKeywords ClockRelations RELATION NAME in Internal block and clarify the difference of Launch/Capture clock by describing ClockRelations RELATION_NAME which is in right after the Vector sentence of Launch/Capture clock in Pattern block. For the detail, please see the proposed Solution of "Problem of Launch/Capture clock definitions which should be distinguished in the Transition pattern" in the STIL Usage Guide 1450.6
				98		Exist	Problem of Scan-Chain Structures	ATPG creates the patterns for the scan-in of the shadow scan chain as well, so parallel loading is required for the shadow scan chain. However, shadow scan chain cannot be simply described with current ScanStructures.	Page116-118	Description		Currently, there is no description that indicates the branching of ScanChain, so it is recommendable to define the branched point using UserKeywords(ScanBranch) to clarify the ScanChain structure of ScanStructures.

Rev	Modification history	Description
1.00	Jan. 25, 2006	The first edition publication
2.00	Mar. 26, 2007	Revisions were made to STIL-UG comments as the followings: No.1: Problem Description and SSTAG's Remarks were Revised. No.2: Problem Description, SSTAG's Solution and Remarks were revised. No.3: Problem Description, SSTAG's Solution and Remarks were revised. No.4: SSTAG's Remarks was revised. No.5: Problem Description, SSTAG's Solution and Remarks were revised. No.6: Problem Description, SSTAG's Solution and Remarks were revised. No.7: Problem Description, SSTAG's Solution and Remarks were revised. No.8: Problem Description and SSTAG's Solution were revised. No.9: Problem Description, SSTAG's Solution and Remarks were revised. No.10: SSTAG's Solution was revised. No.11: The title, Problem Description, SSTAG's Solution and Remarks were revised. No.12: The title, Problem Description, SSTAG's Solution and Remarks were revised. No.13: SSTAG's Remarks was revised. No.14: SSTAG's Solution and Remarks were revised. No.15: Problem Description, SSTAG's Solution and Remarks were revised. No.16: Problem Description and SSTAG's Solution were revised. The explanatory material for WhiteSpace was revised. Link page numbers were revised.
3.00	Aug. 1, 2008	The problem from No17 to No24 in the following table were added.
4.00	Aug. 26, 2009	The problem from No25 to No43 in the following table were added.
5.00	Sept. 30, 2010	The problem from No44 to No45 in the following table were added. No3: Supplements were added on Remarks. No22: Remarks were revised. No41: Remarks were revised. No42: Remarks were revised.
6.00	Feb. 7, 2011	Modified the solution sample. Modified the solution exsample No.9.
6.10	Jul. 15, 2011	The problem from No46 to No47 in the following table were added.
6.20	Oct. 17, 2011	The problem from No48 in the following table were added.

*4 Numbers in the above table correspond to those of the below table.

No	Title
No.1	Problem for case-sensitive user-defined names
No.2	Problem of how to handle a concatenated character string
No.3	Problem of how to use STIL reserved words and characters
No.4	Problem of how to handle double-quoted user-defined names

STIL 1450.0 Usage Guide

Feb.7, 2011（Revision 6.00)

*1 Indicates the line number on which a problems was encountered in IEEE Specification. The line number is started from 1 in each section, i.e., 1-5 (the first line to fifth line).

*2 The item which presented a problem in the discussion is indicated with Exist, and the item which presented no problem is indicated with Nonexist. The item for which no discussion proposal was submitted is indicated with a hyphen (-).

*3 Description: Indicates the contents that should be commonly recognized to describe or interpret STIL data.
Application: Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.
Environment: Indicates the contents that should be commonly recognized to mutually use the STIL data in the test environment supporting STIL.

IEEE Std 1450.0-1999 Specification				Problem							SSTAG	
Chapter	Section	Page	Item	Page	Line number[*1]	Exist/Nonexist[*2]	Title	Problem description	Link to problem-related material	Classification[*3]	Solution	Remarks

No.5	Problem for newline and tab characters included in user-defined names
No.6	Problem of whether the Include statement can be nested
No.7	Problems of how to define the SignalGroups block
No.8	Problems for multiple PatternExec blocks
No.9	Problem of how to interpret multiple events specified to the same time
No.10	Problem for the Marker event expression
No.11	Problem when Meas is selected in the Selector block
No.12	Problem with multiple definitions
No.13	Problem for timing change and pattern omission in the Loop block
No.14	Problem for the ScanChain statement
No.15	Problem for the state of pins undefined in the Procedures block
No.16	Problem for the # and % operators for pattern substitution
No.17	Problem of pattern description allowed when the Alignment statement is defined
No.18	Problem of DataBitCount pattern omission
No.19	Problem with pin definitions in the first Vector statement in the Pattern block
No.20	Problem of the measurement order in IDDq tests
No.21	Problem when “driver switching time>input waveform edge timing”
No.22	Problem of the test type description
No.23	Problem of the description of infinite loops
No.24	Problem of the test type description in the PatternExec block
No25	Problem of DAC units settings of testers in ADC conversion tests
No26	Problem of condition descriptions for reference power supply in ADC conversion tests
No27	Problem of description for data loading unit of testers in ADC conversion tests
No28	Problem of referencing the block names in ADC conversion tests
No29	Problem of description for ADC conversion operation patterns in ADC conversion tests
No30	Problem of ADC unit description in DAC conversion tests
No31	Problem of condition descriptions for reference power supply in DAC conversion tests
No32	Problem of block-name reference in ADC conversion tests
No33	Problem of descriptions for DAC conversion operation patterns in DAC conversion tests
No34	Problem of settings for measuring instruments for testers in PLL tests
No35	Problem of reference of measuring instruments’ settings of testers in PLL tests
No36	Problem of description to specify the loop of Period counter’s measurement patterns in
No37	Problems of settings for MemoryBist pattenrs
No38	Problem of referencing the settings of MemoryBist patterns
No39	Problem of locations for BIST patterns in Pattern block
No40	Problem of locations for repair patterns in Pattern block
No41	Problem of release/capture clock definitions which should be distinguished in the transition pattern
No42	Problem of ScanCells' Cell Names descriptions
No43	Problem of Scan-Chain Structures
No44	Problem of User defined names
No45	Problem of BUS signal name description
No46	Problem of pattern repeated description scope
No47	Problem of pattern substitution
No48	Problem of comments

Reference material for user-defined names

What is the user-defined name?

The user-defined name indicates the following ten categories.
(1) Signal name (Signal name defined in the Signals block)
(2) Group name (Group name defined in the SignalGroups block)
(3) WFC
(4) WaveformTable block name
(5) Variable name (Variable name defined in the Spec block)
(6) User keyword (Keyword defined in the UseKeywords and UserFunctions statements)
(7) Label
(8) Domain name (SignalGroups block name, Timing block name, Spec block name, Selector block name, ScanStructures block name, PatternBurst block name, PatternExec block name, Procedures block name, MacroDefs block name, Pattern block name)
(9) Other block names (ScanChain block name, Category block name, Macro block name, Procedure block name: block name defined in the Procedures block)
(10) Others (Cell name: each cell name of scan flip-flops defined in the ScanCells statement)

Reference material 2

Reference material for whitespace (space, tab and newline characters)

Note on whitespace in the WFC list

The WFC list may contain whitespace. (A newline character, tab and blank may be described freely.)
To describe a WFC list without whitespace in the statement used for pattern expressions such as the Vector statement and Condition statement, care must be taken not to exceed 1024 characters due to the restriction for token length. To describe characters exceeding 1024, whitespace is used for a separator between tokens.

Note on whitespace in the Annotation statement

Since any character strings other than the token “*}” can be used for annotations, whitespace can be described freely in the Annotation statement.

Note on whitespace in the Header block (TITLE_STRING, DATE_STRING and SOURCE_STRING)

The newline character can be used freely in TITLE_STRING and SOURCE_STRING.
To describe a WFC list without whitespace, care must be taken not to exceed 1024 characters due to the restriction for token length.
To describe characters exceeding 1024, whitespace is used for a separator between tokens.
Whitespace can not be described freely in DATE_STRING since DATE_STRING expression is restricted to the output format defined with the ctime() function. The following shows the output format defined with the ctime function.
Example: Thu Aug 18 17:54:15 JST 2005

Note on whitespace in the Include statement

The included filename may be the STIL filename added with the path information as required.
The absolute path and relative path are allowed for path information. Whitespace allowed in path and filename expressions depends on an operating system. The newline character (including tab) can not be used on almost operating systems, and blank can be used on parts of operating systems. Therefore, care must be taken in an applications including an operating system when whitespace is used in the Include statement.
To specify the path by the Include statement, care must be taken not to exceed 1024 characters. If necessary, use . (dot) to joint characters, not whitespace.

Note on whitespace in sigref_expr, time_expr, vec_data and serial_data

In sigref_expr, each of signal names (signal group names) and signal expression operators such as plus (+) and minus (-) becomes a single token, and whitespace is allowed between tokens. Like sigref_expr, whitespace is allowed between tokens in time_expr.
As is the case with *Note on whitespace in the WFC list* , vec_data and serial_data allow whitespace, and require whitespace as a separator in a character string exceeding 1024.

Companies participating in discussions on STIL 1450.0-1999 STIL Usage Guide

Companies participating in STIL-based Semiconductor Test Action Group regular committees in 2011 (in alphabetical order)
ADVANTEST CORPORATION
Fujitsu Semiconductor Limited
Panasonic Co., Ltd.
Renesas Electronics Corporation
Sony Corporation
TOSHIBA CORPORATION (TOSHIBA MICROELECTRONICS CORPORATION)
Yokogawa Test Solutions Corporation.
Companies participating in STIL-based Semiconductor Test Action Group regular committees in 2011 (in alphabetical order)
Companies participating in regular committees
ASTRON Inc.
ATE Service Coropration
Cadence Design Systems, Japan
Mentor Graphics Japan Co., Ltd.
MICRONICS JAPAN CO.,LTD
Nihon EVE K.K.
Nihon Synopsys, G.K.
SANWA TECHNO CO.,Ltd
Syswave Corporation.
Teradyne, Inc
Verigy (Japan) K.K.
Companies participating in STIL-based Semiconductor Test Action Group regular committees from fiscal 2005 to fiscal 2010 (in alphabetical order)
Fujitsu Semiconductor Limited
Panasonic Co., Ltd.
NEC Electronics Corporation*6
OKI SEMICONDUCTOR Co., Ltd. *4
Renesas Electronics Corporation*7
Renesas Technology Corporation*4
ROHM CO., LTD.*1
SEIKO EPSON CORPORATION*1
SHARP CORPORATION*5
Sony Corporation *1
TOSHIBA CORPORATION (TOSHIBA MICROELECTRONICS CORPORATION)
Companies participating in STIL-based Semiconductor Test Action Group regular committees from fiscal 2005 to fiscal 2010 (in alphabetical order)
Companies participating in regular committees
ADVANTEST CORPORATION

STIL Usage Guide【 Discussion member 】

Agilent Technologies International Japan, Ltd.*1
Cadence Design Systems, Japan
Credence Systems Corporation*1
Logic Vision Japan KK*5
Mentor Graphics Japan Co., Ltd.
Rorm/OKI SEMICONDUCTOR Co., Ltd. *7
SHARP CORPORATION*7
Nihon Synopsys, G.K.
Seiko Epson Corporation
Sony Corporation *2
Teradyne, Inc
Verigy (Japan) K.K. *3
Yokogawa Electric Corporation

*1 non-participation for fiscal 2006 to 2010
*2 non-participation for fiscal 2007 to 2010
*3 participation since fiscal 2008
*4 non-paricipation since fiscal 2009
*5 non-participation since fiscal 2010
*6 participation for fiscal 2005 to 2009
*7 participation since 2010

Comments submitted from special members

No	Title	Comment (Company name)

Related linked material

◆Problem for case-sensitive user-defined names

■IEEE Specification (Page55)

6.1 Case sensitivity

■Classification

Environment:

Indicates the contents that should be recognized to use mutually the STIL data in the test environment supporting STIL.

■Problem

STIL is a case-sensitive language. Case-sensitive user-defined names (especially for signal names) are anticipated to cause problems with the following reasons.

To distinguish names only by case-sensitiveness is prohibited except for comments in the RTL Design Style Guide provided by STARC. Formats such as VHDL and EDIF that don‘t distinguish upper case and lower case exist. A malfunction may occur when make the EDA tool read the STIL data.

■Proposed solution

Expressions that do not distinguish only by case-sensitiveness are recommended. (It is recommended not to use the same names of which the differences are only case-sensitiveness. That is, Abc, ABC and abc should not be used together.)

◆Problem of how to handle a concatenated character string

■IEEE Specification (Page58)

6.7 Character strings

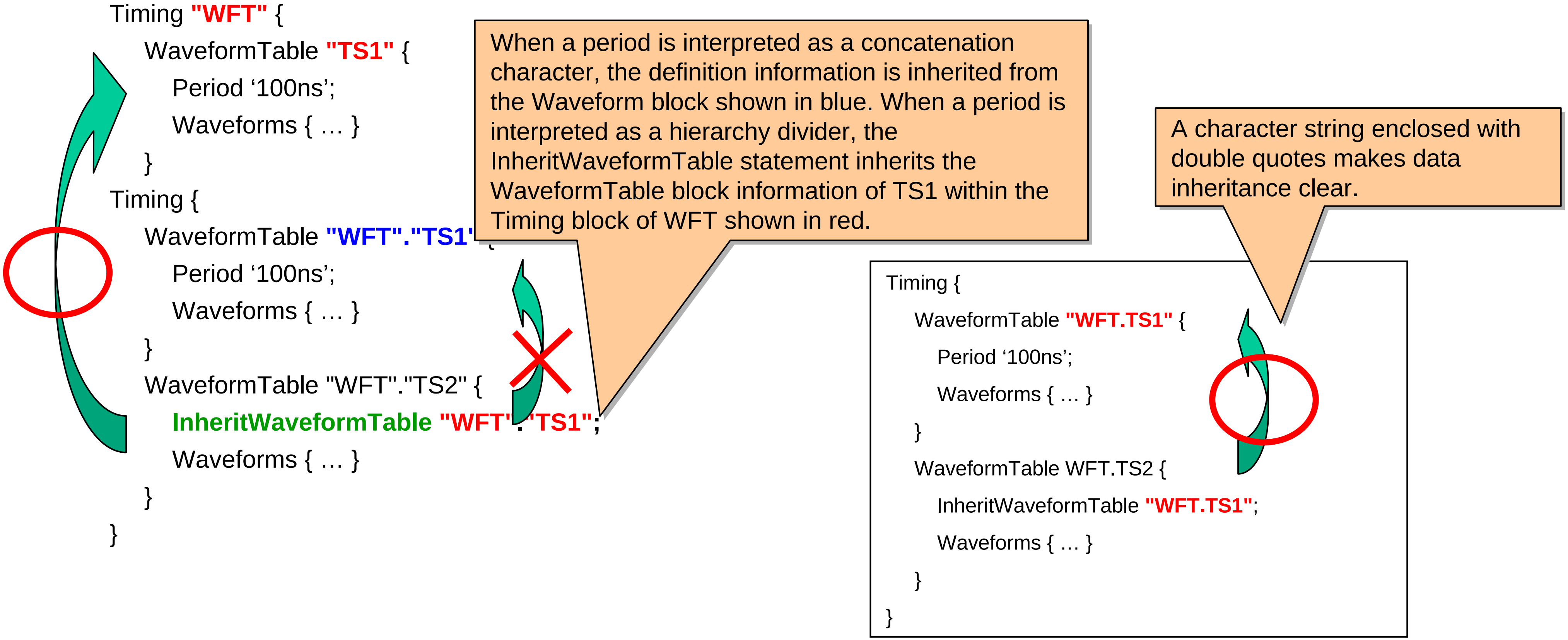
■Classification

Application:

Indicates the contents that present no problem in describing or interpreting STIL data, but should be recognized by the application tool and equipment which handle STIL data.

■Problem

A period is a special character provided as notation to connect two strings. Using this concatenation character notation, strings with over 1024 characters can be notated. However, for Inherit reference function in the Timing block, a period represents two different things: a hierarchy divider to delimit a hierarchical block, and a concatenation character. Therefore it may cause a problem because it is unclear which interpretation should be overridden.



```
    Timing "WFT" {
        WaveformTable "TS1" {
            Period ‘100ns’;
            Waveforms { … }
        }
    Timing {
        WaveformTable "WFT"."TS1" {
            Period ‘100ns’;
            Waveforms { … }
        }
        WaveformTable "WFT"."TS2" {
            InheritWaveformTable "WFT"."TS1";
            Waveforms { … }
        }
    }
```

When a period is interpreted as a concatenation character, the definition information is inherited from the Waveform block shown in blue. When a period is interpreted as a hierarchy divider, the InheritWaveformTable statement inherits the WaveformTable block information of TS1 within the Timing block of WFT shown in red.

A character string enclosed with double quotes makes data inheritance clear.

```
Timing {
    WaveformTable "WFT.TS1" {
        Period ‘100ns’;
        Waveforms { … }
    }
    WaveformTable WFT.TS2 {
        InheritWaveformTable "WFT.TS1";
        Waveforms { … }
    }
}
```

■Proposed solution

Interpretation as a hierarchy divider should be overridden. That is, a period shall be interpreted as a hierarchy divider in Inherit reference function in a Timing block and if there is no definition information to be referenced by this function, it is interpreted as a concatenation character.

It is recommended that an application issues the message indicating a period was interpreted as a hierarchy divider or a concatenation character in Inherit reference functions in a Timing block.

◆Problem of how to use STIL reserved words and characters

■IEEE Specification (Page59)

6.8 User-defined name characteristics

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

For user-defined names with reserved characters, the requirement of double quotes are not clear. Reserved characters include special characters, SI units and prefix characters for SI units (Refer to Table 2 on page 57 to 58 in IEEE Specification). It is unclear that when user-defined names themselves are reserved character, or includes reserved character(s), they should be enclosed by double quote or not.

* Many other expressions are presented in Figure 7 on page 14, Figure 9 on page 18, Figure 11 on page 23, and so on.

■Proposed solution

When a user-defined name itself is a reserved word or includes special character(s) (Special Characters and Whitespace Characters), it shall be enclosed in double quotes.

However reserved characters having multiple characters such as Cel and Ohm do not need to be enclosed by double quotes.
In order to avoid confusion, it is recommended that user-defined names should be enclosed by double quotes.   Except Wafeformchar references, It is also recommended not to use single-character-user-defined names or user-defined names themselves are a reserved word or a reserved characters.

Followings are the examples showing if the reserved words themselves or user defined names(Signal Name) can be used or not. The case-sensitive mixed name case and one letter User-defined name case are shown. However, those descriptions are simply for convenience purpose, therefore for the actual STIL description, it is recommended to be described with following the Solutions of “Problem for case-sensitive user-defined names” and “Problem of how to use STIL reserved words and characters” in STIL Usage Guide 1450.0.

Examples:
(1)User defined name of reserved word itself
```
  Signals { A In; a In; Alignment In; Ann In; } // not OK
  Signals { "A" In; "a" In; "Alignment" In; "Ann" In; } // OK
```
(2)User defined name including reserved word
```
  Signals { AA In; aa IN; Alignment In; XAnn In; } // OK
  Signals { "AA" In; "aa" IN; "XAlignment" In; "XAnn" In; } // OK (These expressions are recommended.)
```
(3)User defined name of Special Character itself
```
  Signals { ; In; * In; } // not OK
  Signals { ";" In; "*" In; } // OK
```
(4)User defined name including Special Character
```
  Signals { name; In; name* In; } // not OK
  Signals { "name;" In; "name*" In; } // OK
```

(5)User defined name for Engineering Prefix itself
Signals { E In; P In; T In; } // not OK
Signals { "E" In; "P" In; "T" In; } // OK
(6)User defined name including Engineering Prefix
Signals { EE In; PP In; TT In; } // OK
Signals { "EE" In; "PP" In; "TT" In; } // OK (These expressions are recommended.)
(7)User defined name for SI Unit itself of single character
Signals { A In; V In; F In; s In; } // not OK
Signals { "A" In; "V" In; "F" In; "s" In; } // OK
(8)User defined name including SI Unit itself of single character
Signals { AA In; VV In; FF In; ss In; } // OK
Signals { "AA" In; "VV" In; "FF" In; "ss" In; } // OK (These expressions are recommended.)
(9)User defined name for SI Unit itself of multiple character (Cel, Hz, Ohm)
Signals { Cel In; Hz In; Ohm In; } // OK
Signals { "Cel" In; "Hz" In; "Ohm" In; } // OK (As expressions, these are recommended but these are reserved characters themselves so should not be used anyway.)
(10)User defined name including SI Unit of multiple character (Cel, Hz, Ohm)
Signals { XCel In; XHz In; XOhm In; } // OK
Signals { "XCel" In; "XHz" In; "XOhm" In; } // OK (These expressions are recommended.)
(11)User defined name for min, max itself
Signals { min In; max In; } // OK
Signals { "min" In; "max" In; } // OK (As expressions, these are recommended but
these are reserved characters themselves so should not be used anyway.)
(12)User defined name including min, max
Signals { Xmin In; Xmax In; } // OK
Signals { "Xmin" In; "Xmax" In; } // OK (This expressions are recommended)

◆Problem of how to handle double-quoted user-defined names

■IEEE Specification (Page59)

Clause 6.8: User-defined name characteristics

Reference: IEEE STIL Clarifications (Page1)

■Classification

Application:

Indicates the contents that present no problem in describing or interpreting STIL data, but should be recognized by the application tool and equipment which handle STIL data.

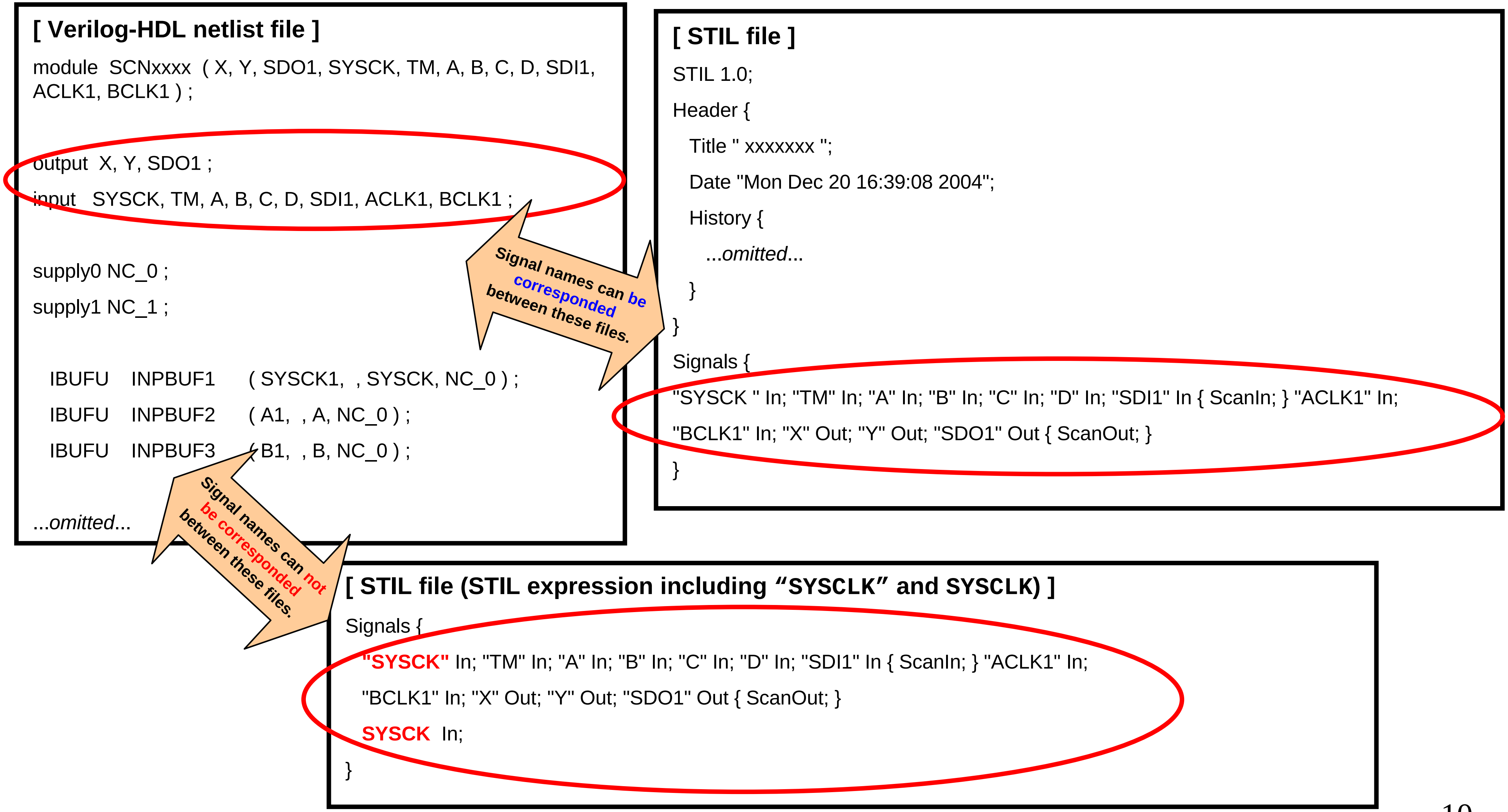
■Problem

Double quotes are not allowed as part of signal names in the IEEE Specification. The IEEE Clarification describes that both double-quoted signal name and unquoted signal name may be expressed in the Signals block.

If double quotes can not be used as part of signal names, double-quoted signal name and unquoted signal name indicate the same signal name, resulting in duplicated definition of the same signal name. Therefore, double-quoted and unquoted signal names must be distinguished in a single STIL file.

Though signal names are not enclosed in double quotes in the netlist file, in many cases signal names are enclosed in double quotes in the STIL file generated by the ATPG tool. When unquoted signal names as well as double-quoted signal names are included in the STIL file, an application can not determine which of signals to use.

Example expression: Problem for corresponding signals between the netlist file and STIL file



[Verilog-HDL netlist file]

```
module  SCNxxxx  ( X, Y, SDO1, SYSCK, TM, A, B, C, D, SDI1,
ACLK1, BCLK1 ) ;


output  X, Y, SDO1 ;

input   SYSCK, TM, A, B, C, D, SDI1, ACLK1, BCLK1 ;


supply0 NC_0 ;

supply1 NC_1 ;


  IBUFU    INPBUF1     ( SYSCK1,  , SYSCK, NC_0 ) ;

  IBUFU    INPBUF2     ( A1,  , A, NC_0 ) ;

  IBUFU    INPBUF3     ( B1,  , B, NC_0 ) ;


...omitted...
```

Signal names can be corresponded between these files.

[STIL file]

```
STIL 1.0;

Header {

  Title " xxxxxxx ";

  Date "Mon Dec 20 16:39:08 2004";

  History {

    ...omitted...

  }

}

Signals {

"SYSCK " In; "TM" In; "A" In; "B" In; "C" In; "D" In; "SDI1" In { ScanIn; } "ACLK1" In;

"BCLK1" In; "X" Out; "Y" Out; "SDO1" Out { ScanOut; }

}
```

Signal names can not be corresponded between these files.

[STIL file (STIL expression including "SYSCLK" and SYSCLK)]

```
Signals {

  "SYSCK" In; "TM" In; "A" In; "B" In; "C" In; "D" In; "SDI1" In { ScanIn; } "ACLK1" In;

  "BCLK1" In; "X" Out; "Y" Out; "SDO1" Out { ScanOut; }

  SYSCK  In;

}
```

■Proposed solution

The application which handles STIL expressions can process the same user-defined name which is double-quoted and unquoted (i.e., “Xyz” and XYZ used for the same type name such as signal name, signal group name and domain name) as an error for duplicated definition of the same user-defined name. This is because double-quoted and unquoted user-defined names might pose a high risk for usage errors through readability or misunderstanding of oral instructions even though the application can differentiate between them.

The application shall process user-defined names internally in a unified format which is either one of unquoted format or double-quoted format.

Example expression 1:

Expression containing double-quoted and unquoted Pattern block names

```
PatternBurst "BLOCK" {
        PatList { "PATTERN"; PATTERN;}
}
PatternExec { PatternBurst "BLOCK"; }
Pattern "PATTERN" {
        W TS1;
        V { RESET=0; CLK=0; Q[7..0]=XXXXXXXX; }
        V { RESET=1; CLK=P; Q[7..0]=LLLLLLLH; }
}
Pattern PATTERN {
        W TS1;
        V { RESET=0; CLK=0; Q[7..0]=XXXXXXXX; }
        V { RESET=1; CLK=P; Q[7..0]=LLLLLLLH; }
}
```

Example expression 2:

Expression containing double-quoted and unquoted label names

```
Pattern "_pattern_" {
  W "_default_WFT_";
  "precondition all Signals": C { "_pi"=¥r9 0 ; "_po"=XXX; }
  Macro "test_setup";
  "chain_test": Call "load_unload" {
    "SDI1"=¥r4 0011 ; "SDO1"=¥r7 X LLHHLLHHL; }
  pattern0: Call "load_unload" {
    "SDI1"=1010110; }
  Call "capture_SYSCK" {
    "_pi"=001100100; "_po"=HLH; }
  "apply 1": Call "master_observe";
  "pattern0": Call "load_unload" {
    "SDO1"=LLHHLLH; "SDI1"=0011011; }
  Call "capture_SYSCK" {
    "_pi"=001001000; "_po"=HHL; }
  "apply 2": Call master_observe;
  "pattern 1": Call "load_unload" {
    "SDO1"=LLHHHHL; "SDI1"=0001001; }
```

It is recommended to differentiate user-defined names by adding the numbers and characters, and not to differentiate them by double-quoted and unquoted names.

It is recommended that the application generating the STIL file outputs double-quoted user-defined names.

For bus signal, “BUS”[0] expression is recommended rather than “BUS[0]” expression with double quotes enclosing the whole bus expression. “BUS”[0..7] can be interpreted as a bus signal expression, whereas “BUS[0..7]” can not be determined as a bus signal expression since it may be interpreted as a character string containing special characters.

◆Problem for newline and tab characters included in user-defined names

■IEEE Specification (Page59)

6.8 User-defined name characteristics

■Classification

Environment:

Indicates the contents that should be recognized to use mutually the STIL data in the test environment supporting STIL.

■Background of problem

For user-defined names in examples in a IEEE Specification, expressions using blank characters exist, but expressions using tab and newline characters do not exist.

A tab character, a newline character and a blank character can not be differentiated in a hard copy respectively. So in the case that the STIL file containing tab, newline or blank characters in user-defined names is transferred, it need to be clarify how to describe user-defined names using tab, newline or blank characters.

```
PatternBurst "PB
▨BLOCK" {
        PatList { BLOCK; }
}
PatternExec {
        PatternBurst "PB
▨BLOCK";
}
Pattern BLOCK {
        W TS1;
        V { RESET=0; CLK=0; Q[7..0]=XXXXXXXX; }
        V { RESET=1; CLK=P; Q[7..0]=LLLLLLLH; }
}
```

Differentiation of whitespace shown with ▨

■Proposed solution

When a user-defined name containing whitespace (blank, tab or newline character) is described, it is difficult for the STIL creator or user to check visually how blank, tab and newline character are used in expressions. If the STIL expression is modified, it is not recognized, which increases the likelihood of problems. Therefore, we should set the minimum requirements to prevent problems occur during a visual check . As a result, user-defined names containing blank may be used, but user-defined names containing tab and newline characters shall not be used.

It is recommended that an application issues the message indicating a period was interpreted as a hierarchy divider or a concatenation character in Inherit reference functions in a Timing block.

◆Problem of whether the Include statement can be nested

■IEEE Specification (Page71)

10. Include statement

■Classification

Description:

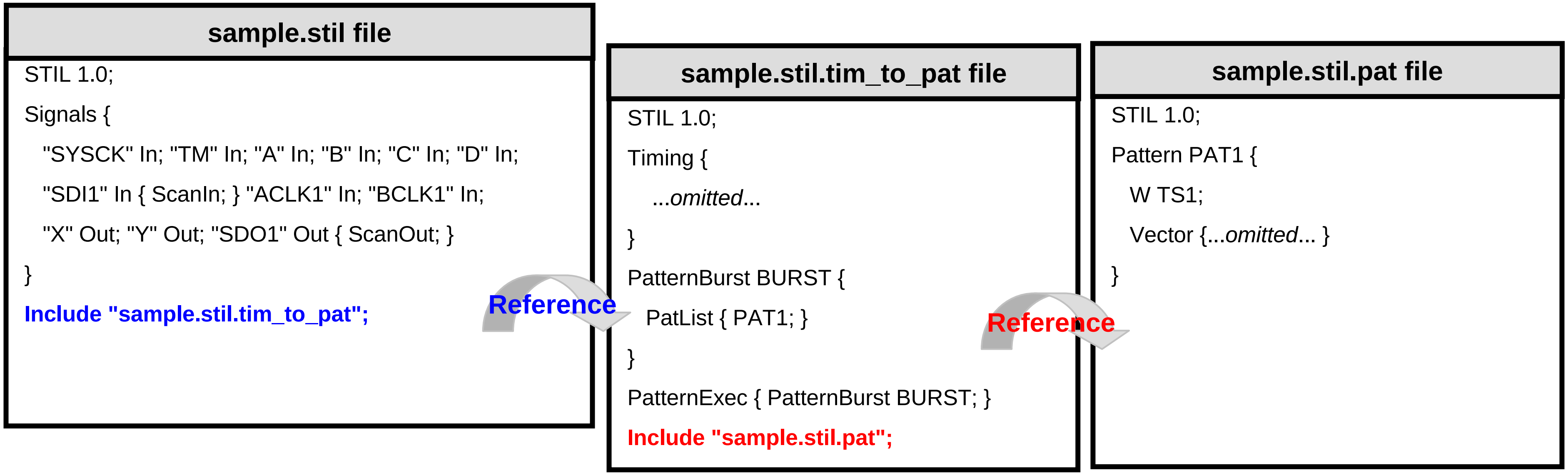
Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

In the STIL file referenced with the Include statement, when there is a nesting description of Include statement which references from another STIL file by using an Include statement, it is unknown how to handle it because the validation of the description is not clearly stated in the specification. That is, in STIL file “B” referenced by Include statement from STIL file “A”, when there is a nesting description which references STIL file “C” by an Include statement, the validation of using the Include statement is not clearly stated in the specification.

■Proposed solution

The nesting of the Include statement shall be allowed and the number of nesting shall be unlimited.



sample.stil file

```
STIL 1.0;
Signals {
   "SYSCK" In; "TM" In; "A" In; "B" In; "C" In; "D" In;
   "SDI1" In { ScanIn; } "ACLK1" In; "BCLK1" In;
   "X" Out; "Y" Out; "SDO1" Out { ScanOut; }
}
Include "sample.stil.tim_to_pat";
```

sample.stil.tim_to_pat file

```
STIL 1.0;
Timing {
    ...omitted...
}
PatternBurst BURST {
   PatList { PAT1; }
}
PatternExec { PatternBurst BURST; }
Include "sample.stil.pat";
```

sample.stil.pat file

```
STIL 1.0;
Pattern PAT1 {
   W TS1;
   Vector {...omitted... }
}
```

■Recommended handling

It is recommended that nesting expressions with the Include statement resulting an endless loop are regarded as an error, and the error message is issued by the application which handles the STIL file. That is, in STIL file (B) referred by the Include statement from STIL file (A), when nesting expressions referring STIL file (A) back by the Include statement exist, it is an endless loop so the application should output an error message.

◆Problem of how to define the SignalGroups block

■IEEE Specification (Page77)

15. SignalGroups block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

Pin group defined in the SignalGroups block can define 0 or more pins as a group but the specification of how to define 0 pin is not clear. Also the interpretation when there was the STIL expression referring a 0 pin group was not clear. For example, when patterns are defined to a 0 pin group in the Patten block or timing is defined to a 0 pin group in the Timing block, the interpretation is unknown.

■Proposed solution

When defining a pin group name, if the number of pin zero, the following two expressions are allowed, but Type 1 shall be used.

(Type 1) `GroupName = '';`

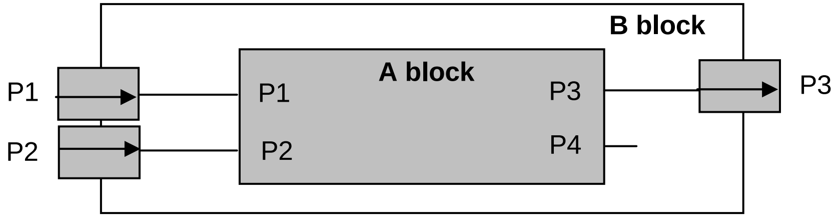
(Type 2) `GroupName = ;`

If there is the STIL expression referring a 0 pin group, it shall be interpreted that there is no expression corresponding to a pin group name as a result of the reference. For example when patterns are defined to a 0 pin group in the Patten block, it is interpreted as no pattern definition to the pin group name. Also when timing is defined to a 0 pin group, it is interpreted as no timing definition to the pin group name. It is recommended to define signals by using the Pseudo pin as much as possible in order to make missing signals clear, except when you need to define a 0 pin group. Except when pin group definition using the same name of the pin is required, it is recommended not to use the same name to the group as much as possible.

[Note] If the pattern is defined to the 0 pin group name, interpret as the following:

V { Pin1=0; Pin2=L; GRP_Empty=000; } //GRP_Empty contains no signal name.
⇕ Both are interpreted as an equivalent expression.
V { Pin1=0; Pin2=L; }

[When using the 0 pin group.]



STIL file in A block

(File name: AFUNC.stil)

```
STIL 1.0;
Signals {
  P1 In; P2 In; P3 Out; P4 Out;
}
SignalGroups {
  ALL='P1+P2+P3+P4';
}
  …The timing information definition is omitted…
PatternBurst A_TEST {
  PatList  { A_PAT1; }
}
PatternExec {
  PatternBurst A_TEST;
}
Pattern A_PAT1 {
  W TS1;
  C { ALL=00XX; }
  V { P1=1;P2=0;P3=L;P4=H; }
  …The pattern description hereafter is omitted…
}
```

When using a test pattern (STIL file) of the block built into IP as it is, the 0 pin group can be used for unused pins.

Example 1

The test pattern description of B block using the test pattern of A block as it is.

```
STIL 1.0;
Signals {
  P1 In; P2 In; P3 Out; // P4's definition removed
  …The definition of other pin information is omitted…
}
SignalGroups B_block {
  P4 = '';
  ALL= 'P1+P2+P3+P4';
}
  … The timing information definition is omitted…
PatternBurst B_TEST {
  SignalGroups B_block;
  PatList  { A_PAT1; }
}
PatternExec {
  PatternBurst B_TEST;
}
Include "AFUNC.stil" IfNeed Pattern;
```

Using the test pattern of A block in the Include statement. In the Pattern block of STIL in B block, consequently "The pattern removal for pin P4" has executed.

Both have the same test items.

Example 2

The test pattern description of B block is recreated based on the test pattern of A block

```
STIL 1.0;
Signals {
  P1 In; P2 In; P3 Out; ;

  … The definition of other pin information is omitted   …
}
SignalGroups {
  ALL= 'P1+P2+P3';
}
   … The timing information definition is omit
  PatternBurst B_TEST {
  PatList  {B_PAT1; }
}
PatternExec {
  PatternBurst B_TEST;
}
Pattern B_PAT1 {
  W TS1;
  C { ALL=00X; }
  V { P1=1;P2=0;P3=L; }
   … The pattern description hereafter is omitted…
}
```

[Recommendation] Do not use the 0 pin group and define the P4 as the Pseudo pin.

Example 3

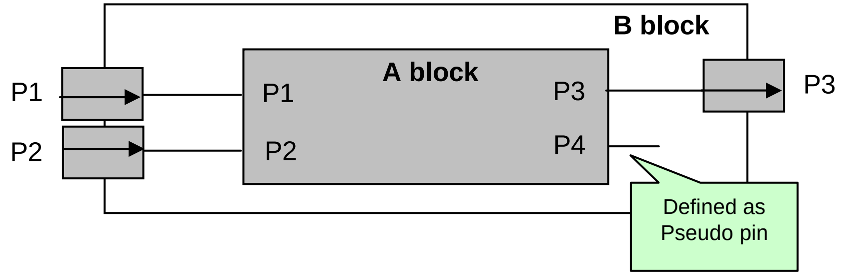
```
STIL 1.0;
Signals {
  P1 In; P2 In; P3 Out; P4 Pseudo;

  … The definition of other pin information is omitted  …
}
SignalGroups {
  ALL= 'P1+P2+P3+P4';
}

  … The timing information definition is omitted  …
PatternBurst B_TEST {
  PatList  { A_PAT1; }
}
PatternExec {
  PatternBurst B_TEST;
}
Include “AFUNC.stil” IfNeed Pattern;
```

In order to use the test pattern of A block as it is, P4, which does not go outside, is regarded as Pseudo.

Example 1 on the previous page uses “a 0 pin group” in order to utilize the STIL of test pattern in A block. STIL Usage Guide recommends handling such pin by Pseudo to clarify that the definition “P4=‘ ’; for a 0 pin group does not represent a missing pin name. If we compare Example 3, which handles the pin by Pseudo, to Example 1, the addition and change in the description (in red) should be less than Example 1 which uses a 0 pin group. However, the interpretation of the pattern for the Pseudo pin depends on the application so it should be carefully used.



Defined as Pseudo pin

◆Problems for multiple PatternExec blocks

■IEEE Specification (Page80)

16. PatternExec block

■Classification

Application:

Indicates the contents that present no problem in describing or interpreting STIL data, but should be recognized by the application tool and equipment which handle STIL data.

■Problem

In the STIL file which multiple PatternExec blocks are defined, since there is no rule of the order for processing multiple PatternExec blocks, it is unknown how to handle them.

■Example Expression containing multiple PatternExec blocks

```
PatternBurst "BURSTBLK1" {
   PatList { PAT1; PAT2; }
}
PatternBurst "BURSTBLK2" {
   PatList { PAT3; }
}
PatternExec "ONE" {
   PatternBurst "BURSTBLK1";
}
PatternExec "TWO" {
   PatternBurst "BURSTBLK2";
}
```

■Proposed solution

In the STIL file which multiple PatternExec blocks are defined, the multiple PatterExec blocks shall be understood that all the PatternExec blocks will be executed by executing each of the multiple PatternExec blocks in the order of description.

Specific handling of these blocks performed by an application shall not be defined, and handling these blocks shall be dependent on individual application.

◆Problem of how to interpret multiple events specified to the same time

■IEEE Specification (Page86)

18.1 Timing and WaveformTable syntax

■Classification

Description:

Indicates the contents that present no problem in describing or interpreting STIL data, but should be recognized by the application tool and equipment which handle STIL data.

■Problem

It is unknown how to interpret multiple events defined to the same event time for one WFC. Also it is unknown whether clearly specified ForceOff definition is required to switch to the output mode. For example, how to interpret the following definitions is unknown.

Example expression indicating the problem: Interpretation of the underscored expression

```
Timing {
   WaveformTable "TS1" {
      Period '300ns';
      Waveforms {
         ABUS { 01 { '0ns' D/U; }}
         BBUS { 01 { '0ns' D/U; }}
         ABUS { HLZX { '0ns' Z; '0ns' X; '250ns' H/L/T/X;}}
         BBUS { HLZX { '0ns' Z; '0ns' X; '250ns' H/L/T/X;}}
      }
   }
```

This type of expression depends on a tester behavior, and requires to be described consciously. Investigations of this expression will be performed for the tester manufacturers and obtained results are to be referenced for discussion.

■Proposed solution

In the case of multiple events specified to the same event time, an input and an output events may be specified to the same time respectively.

When WFC is defined with multiple events specified to the same event time, it shall be understood that these events are executed in the order of the expression. When multiple input events or output events are defined to the same event time, basically it is interpreted as an error and the application outputs an error message. An input and output events shall be handled to be unaffected mutually. Therefore the exclusive operation that the driver is turned off as turning the output mode on will not be performed. If the driver needs to be turned off on a tester, the event must be clearly defined with ForceOff(Z).

It is up to each application how to continue the process after the output of the error message, such as performing in the order of expressions, or as a result, taking the latter expression. It is recommended to output a message about the performance.

It is recommended that an application on a terser issues a warning or error message if production test presents behaviors different from the STIL expression on a device testing expected.

◆Problem for the Marker event expression

■IEEE Specification (Page88 Table11)

■Classification

Application:

Indicates the contents that present no problem in describing or interpreting STIL data, but should be recognized by the application tool and equipment which handle STIL data.

■Problem

It is unknown how to handle the Marker event expression. Since there is no example expression in the IEEE Specification, it is unknown which case is assumed to require this expression.

■Proposed solution

Since the Marker event is not an event related to test behavior, the typical application ignores it.

For the application which does not require to handle the Maker event information, it is recommended to issue a message indicating that the Maker event definition in the STIL file was ignored.

◆Problem when Meas is selected by the Spec variable

■IEEE Specification (Page94)

19. Spec and Selector blocks

■Classification

Application:

Indicates the contents that present no problem in describing or interpreting STIL data, but should be recognized by the application tool and equipment which handle STIL data.

■**Problem**

When Meas is selected, the timing value is set as it executes Meas, so the timing value is not clearly set until the actual measurement of the device on the tester is executed. Therefore, for some applications how to handle timing values is important. For example, when a simulation runs using the STIL expressions with Meas selected, the execution timing can not be set.

■Example expression: [Example expression containing the variable name oe_clk_edge specified by Meas]
The rising edge timing of the enable signal is specified by `Meas`.

```
STIL 1.0;
Signals {
  DIR In;
  OE In;
  A[7..0] In; B[7..0] Out;
}
SignalGroups {
  ABUS='A[7..0]';
  BBUS='B[7..0]';
  ALL ='DIR + OE + ABUS + BBUS';
}
Selector "measure" {
  "oe_clk_edge" Meas;
}
Timing "tester timing" {
  WaveformTable TS1 {
    Period '500ns';
    Waveforms {
      DIR { 01 { '0ns' D/U; } }
      OE { 01 { '0ns' U; '"oe_clk_edge"' D/U; '350ns' U; } }
      ABUS { 01 { '10ns' D/U; } }
      BBUS { HLZ { '0ns' Z; '0ns' X; '250ns' h/l/t; '300ns' X;} }
    }
  }
}
```

continued on right ⟶

```
PatternBurst "tester burst" {
  PatList { "tester pattern"; {
}
PatternExec {
  Timing "tester timing";
  Selector "measure";
  PatternBurst "tester burst";
}
Pattern "tester pattern" {
  W TS1;
  V { ALL=0000000000LLLLLLLL; }
  V { ALL=0010000000HLLLLLLL; }
  V { ALL=0001000000LHLLLLLL; }
  V { ALL=0000100000LLHLLLLL; }
  V { ALL=0000010000LLLHLLLL; }
  V { ALL=0000001000LLLLHLLL; }
  V { ALL=0000000100LLLLLHLL; }
  V { ALL=0000000010LLLLLLHL; }
  V { ALL=0000000001LLLLLLLH; }
}
```

■Proposed solution

For STIL expressions containing an execution timing set by Meas, the same handling is not necessarily required for applications which can operate and can not operate in accordance with STIL expressions. An individual application shall be responsible for handling an execution timing set by Meas.

It is recommended that application output a message how to handle an execution timing set by Meas.

◆Problem with multiple definitions of Pattern

■IEEE Specification (Page103)

22.1 Vector (V) statement

■Classification

Application:

Indicates the contents that present no problem in describing or interpreting STIL data, but should be recognized by the application tool and equipment which handle STIL data.

■Problem

The interpretation is not clear when different multiple patterns are specified to the same pin in one address. Also as for the pattern description of one address to a bidirectional pin, the description is not clear when the pattern (WFC) for input and the pattern (WFC) for output are specified separately. As a result, a simulation error occurs due to the difference in interpretation.

■Proposed solution

In a Vector or a Condition statement, the multiple definitions of pattern for the same pin in one address is not allowed. That is, in a Vector or Condition statement, specifying patterns (WFC) to each pin is allowed at least once. Also for a bidirectional pin, when a pattern description of one address is specified to a pattern for input and a pattern for output separately, it is interpreted as an error so that the application output an error message.

It is up to each application how to continue the process after the output of the error message, such as taking only the latter pattern (WFC) definition. It is recommended to output a message about the performance.

Example 1: Expression splitting a bidirectional pin into input and output within a single Vector block

```
Signals { "IN1" In; "OUT1" Out; "IO1" InOut; }
SignalGroups { "_pi" = ' "IN1"+"IO1" '; "_po" = ' "OUT1"+"IO1" ' }
…
Procedures {
  "proc" {
      W "_default_WFT_";
      V { "_pi"=01; "_po"=XX; }
    …
  }
}
```

STIL expression to cause a problem

The pattern 1 and X are assigned to a bidirectional signal IO1, and the input state 1 is generated by the ATPG.

Corrected expression

When multiple patterns are assigned to the same signal, the latter pattern precedes the former pattern in STIL 1450.0. Therefore, IO1 becomes X. For the correct expression, edit the expression as shown below.

V { "_pi"=01; "_po"=XX; "IO1"=1; }

Example 2: Expression splitting a bidirectional pin into input and output within multiple Vector blocks

```
Signals { "CLK" In; "IN1" In; "OUT1" Out; "IO1" InOut; }
SignalGroups { "_pi" = ' "IN1"+"IO1" '; "_po" = ' "OUT1"+"IO1" ' }
…
Procedures {
  "proc" {
      W "_default_WFT_";
      V { "_pi"=01; }
      C { "_po"=XX; }
      V { "CLK"=P; }
      …
  }
}
```

STIL expression to cause a problem

The Condition statement defines patterns to _po. It is understood that the pattern 1 assigned to _pi in the Vector statement is valid in the following Vector statement. Therefore, the input state 1 is generated for IO1 by the ATPG.

Corrected expression

Same as Example 1. For the correct expression, edit the expression as shown below.

V { "CLK"=P; "IO1"=1; }

◆Problem for timing change and pattern omission in the Loop block

■IEEE Specification (Page106)

22.6 Loop statement

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

When the pattern expressions are omitted in the first Vector statement within the Loop block, or timing is changed within the Loop block, it is unknown which test patterns are applied.

```
Pattern PAT1 {
  W TS1;
  V { P1=0; P2=0; P3=X; }
  Loop 10 {
    V { }
    W TS2;
    V { P1=1; P2=1; P3=H; }
  }
}
```

■Proposed solution

When patterns are omitted in the first Vector statement within the Loop block, that means no patterns are assigned to all signals, the pattern (WFC) described preceding the Loop block shall be taken. It shall be understood that the same test patterns in the Loop block are run by the first and later loop execution.

When timing is not defined on the top in the Loop block, and timing is changed in the Loop block, it shall be understood that the timing information used with the first Vector statement is the same as one used in the Vector statement preceding the Loop block. This timing information shall be regarded as timing used in the first test pattern (Vector statement) within the Loop block, and referenced by first and later loop execution.

[Recommended STIL expression]

●It is recommended not to omit patterns at the top of the Loop block.

●At the beginning of the Loop block, it is recommended that not to omit timing specification. Especially when switching the timing in the middle of the pattern of Loop block, it is recommended to specify timing to explicitly specify the timing of the beginning pattern of the Loop block.

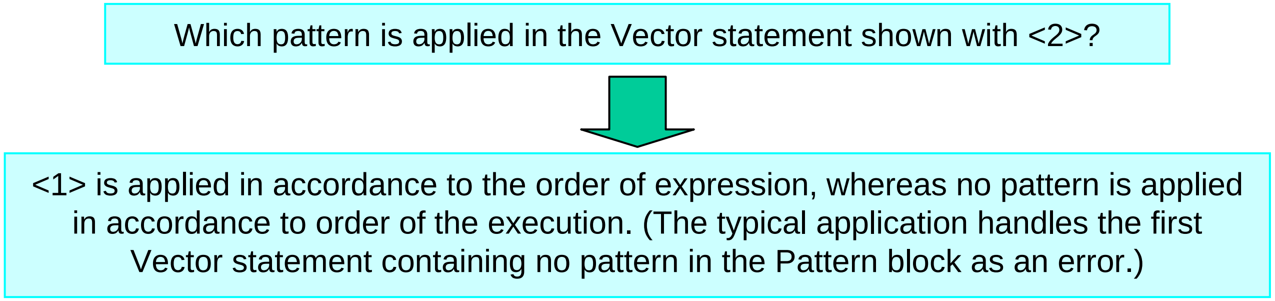
```
Pattern PAT1 {
  W TS1;
  V { P1=0; P2=0; P3=X; }
  Loop 10 {
    W TS1;
    V {P1=0; P2=0; P3=X; }
    W TS2;
    V { P1=1; P2=1; P3=H; }
  }
}
```

Supplemental remarks: Goto and Start statements to control pattern execution except the Loop statement

Not only the Loop statement but also the Goto and Start statements are affected by the order of execution. For the patterns and timings omitted for the signals in (1) and (2), there is no problem when patterns and timings are applied in the order of expression. If the order of execution is used to apply patterns and timings, no pattern is defined in the Pattern block since the Goto or Start statement branches to a vector label. Therefore, it will cause no trouble if Goto and Start statements as well as Loop statement shall be interpreted in the order of the description.

(1)Goto Statement

```
Pattern PATT1 {
     W TS1;
     Goto JUMP;
      V { ALL=11111LLLLL; } // <1>
      JUMP: V {} // <2>
}
```

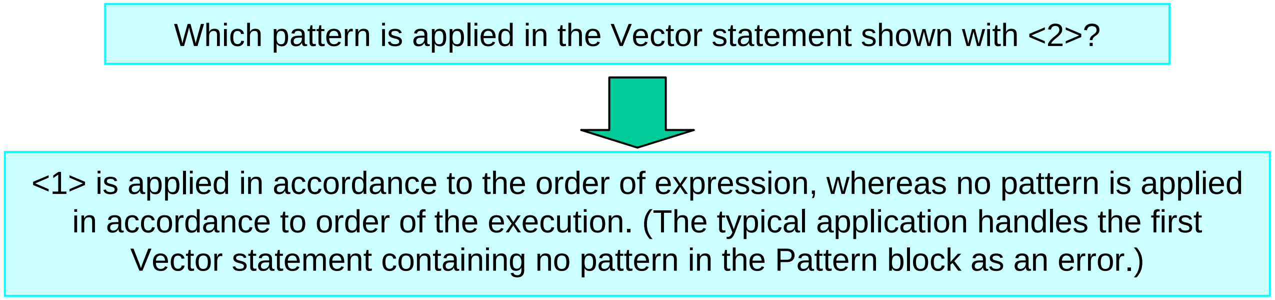


Which pattern is applied in the Vector statement shown with <2>?

<1> is applied in accordance to the order of expression, whereas no pattern is applied in accordance to order of the execution. (The typical application handles the first Vector statement containing no pattern in the Pattern block as an error.)

(2) Start statement

```
 PatternBurst {
     Start FIRST;
     PatList { PATT1; }
 }
Pattern PATT1 {
     W TS1;
     V { ALL=00000XXXXX; } // <1>
     FIRST: V {} // <2>
     V { ALL=11111LLLLL; }
 }
```



Which pattern is applied in the Vector statement shown with <2>?

<1> is applied in accordance to the order of expression, whereas no pattern is applied in accordance to order of the execution. (The typical application handles the first Vector statement containing no pattern in the Pattern block as an error.)

Recommended expression including the Goto and Start statements

As is the case with the Loop block, it is recommended to describe patterns and timings without omitting them in the branch destination of the Goto or Start statement.

◆Problem for the ScanChain statement

■IEEE Specification (Page108)

22.12 ScanChain statement

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

The IEEE Specification does not define clearly how to make multiple ScanChain blocks active.

■Proposed solution

When the ScanChain statement is not described in the Pattern block (or in the Procedures and MacroDefs blocks), all scan chains shall be interpreted as being active. When the ScanChain statement is described, only the scan chain specified in the ScanChain statement is active. It shall be understood that the ScanChain statement is repeated to make multiple scan chains active.

Since the statement to make scan chains inactive is not provided, it shall be understood that the active scan chains become inactive by executing only one shift operation to given scan chains after the ScanChain statement. To make the scan chains active again, the ScanChain statement needs to be described.

■**Supplemental remarks:**

IEEE Specification (Page108)

- Syntax is defined as shown below.

(LABEL **:**) **ScanChain** CHAINNAME **;**

The IEEE Specification allows a single scan chain name, and it is unknown how to specify multiple scan chain names.

Only `CHAIN1` is active for this shift operation.

Only `CHAIN2` is active for this shift operation.

Example expression containing the ScanChain statements

```
MacroDefs {
    unload_load {
         W scan;
         C { all=100 001 00 XX 0000 XXXX; }
         Shift {
               V {si1=#; si2=#; so1=#; so2=#;}
         }
    } // unload_load
}

     ...omitted...

Pattern scan_loads {
   W wft_base;
   V { all = 110 111 00 XX 0000 XXXX;}
   ScanChain "CHAIN1";
   ScanChain "CHAIN2";
   Macro unload_load {
          si1=1101110101;  si2=0000111011;
   }
   V { pis = 1010;} // FORCE PIs
   V { pos = HLHL;} // MEASURE POs
   V { capture_clks = 00; pos = XXXX;}
   ScanChain "CHAIN1";
   Macro unload_load {
          so1=HLHHHLHLHL;
   }
   V { all = 111 111 11 XX 0000 XXXX;}
   ScanChain "CHAIN2";
   Macro unload_load {
          so2=HHHLLHHLHL;
   }
}
```

Supplemental remarks: When the scan chain defined in the ScanChain statement differs from the active scan chain

The ScanChain statement is used for the user only to know that the specified scan chain is active. Therefore, it is not determined in the STIL data whether the scan chain to be called by the Macro or Call statement is in active use.

When the scan chain to which the shift operation of a set of patterns executed after the ScanChain statement differs from the scan chain which has declared as active by ScanChain statement, and that difference can be identified in the application side, whether to continue the process after outputting an error or warning depends on each application.

In order to make multiple scan chains active, users can describe it simply by using ActiveScanChains (Clause 16) and ScanChainGroups (Clause 14) in STIL1450.1, the STIL extended specification to design environment.

◆Problem for the state of pins undefined in the Procedures block

■IEEE Specification (Page109 Table13)

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

When not all pins are described in Procedures, how to handle the pins not described is unknown.

■Proposed solution

For the pins not described, use the values when defined in the Default State statement, and if not defined in the DefaultState statement, the default values of the DefaultState statement as the pins' values. The default value of the DefaultState statement for input shall be Z and the default value of the DefaultState statement for output without clear specification shall be X.

DefaultState will be invalid if patterns (WFC) were described just once in the Condition statement or the Vector statement somewhere in the Pattern block.. That is, it is not defined by the DefaultState statement, however the Procedures block is independent so DefaultState is always valid in the first pattern.

As the same as the MacroDefs block and the Pattern block, at the beginning of the Procedures block, it is recommended that the state of all pins used in the Pattern block, except the pins to which the DefaultState statement is valid shall be described in the Condition statement or the Vector statement.

Example

```
Signals {
  Pin1 In { Default State D; }
  Pin2 In;
  Pin3 Out;
  si1 In;
  SYSCLK In;
}
…omitted…
Procedures {
     unload_load {
           W TS2;
           Shift {
                  V {si1=#;  SYSCLK=P; }
           }
     }
}
…omitted…
Pattern "SCAN" {
    W TS1;
    V { all = 110 111 00 XX 0000 XXXX;}
    Call unload_load {
            si1=1001110101;
    }
   …omitted…
```

Pattern block

Vector Statement in Procedures Block

V{ si1=1; SYSCLK=P; }   //Pin1; Pin2; Pin3;

There are some undefined pins in the Procedures block. How the values to these pins are interpreted is important.

In the proposed solution, the DefautState values are used for the pins which Default State are defined, and for other pins, the default values of the DefautState statement (input: Z, output: X) are used.

Pattern block

Vector statement in Procedures

block

Pins not defined in the Procedures block will be the following values.

V{ si1=1; SYSCLK=P; @0 {Pin1=D; Pin2=Z; Pin3=X;} }

Pin1 becomes the Default State value D in the DefaultState statement. Pin2 becomes the default value Z in the DefaultState statement. Pin3 becomes the default value X in the DefaultState statement.

◆Problem for the # operator for pattern substitution

■IEEE Specification (Page112)

24.5 Procedure and Macro Data substitution

■Classification

Description:

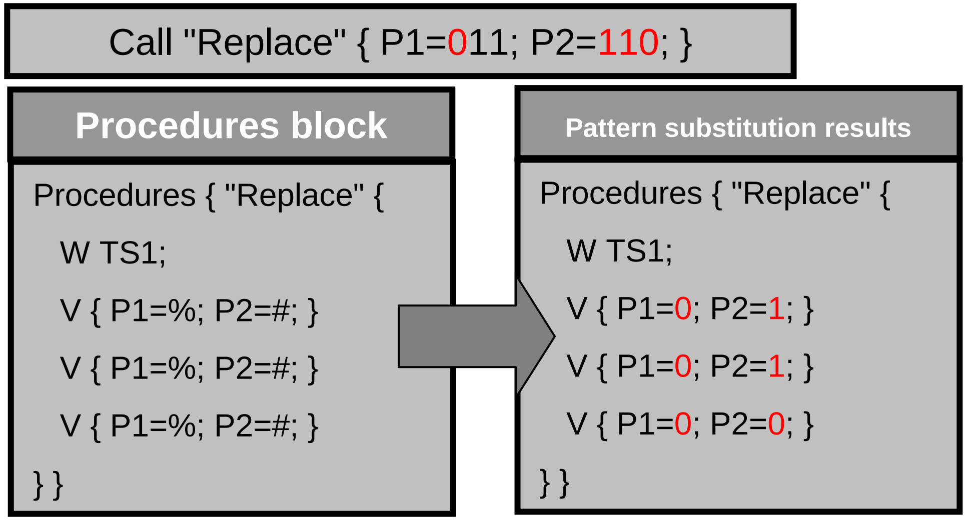
Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

When substituting for pattern, the # operator allows the substitution to the scanpattern, or to the Vector statement in the MacroDefs block, or the Vector statement in the Procedures block. But the substitution to the latter Vector statement is not clearly specified in the standard.

■Proposed solution

It shall be understood that to # in the Vector Statement in the MacroDefs block or # in the Vector statement in the Procedures block, parameter data (patterns) in the Macro statement or in the Call statement is substituted to the # according to the order of the Alignment statement's definition.
(cf. In the case of the % operator substitution, to % in the Vector statement in the MacroDefs block , or in the Vector statement in the Procedures block, only one of parameter data (patterns) specified in the Macro statement or the Call statement is substituted to the % according to the Alignment statement definition.)



```
Call "Replace" { P1=011; P2=110; }
```

Procedures block

```
Procedures { "Replace" {
  W TS1;
  V { P1=%; P2=#; }
  V { P1=%; P2=#; }
  V { P1=%; P2=#; }
} }
```

Pattern substitution results

```
Procedures { "Replace" {
  W TS1;
  V { P1=0; P2=1; }
  V { P1=0; P2=1; }
  V { P1=0; P2=0; }
} }
```

% defines fixed pattern substitution.

# defines incremental pattern substitution.

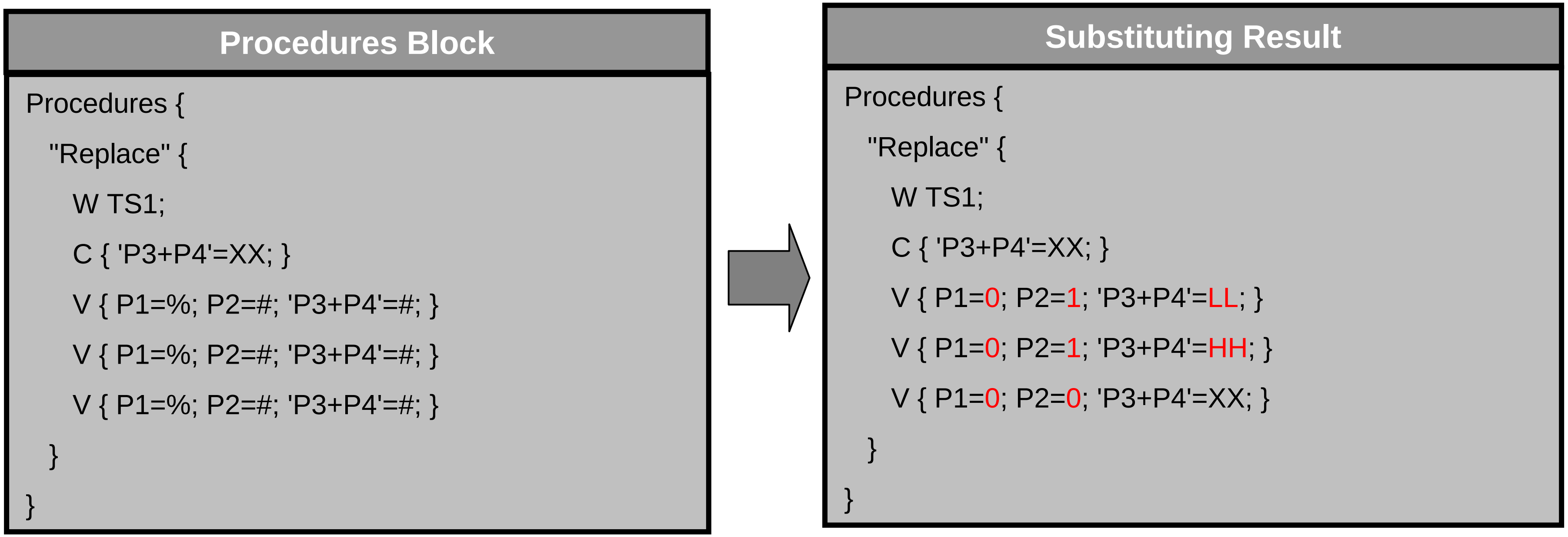
When the parameter data (patterns) described in the Macro statement or the Call statement is defined to a pin or a whole set of the pins in a pin group,
the substitution by the Vector statements with the mixing of # and % operators shall not be allowed to the pin or the pin group.

When the parameter data (patterns) described in the Macro statement or the Call statement is defined to a pin group pin-wise, the substitution by the Vector statements without the mixing of # and % operators through the Vector statements to any pin in the pin group shall be allowed.

■Proposed Solution (Supplementation):

Substituting for Pin Group is as the followings

```
Call "Replace" { P1=011; P2=110; ‘P3+P4'=LLHH;}
```



Procedures Block

```
Procedures {
  "Replace" {
    W TS1;
    C { 'P3+P4'=XX; }
    V { P1=%; P2=#; 'P3+P4'=#; }
    V { P1=%; P2=#; 'P3+P4'=#; }
    V { P1=%; P2=#; 'P3+P4'=#; }
  }
}
```

Substituting Result

```
Procedures {
  "Replace" {
    W TS1;
    C { 'P3+P4'=XX; }
    V { P1=0; P2=1; 'P3+P4'=LL; }
    V { P1=0; P2=1; 'P3+P4'=HH; }
    V { P1=0; P2=0; 'P3+P4'=XX; }
  }
}
```

Example substituting # & % for each pin

```
Call “Replace” { P1=011; P2=110;}
```

Substituting # (or %) for both pins is OK

```
Procedures {
  "Replace" {
    W TS1;
    V { P1=#; P2=#; }
    V { P1=#; P2=#; }
    V { P1=#; P2=#; }
  }
}
```

Substituting # for one pin, % for another pin is OK

```
Procedures {
  "Replace" {
    W TS1;
    V { P1=%; P2=#; }
    V { P1=%; P2=#; }
    V { P1=%; P2=#; }
  }
}
```

Substituting both # and % for one pin is NG

```
Procedures {
  "Replace" {
    W TS1;
    V { P1=%; P2=#; }
    V { P1=#; P2=%; }
    V { P1=%; P2=#; }
  }
}
```

Mixing # and % to one pin is NG

Examples of substituting # and % for the pin group defined in the SignalGroups

```
SignalGroups {"Pin" = 'P1 + P2';}
```

```
Call “Replace” { "Pin"=011110;}
```

Because the Call statement is defined in the pin group, it would be the mixing of # %, so the result is NG

Substituting # for both pins (or %) is OK

```
Procedures {
  "Replace" {
    W TS1;
    V { P1=#; P2=#; }
    V { P1=#; P2=#; }
    V { P1=#; P2=#; }
  }
}
```

Substituting # (or %) in the pin group notation is OK

```
Procedures {
  "Replace" {
    W TS1;
    V { Pin=##; }
    V { Pin=##; }
    V { Pin=##; }
  }
}
```

Substituting # for one pin and % for another pin is NG

```
Procedures {
  "Replace" {
    W TS1;
    V { P1=%; P2=#; }
    V { P1=%; P2=#; }
    V { P1=%; P2=#; }
  }
}
```

Substituting # for one pin and % for another pin in the pin group notation is NG

```
Procedures {
  "Replace" {
    W TS1;
    V { Pin=%#; }
    V { Pin=%#; }
    V { Pin=%#; }
  }
}
```

Substituting both # and % for one pin is NG

```
Procedures {
  "Replace" {
    W TS1;
    V { P1=#; P2=#; }
    V { P1=%; P2=%; }
    V { P1=#; P2=#; }
  }
}
```

Mixing # and % for one pin is NG

Substituting both # and % for one pin in the pin group notation is NG

```
Procedures {
  "Replace" {
    W TS1;
    V { Pin=##; }
    V { Pin=%%; }
    V { Pin=##; }
  }
}
```

In the pin group notation, mixing both # and % is NG

■Reference:

Example expression containing # in the Shift statement

```
Procedures {
     unload_load {
           W scan;
           C { all=100 001 00 XX 0000 XXXX; }
           Shift {
                  V {si1=#; si2=#; so1=#; so2=#;}
           }
     } // unload_load
} // Procedures

     ...omitted...

Pattern scan_loads {
    W wft_base;
    V { all = 110 111 00 XX 0000 XXXX;}
    Call unload_load {
            si1=1101110101;  si2=0000111011;
    }
    V { pis = 1010;} // FORCE PIs
    V { pos = HLHL;} // MEASURE POs
             // FIRE CAPTURE CLK (SUBSET OF all_scan_clks)
    V { capture_clks = 00; pos = XXXX;}
    Call unload_load {
            so1=HLHHHLHLHL;  so2=HHHLLHHLHL;
    }
}
```

Example expression containing # for typical pattern substitution in the Vector statement

```
Procedures {

   "Replace" {

      W TS1;

      C { 'P3+P4'=XX; }

      V { P1=%; P2=#; 'P3+P4'=#; }

      V { P1=%; P2=#; 'P3+P4'=#; }

      V { P1=%; P2=#; 'P3+P4'=#; }

   }

}

     ...omitted...

Pattern scan_loads {
    W wft_base;
    V { all = 110 111 00 XX 0000 XXXX;}
   Call "Replace" { P1=011; P2=110; 'P3+P4'=LLHH;}
}
```

◆Problem when setting the driver switching time more than the input waveform edge timing

- IEEE Specification (page 87)
  18.2 Waveform event definitions

- Classification

  Application:

  Indicates the contents that present no problem in describing or interpreting STIL data, but should be commonly recognized by the application tool and equipment which handle STIL data.

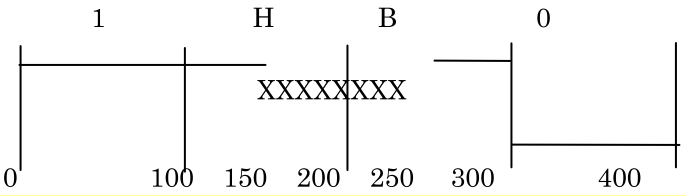
- Problem
  When you set the driver switching time more than the input waveform edge and try to describe the waveform within the scope of the current STIL specification, you need to use different WFCs or switch the WaveformTable for two different cases of input waveforms; one, which has switched from output to input, and another, which has switched from input to input. Without doing this, you can not describe the waveform events.

  On a tester supporting driver-enable timing, to perform a test, test patterns may need to be rewritten in some case due to the above constraint.

Example: When the driver switching time is more than the input waveform edge timing within the scope of the current STIL specification

When Test cycle: 100ns,
Input waveform: NRZ with delay=0,
and I/O switching time: 50ns

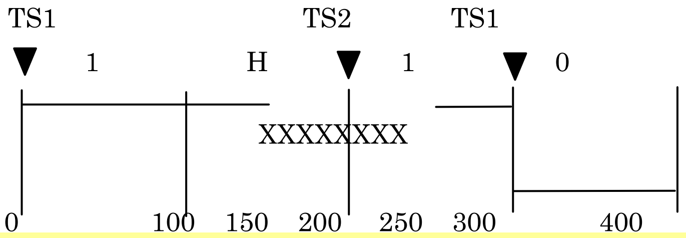
Using different WFCs for the two waveforms; one has switched from output to input, and another has stayed in input mode



```
01 { '0ns' D/U; }
LHTX {'50ns' Z; '80ns' L/H/T/X; }
AB {‘50ns' D/U; }
```

When you describe the input waveform, you have to change WFC 0/1 and A/B considering the mode switching, as a result, describing and understanding patterns are not simple.

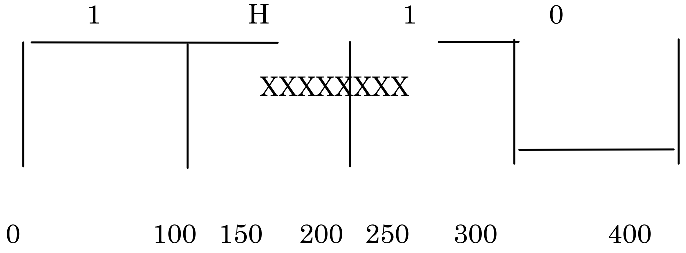
Changing WaveformTable for the two waveforms; one has switched from output to input, and another has stayed in input mode



```
WaveformTable "TS1" {
  01 { '0ns' D/U; }
  LHTX {'50ns' Z; '80ns' L/H/T/X; }
}
WaveformTable "TS2" {
  01 { ‘50ns' D/U; }
  LHTX {'50ns' Z; '80ns' L/H/T/X; }
}
```

You need to change WaveformTable as the mode changes I/O. Therefore, you may be unable to do this depending on timing resource constraint.

■ Further note for the problem (information only)



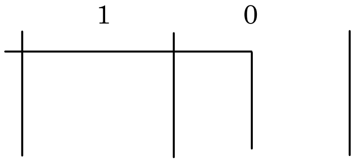
The reason the below is not good.

01 {‘0ns’P; '50ns' D/U; }
LHTX {'50ns' Z; '80ns' L/H/T/X; }

In the STIL specification, the definition of P is:

“Force logic to last driven state. Turn the drive on and go to the last drive state (i.e.,If the last drive state was "D", then go to low; if the last drive state was "U" then go to high).”

P is “Turn the drive on and go to the last drive state” as it said in the spec. In the above example, the I/O switching timing is 50ns, so until the 250ns in the 3rd cycle, it’s output mode, and from the 250ns, the input mode starts, but if “01 {‘0ns’P; '50ns' D/U; }, the driver is turned on at the beginning of the 3rd cycle (200ns). The above example is “input waveform: NRZ with delay=0, I/O switching time: 50ns”, but if the input waveform is defined as “01 {‘0ns’P; '50ns' D/U; }”, then when it changes from input 1 to input 0, the waveform changes at 50ns as shown below, so it becomes “NRZ with delay=50ns”.



Therefore, if the driver switching time is more than the input waveform edge, you need to define two waveforms separately; one, which stays in input mode and, another, which switches from output to input.

■Proposed solution

If the input timing is different for an input waveform switching from output to input and for another staying in input mode, it is recommended to describe (see the examples in the related materials) in accordance with the STIL specifications. However, on a tester supporting driver-enable timing, in order to avoid the complicated pattern data description, even in the following case of application's exception handling, the driver switching timing is made at 50ns as the same as L, H, T, and X when using WFC 0 and 1. So this case can be regarded and processed as the same as the examples in the problem-related materials.

That is,

```
  01{'0ns' D/U; }
  LHTX { '50ns' Z; '80ns' L/H/T/X; }
```

for test patterns defined in the above, pattern data "0, 1" of a cycle which changes from output to input

```
  01{'0ns' D/U; }
  LHTX { '50ns' Z; '80ns' L/H/T/X; }
  AB{ '50ns' D/U; }
```

can be regarded and processed as pattern data "A, B" defined in the immediately above. When an application operates as said above, to notate this process explicitly, you may use a drive event UserKeywords, e, which is newly defined in the problem-related materials.

When the application supporting exception handling mentioned above outputs the STIL data, it can be either in the form of complying with the STIL specifications or using a drive event UserKeywords e.

Defining a new drive event e as a UserKeyword

Declaration by the UserKeywords statement::

```
UserKeywords e;
```

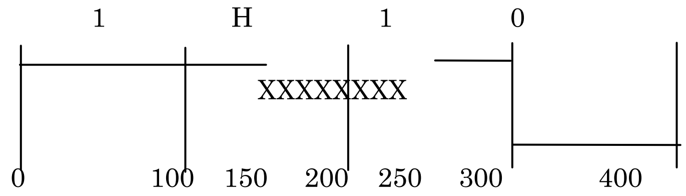
e may be used as an event code for timing definition in the WaveformTable block.

The following is the definition of e.

<Definition of e>

- Event that is valid only in cycles of which state change from driver-off to driver-on
- Event that sets the time to change to the first driver-on from driver-off, and the logic level is the same as event ForcePrior (P), i.e. the last drive state.
- Drive events appeared from the last driver-off state till the driver-on time set by e shall be invalid, and events thereafter shall be valid.

Timing example of using e:



```
UserKeywords e;
        .
        .
        .
01 { '0ns' D/U;  '50ns' E; }
LHTX {'50ns' Z; '80ns' L/H/T/X; }
```

You can describe the timing without changing WFCs or WaveformTable for the two waveforms; one, which has switched from output to input, and another, which has stayed in input mode

◆Problem of the test type description

■IEEE Specification (additional proposal)

TestType statement

■Classification

Description: Indicates the contents that should be recognized to describe or interpret STIL data.

■Problem

No statement to describe test types is provided in the specification.

■Proposed solution

It is recommended to describe test type within User USER_DEFINED of the Purpose statement in the Pattern or Pattern Burst block statement of the 1450.6 PatternInformation block.

<Format>

```
STIL1.0 {
  Design 2005;
  CTLMode 2005;
}
Environment (ENV_NAME) {
  (CTLMode (CTLMODE_NAME)  {
    (PatternInformation {
      (<Pattern | PatternBurst> (pat_or_burst_name) {
        (Purpose ( User TestType < FC | DC | FCDC | AC >) //test type    (1)
   })*//End Pattern or PatternBurst
     })//End PatternInformation
  })*//End CTLMode
}//End Environment
```

<Keywords explanation in the format>

(1)TestType:the keyword indicates the test pattern type

FC: the keyword indicates the function test pattern

DC: the keyword indicates the DC test pattern

FCDC: the keyword indicates the mix of function test pattern and DC test pattern.

AC: the keyword indicates AC test pattern

<Description Example1>

```
STIL 1.0 {
  Design 2005;
  CTL 2005;
}
PatternBurst BURST1 {
  PatList {
    TR1;
  }
}
PatternExec EXEC1 {
  PatternBurst  BURST1;
}
Pattern TR1{
}
Environment ENV1 {
  CTLMode MODE1 {
    PatternInformation {
       Pattern TR1 {
          Purpose   User TestType  FC;
    }//End Pattern
     }//End PatternInformation
  }//End CTLMode
}//End Environment
```

Pattern TR1 indicates it is the function test pattern.

<Description Example2>

```
STIL 1.0 {
  Design 2005;
  CTL 2005;
}
PatternBurst TR1 {
  PatList {
    PAT1;
    PAT2;
  }
}
PatternExec EXEC1 {
  PatternBurst  TR1;
}
Pattern PAT1{
}
Pattern PAT2{
}
Environment ENV1 {
  CTLMode MODE1 {
    PatternInformation {
       PatternBurst TR1 {
          Purpose   User TestType  FC;
    }//End PatternBurst
     }//End PatternInformation
  }//End CTLMode
}//End Environment
```

PatternBurst TR1 indicates the execution of function test.

For the detail, please see the Suggested Solution of “Problem of release/capture clock definitions should be distinguished in Transition pattern(1)” in the STIL Usage Guide 1450.6.

◆Problem of the description of infinite loops

- IEEE Specification (additional proposals)
  Infinite Loop statement

- Classification
  Description: Indicates the contents that should be recognized to describe or interpret the STIL data.

- Problem
  In a tester, the test pattern may need to get into an infinite loop, but no statement is provided.

- Proposed solution
  It is recommended to define InfiniteLoop by UserKeywords and describe InfiniteLoop statement in Pattern block when you get the test pattern into an infinite loop.

```
<format>
UserKeywords InfiniteLoop;
Pattern “pattern_name”{
  W”WaveformTable_domain_name”;
  V {“pin_group_name”=mnemonic;}
                :       :
  V {“pin_group_name”=mnemonic;}
                :      :
   LABEL:InfinitieLoop {                 // We have proposed STIL WG to add the format (may be
                                  //multiple). Describes the Infinite loop range. For the DC
                   //measurement, this represents the measurement address
                   //(pattern operation range). The LABEL name is referenced by
                   //“DC_maesurement_name” in the Measure statement of
                   //PatternExec.
  V {“pin_group_name”=mnemonic;}
 }
}
```

◆Problem of the test type description in the PatternExec block

- IEEE specification (additional proposals)
  TestType usage in PatternExec

- Classification

  Description:

  Indicates the contents that should be recognized to describe or interpret the STIL data

- Problem

  When executing a test, to describe a statement to define a test type in the PatternExec block is required, but such statement is not provided.

- Proposed solution
  It is recommended to describe “TestType test type;” in the PatternExec block.

```
<format>
UserKeywords TestType ;
PatternExec  domain_name{
   TestType     "FC";       //Test type
   PatternBurst "FC1";
}
```

◆ Problem with DataBitCount pattern omission

■IEEE Specification1450.0 (Page101)

21.2 Multiple-bit cyclized data

Vector statements may contain references to groups that were defined to support multiple-bit data definitions. Multiple-bit data is defined in waveforms through the use of square brackets (See Clause 18 for additional details.)

Multiple bit data may be presented in cyclized data in two formats. Each format has a set of requirements in order to process this multiple-bit data.

The first format is identical to the Cyclized Data statement presented in 21.1, with the extension that the VEC_DATA length is longer than the length necessary to assign a bit to each signal referenced in *sigref_expr*. The number of WaveformChars is defined by the DataBitCount statement associated with the definition of the Signal or Group being referenced (and the default is 1).

The second statement format is:

sigref_expr { (*vec_data*;)+ }

In this format, there are one or more *vec_data*; statements. Each statement contains the number of WaveformChar references necessary to apply to all signals defined in the *sigref_expr*. There are as many statements defined as index values present in the waveform definition applied.

In addition to these constructs, multiple-bit data has the following additional requirements:

- •The *sigref_expr* shall be a group only. The definition of this group shall include a DataBitCount attribute. The definition of this group shall also include a Base attribute identifying the multiple-bit WaveformChars that may be applied to the signals in this group.
- •The waveform referenced by the WaveformChars shall define all WaveformChars listed in the Base attribute in a single waveform definition. All waveforms applied across all signals defined in a multiple-bit group shall have the same number of indexed timed events.

■ Classification

Description:

Problem with DataBitCount description

■ Problem

If an empty Vector statement is described after the Vector statement in which DataBitCount is described, it is not clear whether the empty Vector statement follows the last state, or is regarded as the same description as the Vector statement immediately before.

```
Signals {
  A In; B Out; "Pin"  In { DataBitCount 2; }
}
Timing {
  WaveformTable "TS1" {
    Period '100ns' ;
    Waveforms  {
      "A" { 01 { '0ns' D/U; } }
      "B" { LHTX { '0ns' X; '10ns' L/H/T/X; } }
      "Pin"  { 01 { '0ns' D/U [0]; '10ns' D/U [1]; }}}
  }
}
Pattern "PAT1"{
  W "TS1";
    C { "A"=0; "B"=X; }
    V { "Pin"{0;1;} }
    V {}
}
```

→ Which of the following would be the description of this Vector statement is not clear.

Follows the last state: V{“Pin” {1;1;}}

The same as the Vector immediately before: V{ "Pin"{0;1;} }

■ Proposed solution

It shall be interpreted that if an empty Vector statement is described after the Vector statement in which DataBitCount is described, the description of the empty Vector is the same as that of the Vector statement immediately before.

```
Signals {
  A In; B Out; "Pin"  In { DataBitCount 2; }
}
Timing {
  WaveformTable "TS1" {
    Period '100ns' ;
    Waveforms  {
      "A" { 01 { '0ns' D/U; } }
      "B" { LHTX { '0ns' X; '10ns' L/H/T/X; } }
      "Pin"  { 01 { '0ns' D/U [0]; '10ns' D/U [1]; }}}
  }
}
Pattern "PAT1"{
  W "TS1";
     C { "A"=0; "B"=X; }
     V { "Pin"{0;1;} }
     V {}   →   V { "Pin"{0;1;} }
}
```

To be consistent with the description when WFCs are omitted in the Vector statement, the description of DataBitCount shall be interpreted as the same as that of the Vector statement immediately before. So for DataBitCount, repetition of the same description is not needed.

◆ Problem of the measurement order in IDDq tests

- IEEE Specification1450.0 (Page107)

22.10 IddqTestPoint statement

The IddqTestPoint statement defines a place in the Pattern where an IDDq measurement may be performed. IddqTestPoint statements are optional. The syntax of the IddqTestPoint statement is:

(LABEL :) **IddqTestPoint;**

The following is an example Iddq TestPoint statement:

```
V { new-delta-change-data }
IddqTestPoint; // perform Iddq measurement between 2 vectors
V { new-delta-change-data }
```

- Classification

Description:

Problem in the STIL description

■ Problem

In an IDDq test, it is more efficient to start the measurement in order from highest fault detection capability point to lowest. However, in the current IddqTestPoint statement specification, information concerning the measurement order of multiple IDDq test points, such as fault coverage, can not be described. Therefore, in order to create efficient test programs, users need to refer the result file of fault simulation to set the measurement order in the IDDq test from highest fault detection capability point to lowest.

■ Proposed solution

It is recommended to enhance the IddqTestPoint statement and use UserKeywords Detection when describing the information concerning the measurement order of IDDq test points.

UserKeywords Detection;

Detection information description is added to the current statement {IddqTestPoint}, so it has to be defined by UserKeywords.

Current (LABEL :) **IddqTestPoint**;

Added (LABEL :) **IddqTestPoint {Detection *detection_expr(detection_expr);* }**

***detection_expr* : numerical value…indicates the measurement order in the IDDq test**

numerical value%...indicates fault coverage

“Detection detection_expr (detection expr);” has been added to IddqTestPoint, so that an application to create test programs can do so taking the measurement order in an IDDq test into account.

■ Example

[Current]

<STIL-1>

```
Pattern PAT1 {
  V { PATT=001001001HLLHLLHLHLH;}
  V { PATT=100100111LLHLHLLHLHL;}
  V { PATT=111110010HHLHHHHLHHL;}
  IddqTestPoint;   ...(1)
  V { PATT=001001001HLLHLLHLHLH;}
  IddqTestPoint;   ...(2)
  V { PATT=100100111LLHLHLLHLHL;}
  V { PATT=111110010HHLHHHHLHHL;}
  IddqTestPoint;   ...(3)
  V { PATT=111110010HHLHHHHLHHL;}
}
```

<STIL-2>

```
Pattern PAT1  {
   V { PATT=100100111LLHLHLLHLHL;}
   V { PATT=111110010HHLHHHHLHHL;}
   IddqTestPoint; ...(4)
   V { PATT=001001001HLLHLLHLHLH;}
   IddqTestPoint; ...(5)
   V { PATT=100100111LLHLHLLHLHL;}
   V { PATT=111110010HHLHHHHLHHL;}
}
```

<IDDq test measurement order>

<STIL-1>(1)

<STIL-1>(2)

<STIL-1>(3)

<STIL-2>(4)

<STIL-2>(5)

*The measurement order is not specified, so they are measured in the order of description.

[When specifying fault coverage (accumulated) and the measurement order with Detection]

<STIL-1>

```
Pattern PAT1 {
   V { PATT=001001001HLLHLLHLHLH;}
   V { PATT=100100111LLHLHLLHLHL;}
   V { PATT=111110010HHLHHHHLHHL;}
   IddqTestPoint { Detection 50.0% 1;}...(1)
   V { PATT=001001001HLLHLLHLHLH;}
   IddqTestPoint { Detection 70.0% 3;}...(2)
   V { PATT=100100111LLHLHLLHLHL;}
   V { PATT=111110010HHLHHHHLHHL;}
   IddqTestPoint { Detection 91.7% 2;}...(3)
   V { PATT=111110010HHLHHHHLHHL;}
}
```

<STIL-2>

```
Pattern PAT1 {
    V { PATT=100100111LLHLHLLHLHL;}
    V { PATT=111110010HHLHHHHLHHL;}
    IddqTestPoint { Detection 92.5% 5;}...(4)
    V { PATT=001001001HLLHLLHLHLH;}
    IddqTestPoint { Detection 98.5% 4;}...(5)
    V { PATT=100100111LLHLHLLHLHL;}
    V { PATT=111110010HHLHHHHLHHL;}
}
```

<IDDq test measurement order>

<STIL-1>(1)

<STIL-1>(3)

<STIL-1>(2)

<STIL-2>(5)

<STIL-2>(4)

*Can measure from highest fault detection capability point to lowest.

◆ Problem with pin definitions in the first Vector statement in the Pattern block

- IEEE Specification (Page109 23.2 Pattern initialization)
- Classification

  Description:

  Indicates the contents that should be recognized to describe or interpret STIL data
- Problem

  (1) If pins not described in the first Vector statement in the Pattern block are described in the Include file or in the Procedure block, which is used as a separate Pattern block in the middle of the Pattern block, it is unknown whether the Default State value should be used for the undescribed pins or this case should be a violation of the specification.

  (2) If there are pins not described in the first Vector statement in the Procedures block or Include file used as a separate Pattern block at the top of the Pattern block, it is unknown whether the DefaultState value should be used for pins not described at the top of the Procedures block or Include file, or they should be handled as a violation of the specification.

Problem (1) example

```
  Signals{
   "A1" In; "B1" Out; "C1" InOut;}
  Procedures {
    "xxx" {
        W “T1”;
V{"A1"=1;"B1"=X;"C1"=0;}
      }
  }
    Pattern "P1" {
       W "T1";
       V {"A1"=1;"B1"=H;}
       Call "xxx";
    }
```

It is unknown whether the pattern of signal C1 should be the Default State value or handled as an error.

Problem (2) example

There are pins not described in the first Vector statement in the Procedures block used as a separate Pattern block at the top of the Pattern block, and the pins are used in that Pattern block.

```
 Signals{
       "A1" In; "B1" Out; "C1" InOut;}
      Procedures {
         "xxx" {
         W "T1";
         V{"A1"=1;"B1"=0;}
      }
    }
    Pattern "P1" {
         W "T1";
         Call "xxx";
         V {"A1"=1;"B1"=0;"C1"=X; }
    }
```

It is unknown whether the pattern of signal C1 should be the Default State value or handled as an error.

Problem (2) example

There are pins not described in the first Vector statement in the Procedures block used as a separate Pattern block at the top of the Pattern block, and the pins are not used in that Pattern block.

```
  Signals{
      "A1" In; "B1" Out; "C1" InOut;}
        Procedures {
          "xxx" {
            W "T1";
            V{"A1"=1;"B1"=0; }
          }
      }
      Pattern "P1" {
          W "T1";
          Call "xxx";
          V {"A1"=1;"B1"=0; }
       }
```

It is unknown whether the pattern of signal C1 should be the Default State value.

■ Proposed solution

(1) If pins not described in the first Vector statement in the Pattern block are described in the Include file or in the Procedure block, which is used as a separate Pattern block in the middle of the Pattern block, it shall be a violation of the specification because this case should be treated the same as pins not used in the first Vector in the Pattern block being used in a later Vector.

It is recommended that the states of all the pins used in a Pattern block should be described in the first Vector statement in that Pattern block.

(2) If there are pins not described in the first Vector statement in the Procedures block or Include file used as a separate Pattern block at the top of the Pattern block, and the pins are not described through the Pattern blocks, the pin at the top of the Pattern block accepts the DefaultState value. If pins not described in the first Vector statement in the Procedures block or Include file used as a separate Pattern block are described in the middle of the Pattern block, it shall be a violation of the specification.

◆Problem of pattern description allowed when the Alignment statement is defined

■ IEEE Specification(Page109 23.2 Pattern initialization)

Alignment: An optional statement indicating how to map the bits of a non-WaveformChar numeric value into the individual signals of a multiple-signal definition or group. This attribute is significant only for multiple-signal definitions or groups, and is applied only in the context of Vector assignments (see 21.1 and 21.2). It is not applied to scan environments (21.4), as scan relies on padding operations to resolve data length issues.

■ Classification

Description: Indicates the contents that should be recognized to describe or interpret STIL data.

■ Problem

It is unclear whether both WFCs and numerical values may be described as pattern to the Vector statement when you describe the Alignment statement in defining signals or a signal group in the or SignalGroups block.

```
(ex) The pattern is described by Hex
   SignalGroups {
      "group"=‘"A1"+"B1"+"C1"'{
         Base Hex 01;  Alignment LSB;
       }
    }

   Pattern "P1" {
      W "T1";
      :
      V { "group"=¥h A; }
    }
```

```
(ex) The pattern is described by WFC
    SignalGroups {
      "group"=‘"A1"+"B1"+"C1"' {
        Alignment LSB;
      }
    }

   Pattern "P1" {
     W "T1";
      :
     V { "group"=1010; }
   }
```

■Proposed solution

When you describe the Alignment statement in defining signals or a signal group in the Signals or SignalGroups block, if Hex or Dec numerical values are used for the target Vector statement, the sequence of bit data from MSB or LSB specified by the Alignment statement is assigned to the signals and it becomes a WFC string corresponding to the sequence of bit data. The Alignment statement becomes invalid for the order of WFCs. When you describe WFCs for patterns, the numbers of WFCs and signals have to be the same.

(ex) The pattern is described by Hex

```
  SignalGroups {
     group=‘"A1"+"B1"+"C1"'{
         Base Hex 01;  Alignment LSB;
     }
  }

  Pattern "P1" {
      W "T1";
         :
      V { group=¥h A; }
  }
```

(ex) The pattern is described by WFC

```
    SignalGroups {
     group=‘"A1"+"B1"+"C1"' {
      Alignment LSB;
     }
  }

   Pattern "P1" {
     W "T1";
      :
      V { group=1010; }
     }
```

This is a violation of the specifications.

◆Problem of DAC units settings of testers in ADC conversion tests

■IEEE Specification (additional proposal)

ADCUnits block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the settings of DAC units in ADC conversion tests.

■Proposed solution

It is recommended to describe the DAC unit settings of the tester using ADCUnits block defined by UserKeywords when declaring AMD P7 in STILextentions.

```
<Format>
STIL 1.0{
    DCLevels 2002; AMD P7;
}
UserKeywords ADCUnits;
ADCUnits  " Domain name “{ //Block to describe settings of DAC units of testers
    InheritADCUnits  “ADCUnits Domain Name”; //Inherits conditions of other ADC units blocks
    AinSignals  ’“signal_pin_name”+ ”signal_pin_name”’ ··· ;
              // DAC units’ output connects to the ADC input signals
    ForceRange   ‘Measurement_range ’,‘resolution’,‘setting_accuracy’;
              //Specify applied setting range (the value ranges from – to +), resolution,
              //and setting accuracy (the value ranges from – //to +).
              //Used as conditions for selecting tester’s power supply unit range.
              //The ForceRange statement may be omitted. Also not all arguments need to be described
              //(some may be omitted) when using a ForceRange statement.
    ZeroCode_Voltage  ‘voltage_value’ ; // Specify the ideal voltage value of input signals which outputs zero code
    FullCode_Voltage  ‘voltage_value’ ; // Specify the ideal voltage value of input signals which outputs full code
    ADC_Bit_Number bit_number ; // Specify the bit number of ADC data
    Ain_Resolution ‘resolution’;   // Specify the resolution of the input signal.
                                   //Operational expression may also be descripted
    ADC_Resolution ‘resolution’; // Specify the resolution of ADC. Operational expression may also be descripted
    Limit ‘low_limit,high_limit’;    // Judgement Limit
    OverDrive_Voltage ‘low_voltage,high_voltage’; // Specify the allowable voltages of analog inputs
              // Low voltage: Specify a difference from the reference power supply (negative)
              // High voltage: Specify a difference from the reference power supply (positive)
    DC_Offset_Voltage  < ‘Low_voltage , HIgh_voltage’; | ‘voltage_value’>;
            //Specify analog input offset voltage.
            //Low voltage: Specify negative offset value
            //High voltage: Specify positive offset value
            //Only voltage: Specify the uniform offset value for the positive and negative.
    Gain_Ratio  ‘Gain Coefficient ’ ;
            //Describe the gain coefficient (ratio per amplitude) of DAC unit amplitude.
            //The amplitude that imultiplies the gain coefficient by DAC unit amplitude is
            //the amplitude which is input actually.
            // Gain coefficient is described decimally(<1).
            //*DAC unit amplitude multiplied by Gain coefficient equals Input amplitude to ADC
            //It is not specified when not changing (attenuating) the Gain.
    Slope < Positive | Negative > ;  //Specify the slope of the input waveform.
    DoutCapUnits  DoutCapUnits domain_name  ;
            //Domain name of DoutCapUnit to be referenced.
            //Specify it only when measuring them at the same time using multiple units in PatternExec
}
```

◆Problem of condition descriptions for reference power supply in ADC conversion tests

■IEEE Specification (additional proposal)

VrefLevels block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the conditions for reference power supply in ADC conversion tests.

■Proposed solution

It is recommended to specify the conditions for reference power supply using VrefLevels block defined by UserKeywords when declaring AMD P7 in STILextentions.

```
<Format>
STIL 1.0{
    DCLevels 2002; AMD P7;
}
UserKeywords VrefLevels;
VrefLevels  domain_name {
   reference_power_supply_name {
    Force_VRH     'voltage_value';   //Specify the applied voltage value of the positive supply
    Force_VRL     'voltage_value';   //Specify the applied voltage value of the negative supply
    Force_VRM     'voltage_value';   //Specify the mid-voltage
    ForceRange   'force_range ','resolution','setting_accuracy';
                // Specify applied setting range (the value ranges from – to +), resolution, and setting
                //accuracy (the value ranges from – to +.) Used as conditions for selecting tester's power       //supply
unit range.
                //The ForceRange statement may be omitted. Also not all arguments need to be described     //(some
may be omitted) when using a ForceRange statement.
    IClamp  'negative_clamp_value , positive_clamp_value ';   // Specify the clamp value range
   }
 }
```

◆Problem of description for data loading unit of testers in ADC conversion tests

■IEEE Specification (additional proposal)

DoutCapUnits block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the data loading unit of testers in ADC conversion tests.

■Proposed solution

It is recommended to specify the data loading unit using DoutCapUnits block defined by UserKeywords when declaring AMD P7 in STILextensions.

```
<Format>
STIL 1.0{
  DCLevels 2002; AMD P7;
}
UserKeywords DoutCapUnits;
DoutCapUnits domain_name{ //This block describes the settings of tester’s data loading unit
     DoutSignals < MSB | LSB> ’“ signal_pin_name or signal_group_name”+ ”signal_pin_name or signal_group_name’ ··· ’;
         // Specify signals whose data is to be loaded. The left signal pin is MSB and the
         // right signal pini is LSB by default. If ”Lsb” is specified, the left signal pin
         //is LSB and the right signal pini is MSB.
         //One or more signal pin names or signal group names can be specified.
         //In the case of signal group, the order of the signal pins which are defined in
         //SignalGroups depends on the data alignment order.
    DoutSignalsMapping  <BusNo | Alpha | BusNoAlpha | AlphaBusNo> ;
         //Enable to define only when DoutSignals’ signal pin names are specified with  SignalGroups names. //Definition is not always
         required. This specifies the order of signals whose data is to be loaded.
         //BusNo: Bus number order. The larger number is MSB.
         //Alha: alphabetic order. ABC side is MSB.
         // BusNoAlha: if Bus numbers and alphabets are mixed, the Bus number order takes precedence.
         // AlhaBusNo: if Bus numbers and alphabets are mixed, alphabetic order takes precedence.
    CapMode < Pararel | Serial < MSB | LSB> > ;
         //Select a data load mode, either parallel or serial
    Sampling_Count count ; //Specify the number of data loading times per resolution of analog
         //input voltage after digital conversion. If more than once, take the avarage as the data value.
}//End DoutCapUnits
```

◆Problem of referencing the block names in ADC conversion tests

■IEEE Specification (additional proposal)

DCSets block, PatternExec block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to reference ADCUnits/VrefLevels/DoutCapUnits block names in PatternExec.

■Proposed solution

(1)Define ADCUnits/VrefLevels/DoutCapUnits block names defined by UserKeywords in DCSets Block to enable referencing in PatternExec.

(2)Enable referencing ADCUnits/VrefLevels/DoutCapUnits block names defined by UserKeywords directly in Pattern Exec block.

<Format 1> Case of defining ADCUnits/VrefLevels/DoutCapUnits block names in DCSets block and reference it in PatternExec

```
UserKeywords ADCUnits VrefLevels DoutCapUnits;
DCSets “domain_name”{
    ADCUnits “ADCUnits_domain_name”;
              //Specify the domain name of the ADCUnits block to be referenced.
    VrefLevels “VrefLevels_domain_name”;
              // Specify the domain name of the VrefLevels block to be referenced.
    DoutCapUnits “DoutCapUnits_domain_name”;
              // Specify the domain name of the DoutCapUnits block to be referenced.
}//End DCSets
PatternExec “domain_name”{
              DCSets     “DCSets domain_name”;
                  :                            :
}
```

```
<Format 2> Case of referencing ADCUnits/VrefLevels/DoutCapUnits block names directly in PatternExec blocks
UserKeywords ADCUnits VrefLevels DoutCapUnits;
PatternExec “domain_name”{
    ADCUnits “ADCUnits_domain_name”;
                // Specify the domain name of the ADCUnits block to be referenced
                // Description for measurement using multiple ADC units.
                //(If necessary, the application range can be switched for every ADCUnits. )
                // < simultaneous application >
    ADCUnits { ParallelForce "ADCUnits_domain_name_1" "ADCUnits_domain_name_2"
                                          "ADCUnits_domain_name 3"‥; } // < serial application >
    ADCUnits { SerialForce   "ADCUnits_domain_name_1" "ADCUnits_domain_name_2"
                                          “ADCUnits_domain_name_3” ‥; }
                :           :
    VrefLevels “domain_name”; // Specify the domain name of the VrefLevels block to be referenced.
    DoutCapUnits “domain_name”;
                           // Specify the domain name of the DoutCapUnits block to be referenced
                           // Description for measurement using multiple DoutCap units.
                           // <simultaneous data loading>
    DoutCapUnits { ParallelCapture "DoutCapUnits_domain_name_1"
                           “DoutCapUnits_domain_name_2”  “DoutCapUnits_domain_name_3” ‥; }
                           // <serial data loading>
    DoutCapUnits { SerialCapture "DoutCapUnits_domain_name_1"
                            “ DoutCapUnits_domain_name_2”  “DoutCapUnits_domain_name_3” ‥; }
                :           :
}
```

◆Problem of description for ADC conversion operation patterns in ADC conversion tests

■IEEE Specification (additional proposal)

Pattern statement

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the ADC conversion operation patterns in ADC conversion tests.

■Proposed solution

It is recommended to define StartAnalogVoltage, StopAnalogVoltage and ADCMeasLoop by UserKeywords, and describe the starting and ending points of forcing DAC unit outputs to device analog input using the StartAnalogVoltage and StopAnalogVoltage statements, and describe the repeat of ADC conversion operation using the ADCMeasLoop statement.

```
<Format>
UserKeywords StartAnalogVoltage StopAnalogVoltage ADCMeasLoop;
Pattern “pattenr_name” {
                    :
   StartAnalogVoltage ; //Indicates starting point of forcing DAC unit outputs to device analog input
   ADCMeasLoop {  //Repeats ADC conversion operation
       V { “pin_group_name”=mnemonic ; } //Specify ADC conversion operation patterns
   }
   StopAnalogVoltage ; //Indicates the ending point of forcing DAC unit outputs to device analog input
                    :
}
```

◆Problem of ADC unit description in DAC conversion tests

■IEEE Specification (additional proposal)

DACUnits block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the settings for ADC units in DAC conversion tests.

■Proposed solution

It is recommended to specify the conditions for the ADC units using DACUnits block defined by UserKeywords when declaring AMD P7 in STILextensions.

<Format>
STIL 1.0{
  DCLevels 2002; AMD P7;
}
UserKeywords DoutCapUnits;
DACUnits « **domain_name**” **{** *// Block to describe the ADC unit settings of testers*
    InheritDACUnits   DACUnits domain_name; *// Inherits other DACUnits block conditions*
    AoutSignals ’“signal_pin_name (or signal_group_name)”+” signal_pin_name (or
        signal_group_name)”+··’;
    //Output signals to which ADC unit inputs connect (may be multiple).
    DcodeSignals < MSB | LSB> '" signal_pin_name (or signal_group_name)”+
        " signal_pin_name (or signal_group_name)”+··’;
    //In case of synchronizing the loading per DAC digital codes when loading DAC outputs in ADC units,
    //specify the digital signal name. For orresponding digital code values, refer to the Cap pattern in
    //Pattern block. In case digital input signals transit from zero-code to full code(or full-code to zero-
    //code), Slope statement can be used, so this statement is not needed.
    //In case the DAC digital code order is D7 D0, specify msb(by default. Can be omitted)
    //If D0 D7, specify lsb. If Slope statement is also defined, this statement takes precedence.
    //If signal group names are specified, orders of signal pins in SignalGroups will be considered
    //for their orders.

```
DcodeAoutMapping {“AoutSignals signal_pin_name1” ‘“signal_pin_name (or
               signal_group_name)”+
     ” signal_pin_name (or signal_group_name)”+··’;
     “AoutSignals signal_pin_name2” ‘“signal_pin_name (or signal_group_name)”+
     ” signal_pin_name (or signal_group_name)”+··’;
               :
    }
    //Match the digital input signals to the analog output signals in case multiple pins are
    //specified for DcodeSignals and AoutSignals.
MeasRange  ‘measure_range’,‘resolution’,‘setting_accuracy’;
       //Describe Measure range((the value ranges from – to +),  resolution, and setting accuracy (the value //ranges
       from – to +). Used for conditions of selecting ADC unit range of testers.
       //MeasRange statement can be omitted. Also not all arguments need to be described
       //(some may be omitted) when using a ForceRange statement.
ZeroCode_Voltage  ‘voltage_value’
       //Specify the ideal voltage value of DAC input signals which outputs zero code
FullCode_Voltage  ‘voltage_value’
       // Specify the ideal voltage value of DAC output signals which outputs full code
DC_Offset_Voltage  < ‘low_voltage, high_voltage’ | ‘voltage_value’>;
       // Specify the analog output offset value
       //Specify analog input offset voltage.
       //Low voltage: Specify negative offset value
       //High voltage: Specify positive offset value
       //Only voltage: Specify the uniform offset value for the positive and negative.
Gain_Ratio  ‘Gain Coefficient ’ ;
        //Describe the gain coefficient (ratio per input full-scale voltage) of DAC units block.
        //*DAC unit input full-scale voltage multiplied by Gain coefficient equals the full-scale voltage which //can be
        input actually to the DCA unit.
        //It is not specified when no changing (attenuating) the Gain.
Slope < Positive | Negative > ;  //Specify the slope of the DAC output waveform
            // If DcodeSignals statement is described, this statement is not necessary.
            // If it is described at a same time, DcodeSignals statement takes precedence.
DAC_Bit_Number bit_number ; //Specify DAC data bit number
DAC_Resolution ‘resolution’;  // Specify DAC resolution. Operational expression may also be described.
Sampling_Count count;
        //Specify the number of sampling times per digital code at loading analog output
    Limit ‘low_limit,high_limit’; //Judgement Limit
    FullScaleLevelResio < On | Off > ;
        // Indicatae this is a DACUnit block in which full scale level ratio
        // testing between plural DAC output signals is performed.
}
```

◆Problem of condition descriptions for reference power supply in DAC conversion tests

■IEEE Specification (additional proposal)

VrefLevels block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the conditions for reference power supply in DAC conversion tests

■Proposed solution

It is recommended to specify the conditions for reference power supply using VrefLeveles block defined by UserKeywords when declaring AMD P7 in STILextensions.

```
<Format>
STIL 1.0{
   DCLevels 2002; AMD P7;
}
UserKeywords VrefLevels;
VrefLevels ”domain_name” {
 ”reference_power_supply_name” {
                  Force_VRH     ‘voltage_value’; //Specify the applied voltage value of the positive supply
                  Force_VRL     ‘voltage_value’; // Specify the applied voltage value of the negative supply
                  Force_VRM     ‘voltage_value’; //Specify the mid-voltage
                  ForceRange   ‘force_range ’,‘resolution’,‘setting_accuracy’;
                  // Specify applied setting range (the value ranges from – to +), resolution, and setting
                  //accuracy (the value ranges from – to +.) Used as conditions for selecting tester’s power       //supply
unit range.
                  //The ForceRange statement may be omitted. Also not all arguments need to be described     //(some
may be omitted) when using a ForceRange statement.
     IClamp  ‘negative_clamp_value , positive_clamp_value ’;   // Specify the clamp value range
     }
 }
```

◆Problem of block-name reference in DAC conversion tests

■IEEE Specification (additional proposal)

DCSets block, PatternExec block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to reference ADCUnits/VrefLevels block names in PatternExec

■Proposed solutions

(1) Define the block names of DACUnits/VrefLevels defined by UserKeywords in DCSets block to enable referencing in PatternExec

(2) Enable referencing the block names of ADCUnits/VrefLevels defined by UserKeywords directly in PatternExec block

<Format 1> Define DACUnits/VrefLevels block names in DCSets block and ferer to them in PatternExec

```
UserKeywords DACUnits VrefLevels;
DCSets “domain_name”{
        DACUnits “DACUnits_domain_name”; //Describe the domain name of DACUnits to be referenced
        VrefLevels “VrefLevels_domain_name”; //Describe the domain name of VrefLevels block to be
                                                  //referenced
}//End DCSets

PatternExec  “domain_name”{
        DCSets  “DCSets_domain_name”;
         :                       :
}
```

```
<Format 2> Refer to DACUnits/VrefLevels block names directly in PatternExec block
UserKeywords DACUnits VrefLevels;
PatternExec “domain_name”{
    DACUnits   “DACUnits_domain_name”;
            // Specify the domain name of the DACUnits block to be referenced
            // Description for measurement using more than one DAC unit.
            // If necessary, measurement range can be switched per DACUnits
            //<simultaneous measurement >
    DACUnits { ParallelMesure “DACUnits_domain_name_1”“DACUnits_domain_name_2”
                        “DACUnits_domain_name_3” ··; }
            // * “DACUnits_domain_name” must be more than one.
            //<serial measurement>
    DACUnits { SerialMeasure “DACUnits_domain_name_1”“DACUnits_domain_name_2”
                        “DACUnits_domain_name_3” ··; }
            :           :
            // * “DACUnits_domain_name” must be more than one.
    VrefLevels “VrefLevels_domain_name”;
            :           :
            //Specify the domain name of the VrefLevels block to be referenced
}
```

◆Problem of descriptions for DAC conversion operation patterns in DAC conversion tests

■IEEE Specification (additional proposal)

Pattern statement

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the patterns of DAC conversion operation in DAC conversion tests

■Proposed solution

It is recommended to define DACMeasPattern, Cap and DaCUnits by UserKeywords, define DACMeasPattern block in Pattern block, and describe the patterns of DAC conversion operations using Cap statement and DACUnits statement.

```
<Format>
UserKeywords DACMeasPattern Cap DACUnits;
Pattern “pattern_name” {
        :
    DACMeasPattern { //Indicate the DAC conversion range. In this block, DACUnits signal names
                     //connected to AutoSinal are omitted.
    V  {“pin_group_name”=mnemonic; } //Specify the DAC conversion opration pattern.
    Cap {“pin_group_name”=mnemonic; } // Specify the DAC conversion opration pattern.
                     //Specify the location where a digitizer loads DAC results.
    DACUnits “DACUnits_domain_name”; //Specify the domain name of the DACUnits block to be
                     // referenced. Switch the DACUnits conditions in the middle of pattern
    V  {“pin_group_name”=mnemonic; }
    Cap {“pin_group_name”=mnemonic; }
                :
    }
                :
}
```

◆Problem of settings for measuring instruments for testers in PLL tests

■IEEE Specification (additional proposal)

PeriodCounter block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to specify the settings for measuring instruments for testers in PLL tests

■Proposed solution

It is recommended to define PeriodCounterLoop by UserKeywords and use PeriodCounter block in Pattern block in order to describe the settings for measuring instruments for the testers.

```
<Format>
UserKeywords PeriodCounter;
PeriodCounter “domain_name “ { //Block to describe the settings for measuring insturments for the testers
    InheritPeriodCounter  “PeriodCounter_domain_name”;
                //Inherits conditions of other PeriodCounter block
                //May be omitted.
   TestMode <Priod | Freq>; //Specify the measurment mode. Can be omitted.
                                //It is Period by default.
                                //Priod: Handles measurement value by the period (time).
                                //Freq: Handles measurement value by the frequency (Hz).
    MeasureRange ‘minimum_measurement_value’, ‘maximum_measurement_value’,’resolution’:
                                //Describe the measurement range and resolution by period (time) or frequency (Hz)
    MeasureSignals “signal_pin_name (or signal_group_name)”;
                                //Specify the pin name to measure priod (may be more than one)
    ComparatorLevels ” signal_pin_name (or signal_group_name)” {‘Variables (values) ’;}
                //Sets the waveform detection voltage of the measured signal.
                //Can be omitted or more than one. Use the VOL value by default.
    Limit‘Low_limit,High_limit’;
                //Judgement Limit. If the TestMode is Priod, specify by the period time.
                //If TestMode is Freq, specify by the Hz.
}//End PeriodCounter
```

◆Problem of reference of measuring instruments' settings of testers in PLL tests

■IEEE Specification (additional proposal)

PatternExec block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

In PatternExec, unable to refer to the settings of measuring instruments of testers in PLL tests

■Proposed solution

It is recommended to enable referencing PeriodCounter block defined by UserKeywords in PatternExec block.

```
<Format>
UserKeywords PeriodCounter;
PatternExec "domain_name"{
    PeriodCounter "PeriodCounter_domain_name";
                //Describe the domain name of PeriodCounter block to be referenced
                :               :
}
```

◆Problem of description to specify the loop of Period counter’s measurement patterns in PLL tests

■IEEE Specification (additional proposal)

Pattern statement

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to describe the loop of the measuring patterns of Period counters in PLL tests

■Proposed solution

It is recommended to describe the loop of the measuring patterns of Period counters using PriodCounterLoop statement in Pattern block after defining PeriodCounterLoop by UserKeywords.

```
<Format>
UserKeywords PeriodCounterLoop;
Pattern “pattern_name” {
    LABEL:PeriodCounterLoop{
                 // Specify the loop of the measuring patterns of Period counters. Exit the loop after finishing
                 // measurements by the measuring instruments.
    W “WaveformTable_domain_name”;
    V {“pin_group_name”=mnemonic; }
                 :
    }
}
```

◆Problems of settings for MemoryBist pattenrs

- IEEE Specification (additional proposal)

  MemoryBist block

- Classification

  Description:

  Indicates the contents that should be recognized to describe or interpret the STIL data.

- Problems

  (1)Unable to describe the settings for MemoryBist patterns

  (2)Unable to describe the settings for MemoryBist patterns per MemoryBist circuit and Redundancy(BIRA) circuit

  (3)Unable to describe the settings for MemoryBist patterns per RAM macro

- Proposed solutions

  (1)Describe the settings for MemoryBist patterns using MemoryBist block defined by UserKeywords.

  (2)Use Instance block in MemoryBist block defined by UserKeywords.

  (3)Use MacroName block in MemoryBist block defined by UserKeywords.

<Format 1> Case of describing MemoryBist pattern settings using MemoryBist block

```
UserKeywords MemoryBist;
MemoryBist domain_name{// Beginning of MemoryBist block
            //If the BIRA <Built-In Redundancy Allocation> circuit is not provided, BIRA related keywords (BiraIf,
            //RepairFlagPin, RepairDatatPin, BiraStart, RepairBitLength, RepairOutPin, RepairBitNo, and
            //RepairPatternOut ) are not needed.
  BistIf <Serial | Parallel>; //I/F type of BIST circuit. Specify Serial or Parallel. <can be omitted>
  BiraIf <Serial | Parallel>; //I/F type of BIRA circuit. Specify Serial or Parallel. <can be omitted>
  BistFlagPin  signal_name; // Signal pin name for the BIST results determination <can be omitted>
  RepairFlagPin  signal_name; //Signal pin name that determines whether Repair is possible or not.
                                                                                        //<can be omitted>
  BistDataPin  {signal_pin_name_1; signal_pin_name_2; ··· signal_pin_name_n;} <can be omitted>
            // All signal pin names which output the BIST information should be described.
            // If BistIf is Serial, specify 1 pin, and if Parallel, specify all signal pin names.
  RepairDataPin {signal_pin_name_1; signal_pin_name_2; ··· signal_pin_name_n;} <can be omitted>
            //All signal pin names which output repair information should be described.
            //If BiraIf is Serial, specify 1 pin, and if Parallel, specify all signal pin names.
  BistStart label_name; //Label which indicates the beginning of a BIST pattern. <can be omitted>
  BiraStart label_name; //Label name that indicates the Vector location where obtaining repair data starts.
                                                              // <can be omitted>
  BistBitLength bit_width; //BIST bit width
            // Should be described in each block for Macro by Macro or Instance by Instance description.
            //Can be an integer 1 or more in decimal or hexadecimal. If in hexadecimal, use format ”¥h_number”
  BistOutPin {output_signal_pin_name_1; output_signal_pin_name_2;
            ···output_signal_pin_name_n;}     // <can be omitted>
            // Should be described in each block for Macro by Macro or Instance by Instance description.
```

//Signal pin name which outputs the BIST failure information. If Serial I/F, specify 1 pin, and if Parallel, //specify as many signal pin names as the number of the above Bit widths.

RepairBitLength bit_width; *//BIRA bit width <can be omitted>*

// Should be described in each block for Macro by Macro or Instance by Instance description.

RepairOutPin {output_signal_pin_name_1; output_signal_pin_name_2; ⋯
output_signal_pin_name_n;} <can be omitted>

// Should be described in each block for Macro by Macro or Instance by Instance description.
//Signal pin name to which repair data is output. If Serial I/F, specify 1 pin. If Parallel I/F, specify as
//many signal pin names as the number of the above Bit widths

BistBitNo{bit_number;⋯bit_number;} *//Bit number for the above BIST output signal pin.*

//Should be described in each block for Macro by Macro or Instance by Instance
//description. If Serial I/F, specify Bit numbers in the order of output. If Parallel I/F,
//specify the Bit number for the output signal pin.

BistPatternOut {label_name (Increment (step_count)/Decrement(step_count));
label_name;   ⋯label_name;}

// Specify the lable name that indicates the Vector location of BIST data output and pattern counting
//method. <can be omitted> Should be described in each block for Macro by Macro or Instance by Instance
//description. Specify the label name for the above BistBitNo. If Increment or Decrement is
//specified for the pattern counting method, no need to specify a label name for every Bit number.
//Increment and decrement can be specified only when BistIf is Serial. Not when it is Parallel.
//The step count is the step count of pattern count. Specified in integer. If omitted, the step count is
//"1". The pattern counting method can be omitted. If omitted, only the patterns of the Vector indicated
//by the label are counted. If multiple label names are described, the description of label names and
//the description of label names and pattern counting method can not be mixed.
// If Increment is specified for the pattern counting method, it goes down the pattern from the Vector
// indicated by the label, by the step count at a time, up to the number of Bit width. For instance, if the //pattern count of the Vector indicated by the label is 100, the step count is 2, and the Bit width is 4,

```
        //the pattern count goes 100, 102, 104 and 106.  If Decrement is specified for the pattern counting
        //method, it goes up the pattern from the Vector indicated by the label, by the step count at a time,
        //down to the number of Bit width. For instance, if pattern count of Vector indicated by the label is
        //100, the step count is 2, and the Bit width is 4, the pattern count goes 100, 98, 96 and 94.
RepairBitNo{Bit_number; …Bit_number;}
        //Bit number for the above Repair output signal pin. <can be omitted> Should be described in each
        //block for Macro by Macro or Instance by Instance description. If Serial I/F, specify Bit numbers in the    //order of output. If
        Parallel I/F, specify the Bit number for the output signal pin.
RepairPatternOut {label_name (Increment (step_count)/Decrement (step_count)); label_name;  …label_name;}
        //Specifies the label name that indicates the Vector location where obtaining repair data starts and pattern
        //counting method. <can be omitted> Should be described in each block for Macro by Macro or Instance by
        //Instance description. Specifies the label name for the Bit number of the above RepairBitNo. If Increment or
        //Decrement is specified for the pattern counting method, no need to specify a label name for every Bit
        //number. Increment and decrement can be specified only when BiraIf is Serial. Not when it is Parallel.
        //The step count is the step count of pattern count. Specified in integer. If omitted, the step count is “1”.
        //The pattern counting method can be omitted. If omitted, only the patterns of the Vector indicated by the
        //label are counted. If multiple label names are described, the description of label names only, and the
        //description of both label names and pattern counting method, can not be mixed. If Increment is specified for
        //the pattern counting method, it goes down the pattern from the Vector indicated by the label, by the step
        //count at a time, up to the number of Bit width. For instance, if the pattern count of the Vector indicated by
        //the label is 100, the step count is 2, and the Bit width is 4, the pattern count goes 100, 102, 104 and 106.
        //If Decrement is specified for the pattern counting method, it goes down the pattern from the Vector indicated
        //by the label, by the step count at a time, down to the number of Bit width.  For instance, if pattern count of
        //Vector indicated by the label is 100, the step count is 2, and the Bit width is 4, the pattern count goes 100,
        //98, 96 and 94.
}
```

```
<Format 2> Case of using Instance block in MemoryBist block
UserKeywords MemoryBist;
MemoryBist domain_name{
  Instance  instance_name{  //Can be omitted in each block
              //Instance information of MemoryBIst circuit and Redundancy(BIRA) circuit.
              // Should be described in Macro block for Macro by Macro description.
    BistBitLength bit_wodth;       //<Can be omitted>
    BistOutPin{output_signal_pin_name_1; output_signal_pin_name_2; ··· output_signal_pin_name_n;}
              //<Can be omitted>
    BistBitNo{bit_number; ··· bit_number;}   //<Can be omitted>
    BistPatternOut {label_name(Increment (step_count)/Decrement(step_count));label_name;
                                          ··· label_name;} //<Can be omitted>
    RepairBitLength bit_width;     //BIRA bit width <Can be omitted>
    RepairOutPin {output_signal_pin_name_1; output_signal_pin_name_2; ···
                                output_signal_pin_name_n;}      //<Can be omitted>
    RepairBitNo{bit_number ; ··· bit_number;}
                                //Bit number for the above Repair output signal pin. <Can be omitted>
    RepairPatternOut {label_name(Increment (step_count)/Decrement (step_count));
                                          label_name; ··· label_name;}      //<Can be omitted>
    }//END Instance
}
```

<Format 3> Case of using MacroName block in MemoryBist blcok

```
UserKeywords MemoryBist;
MemoryBist domain_name{
  MacroName domain_name{
        MacroNo number; //Macro number <Can be omitted>
                        //Can be an integer 0 or more in decimal or hexadecimal. If in hexadecimal, use format
                        // ”¥h_number”
        MacroType macro_type; //Type of macro (any strings)<Can be omitted>
        Instance instance_name { <Can be omitted in each block>
              //Instance information for Macro. Multiple blocks may be described.
          BistBitLength bit_width; //< Can be omitted>
          BistOutPin{output_signal_pin_name_1; output_signal_pin_name_2;           ···output_signal_pin_name_n;}  //< Can be
        omitted>
          BistBitNo{bit_number; ··· bit_number;} //<Can be omitted>
          BistPatternOut {label_name(Increment (step count)/Decrement(stepcount)); label_name;      ···label_name;} //<Can be
        omitted>
          RepairBitLength bit_width; //BIRA bit width <Can be omitted>
          RepairOutPin {output_signal_pin_name_1; output_signal_pin_name_2; ···
                output_signal_pin_name_n;} //<Can be omitted>
          RepairBitNo{bit_number; ··· bit_number;}
                //Bit number for the above Repair output signal pin. <Can be omitted>
          RepairPatternOut {label_name (Increment (step_count)/Decrement (step_count)); label_name; ···          label_name;}
          //<Can be omitted>
     }//END Instance}
   }//END Macro
}
```

◆Problem of referencing the settings of MemoryBist patterns

■IEEE Specification (additional proposal)

MemoryBist block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to refer to MemoryBist pattern settings in PatternBurst

■Proposed solution

It is recommended to enable referencing MemoryBist block names defined by UserKeywords in PatternBurst block

```
<Format>
UserKeywords MemoryBist;
PatternBurst domain_name{
  MemoryBist domain_name; //Domain name in MemoryBist block
       :      :
}
```

◆Problem of locations for BIST patterns in Pattern block

■IEEE Specification (additional proposal)

MemoryBist block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to specify the locations to begin BIST pattern of MemoryBIST pattern or vector locations for output of BIST data in Pattern block

■Proposed solution

It is recommended to define BistStart and BistPatternOut by UserKeywords and describe locations to begin BIST using BistStart statement, and vector locations to load BIST data outputs using BistPatternOut statements in Pattern block.

```
<Format>
UserKeywords BistStart BistPatternOut;
Pattern ”pattern_name_1" {
     W "WaveformTable_domain_name";
     V {“pin_group_name”=mnemonic; }
     BistStart {label_name;}
        :       :
     V {“pin_group_name”=mnemonic; }
     BistPatternOut {label_name;}
        :       :
}
```

◆Problem of locations for repair patterns in Pattern block

■IEEE Specification (additional proposal)

MemoryBist block

■Classification

Description:

Indicates the contents that should be recognized to describe or interpret the STIL data.

■Problem

Unable to specify the locations to begin repair pattern of MemoryBist pattern and vector location to load repair data in Pattern block

■Proposed solution

It is recommended to define BiraStart and RepairPatternOut by UserKeywords, and describe the locations to begin repairing using BiraStart statement, and vector locations of loading repair data using RepairPatternOut statements in Pattern block.

```
<Format>
UserKeywords BiraStart RepairPatternOut;
Pattern ”pattern_name_1" {
    W "WaveformTable_domain_name";
    V {“pin_group_name”=mnemonic; }
    BiraStart {lable_name;}
       :      :
    V {“pin_group_name”=mnemonic; }
    RepairPatternOut {lable_name;}
       :      :
}
```

◆Problem of release/capture clock definitions which should be distinguished in the Transition pattern

■IEEE Specification (Page100)

Clause 21: STIL Pattern Data

■Classification

Application:

There is no problem in describing or interpreting the STIL data, but it is the content that should develop a common recognition among the application tools and devices that handle STIL data.

■Problem

(1) STIL doesn’t have the statements which indicate its pattern attribute, therefore it cannot be simply distinguished what test pattern it was by looking at only STIL data. For example, it cannot be distinguished whether it was the Transition test pattern or not, or whether such test pattern method was ”LoC”(*1) or ”LoS”(*2).

•1 LoC: Launch On Capture/ Launch Off Capture (also called as Broadside)
* 2 LoS: Launch On Shift/ Launch Off Shift (also called as Skewed-Load)

(2) When executing fault diagnosis as Transition pattern, Launch(*)/Capture clock cannot be distinguished. Therefore there is a problem of executing fault diagnosis.

*Launch clock is also called as Release clock.

■Proposed Solution

(1) For fault diagnosis tool, it is necessary to recognize what information (attribute) ATPG output-STIL has. To distinguish such, in STIL Usage Guide 1450.0Rev4.00, it was recommended to describe with UserKeywords PatType;, but changed that recommendation, we now recommend to describe using User USER_DEFINED of Purpose statement which can be defined in Pattern or PatternBurst block statement in PatternInformation block.

```
<Format>
STIL1.0 {
  Design 2005;
  CTLMode 2005;
}
Environment (ENV_NAME) {
  (CTLMode  (CTLMODE_NAME)  {
    (PatternInformation {
      (<Pattern | PatternBurst> (pat_or_burst_name) {
        (Purpose ( User TestType < FC | DC | FCDC | AC >) // by test type                    (1)
                 ( User ScanLaunch < Shift | Capture | Mix | Any >) // Scan pattern launch method  (2)
                 ( User Load < Normal | DeCompression >)// Scan input method                  (3)
                 ( User Unload < Normal | Compression >) ; )// Scan output method             (4)
    })*//End Pattern or PatternBurst
     })//End PatternInformation
  })*//End CTLMode
}//End Environment
```

For the detail, please see the proposed Solution of “Problem of pattern attribute description such as test pattern type and test method” in the STIL Usage Guide 1450.6.

■Proposed Solution (continued from the previous page)

<Keywords explanations in the format>

(1)TestType: the keyword indicates test pattern

FC: the keyword indicates function test pattern

DC: the keyword indicates DC test pattern

FCDC: the keyword indicates mix of function test pattern and DC test pattern

AC: the keyword indicates AC test pattern

(2)ScanLaunch: the keyword describes the launch method of the test pattern of Dynamic in the ScanType.

Shift: the keyword of LoS method pattern, which launches with the transition by shift behavior.

Capture: the keyword of LoC method pattern, which launches with the transition by capture behavior.

Mix: the keyword indicates the mixture of above methods.

Any: the keyword indicates any pattern include above methods are available.

(3)Load: the keyword indicates the pattern does scan input

Normal: the keyword indicates the pattern doesn’t use the compression-scan structure.

The pattern described in STIL can directly correspond to the Scan FF one on one.

DeCompression: the keyword indicates the pattern uses the compression-scan structure.

The pattern described in STIL cannot directly correspond to the Scan FF one on one.

(4)Unload: the keyword indicates the pattern does scan output.

Normal: the keyword indicates the pattern doesn’t use the compression-scan structure.

The expectation described in STIL can directly correspond to the Scan FF one on one.

Compression: the keyword indicates the pattern uses the compression-scan structure.

The expectation described in STIL cannot directly correspond to the Scan FF one on one.

```
<Description Example1>
  STIL 1.0 {
    Design 2005;
    CTL 2005;
  }
PatternBurst BURST1 {
  PatList {
    TR1;
  }
}
PatternExec EXEC1 {
  PatternBurst  BURST1;
}
Pattern TR1{
}

Environment ENV1 {
  CTLMode MODE1 {
    PatternInformation {
       Pattern TR1 {
          Purpose   User TestType  FC
                    User ScanLaunch Shift
                    User Load  DeCompression
                    User Unload  Compression  ;
    }//End Pattern
     }//End PatternInformation
  }//End CTLMode
}//End Environment
```

Pattern TR1 indicates that is function test pattern, LOS method Transition test pattern and the pattern uses compression structure in scan input/output.

```
<Description Example2>
  STIL 1.0 {
    Design 2005;
    CTL 2005;
  }
PatternBurst TR1 {
  PatList {
    PAT1;
    PAT2;
  }
}
PatternExec EXEC1 {
  PatternBurst  TR1;
}
Pattern PAT1{
}
Pattern PAT2{
}

Environment ENV1 {
  CTLMode MODE1 {
    PatternInformation {
       PatternBurst TR1 {
          Purpose   User TestType  FC
                    User ScanLaunch Capture ;
    }//End PatternBurst
     }//End PatternInformation
  }//End CTLMode
}//End Environment
```

PatternBurst TR1 indicates the execution of function test and LOC method at Transition test.

(2) As for Launch/Capture clock, it is necessary to distinguish them. In STIL Usage Guide 1450.0Rev4.00, it was recommended to describe with UserKeywords ClockRelations; to distinguish them, however we revised that recommendation and now we recommend it to define Launch/Capture clock with using UserKeywords ClockRelations RELATION NAME in Internal block and clarify the difference of Launch/Capture clock by describing ClockRelations RELATION_NAME which is in right after the Vector sentence of Launch/Capture clock in Pattern block.

```
<Format>
 STIL 1.0 {
   Design 2005;
   CTL 2005;
 }
 UserKeywords ClockRelations;
 Environment ( ENV_NAME ) {
    CTLMode ( CTLMODE_NAME ) {
       Internal {
          (ClockRelations RELATION_NAME {
             (sigref_expr { //1 pin or more signal definition
                IsConnected In {
                   (TestAccess ( User VectorCountToLaunch DECIMAL_INTEGER (StartDelay time_exor) )
                                  ( User VectorCountToCaptue DECIMAL_INTEGER (StartDelay time_exor) )
                               < Macro | Procedure > NAME;)*
                 ( < LaunchClock | CaptureClock ( offset DECIMAL_INTEGER) > SIGNAME { //1 pin signal definition
                  <LeadingEdge | TrailingEdge>;
                  StateAfterEvent <ExpectLow | ExpectHigh > ; )
                 } )* // end LaunchClock
             }* // end IsConnected
          })*  // end sigref_expr
        })*  // end ClockRelations
     }  // end Internal
   } // end CTLMode
 } // end Environment
```

In the DECIMAL_INTEGER, count the head Vector of **Macro or Procedures block** (defining divider clock behavior) as 1 and indicate Vector number until Launch or Capture clock is in the Vector.

In the <Macro | Procedure> NAME of TestAccess, describe defining divider clock behavior Macro or the name of Procedure block.

For the Offset, count Vector of LaunchClock is 0 and indicate the Vector number until CaptureClock is in the Vector. Therefore when Launch clock and Capture clock are in 1 cycle, the Offset is 0, and this Offset value 0 should be the default.

Delay time until the start of clock after synchronizing at PLL.

sigref_expr: Signal expressions define an ordered list of signals; they are either a single token, or an expression enclosed in single quotes. Signal expression operators are plus (+), minus (-), ellipsis (..), and parentheses. These operators are not extendable. Expressions are evaluated left-to-right, with parentheses used to override this order. Signals referenced in signal expressions may occur only once in the sub-expressions generated during evaluation of the expression.

```
<Format>
 PatternBurst DOMAIN_NAME {
      ParallelPatList  SyncStart {
          PATTERN_NAME1;
          PATTERN_NAME2;
      }
  }
  Pattern PATTERN_NAME1 {
     V{                          }
      ClockRelations (RELATION_NAME)+
  }
 Pattern PATTERN_NAME2 {
    V{                           }
      ClockRelations (RELATION_NAME)+
 }
//ClockRelations indicates the prior Vector.
```

For the detail, please see the proposed Solution of “Problem of Launch/Capture clock definitions which should be distinguished in the Transition pattern” in the STIL Usage Guide 1450.6.

<Description Example 1>

```
 STIL 1.0 {
   Design 2005;
   CTL 2005;
 }
UserKeywords “ClockRelations”;
Procedures {
   “load_unload”{
      W  “TS”;
      C{ ….;}
         Shift { V{ si=#; so=#;…..;} }
  }
 }
 Pattern PAT1 {
   W  “Default”;
   Macro “test_setup”;//In the MacroDefs, define all pins default
                      // settings and testmode then call here.
   Call “load_unload” { si= …;  so=….; }
   V{                    }  // at the rising CLK3, Launch,
                         //at the falling CLK3, Capture
   Relation “TEST1”;
   V{                    }
   V{                    }
   ::
 }
```

```
Environment {
   CTLMode CTLMODE1 {//CTLMODE_NAME
     Internal {
       ClockRelations “TEST1” {
          "CLK3" {
             IsConnected In {
               LaunchClock "CLK3" {
                 LeadingEdge;
                 StateAfterEvent ExpectHigh;
               } // end LaunchClock
             } // end IsConnected
             IsConnected In {
               CaptureClock "CLK3" {
                 TraillingEdge;
                 StateAfterEvent ExpectLow;
               } // end CaptureClock
             } // end IsConnected
          } // end sigref_expr
       }  // end ClockRelations
     }  // end Internal
   } // end CTLMode
} // end Environment
```

When Launch and Capture are in 1 cycle.

◆ Problem of ScanCells' ScanCell Names description

■IEEE Specification (Page 98)

20.1 ScanStructures block syntax

■Classification

Application:

There is no problem in describing or interpreting the STIL data, but it is the content that should develop a common recognition among the application tools and devices that handle STIL data.

■Problem

Description of CELLNAME-LIST is defined as follows in STIL1450.0 ScanCells.

20.1 ScanStructures block syntax(P98)

ScanCells: Identifies the scan cells comprising the scan chain.

CELLNAME-LIST is ScanLength scan cell NAMEs separated by whitespace.

Inversion is also specified by interleaving the "!" character between scan cell NAMEs (also includes before the first scan cell and after the last scan cell). Scan cell NAMEs are ordered from the first scan cell to be shifted (input) to the last scan cell to be shifted (output).

In the ScanCells statement of the ScanStructures Block, it is not clear whether only description of the FF instance name is required, or the pin name of the FF is also required to the Scan Cell Names description.

In addition, it is impossible to identify the instance name description whether it is Verilog format, VHDL format or own format, therefore it depends on tools' implementation to decide the instance name description format.

This is causing problems in execution of the parallel loading simulation.

■Suggested Solution

About the information needed for parallel loading simulation, since it can be described in the ScanCellType block and ScanCells block in the STIL1450.1, it is recommendable to describe it in the STIL 1450.1.

It is recommended to write FF(cell) instance name in the ScanCells block instead of ScanCells statement, describe FF(cell) pin name in the scan input/output of the ScanCellType.

Also, it is recommended to describe the instance name of each FF(cell) with either Verilog format or VHDL format. It depends on the application to adapt which format.

The detail should be referred to “Problem of ScanCells definitions” in the STIL Usage Guide 1450.1.

Example)

```
STIL 1.0 { Design 2005; }
   ScanCellType FD1 {
         CellIn "MASTER" : "SIN1" ; // Define the scan in name SIN1 of the cell type
         CellOut "SLAVE" : "SOUT1" ; //Define the scan out name SOUT1
    }
  ScanCellType FD2 {
          CellIn “MASTER” : “SIN2” ; // Define the scan in name SIN2 of the cell type
          CellOut “SLAVE” : “SOUT2” ; // Define the scan out name SOUT2
    }
  ScanCells {
           "top.a1" "FD1" ;  "top.a2" "FD1" ;  "top.a3" "FD2" ;.....;
    } // “top.a1”,“top.a2” should refer to ScanCellType FD1 and "top.a3” should refer to ScanCellType FD2.
```

◆ Problem of Scan-Chain Structures

■IEEE Specification (Page 98)

20.1 ScanStructures block syntax

■Classification

Application:

There are no problems in describing or interpreting the STIL data, but it is the content that should develop a common recognition among the application tools and devices that handle STIL data.

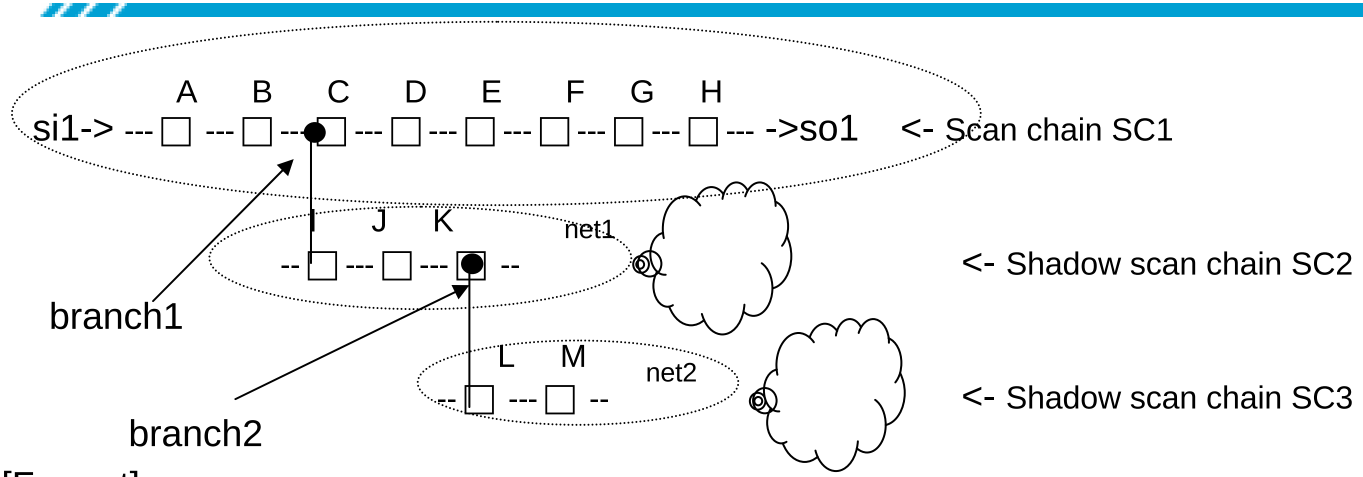
■Problem

ATPG creates the patterns for the scan-in of the shadow scan chain as well, so parallel loading is required for the shadow scan chain. However, shadow scan chain cannot be simply described with current ScanStructures.

■Proposed Solutions

Currently, there is no description that indicates the branching of ScanChain, so it is recommendable to define the branched point by UserKeywords(ScanBranch) to clarify the ScanChain structure of ScanStructures.

```
[Format]
UserKeywords ScanBranch;
ScanStructures (SCAN_NAME){
            (ScanChain CHAINNAME {
            (ScanIn SIGNALNAME;)    // Describe label name of the point that ScanBranch begins branching.
            (ScanOut SIGNALNAME;)  // If the final step of FF in the branched scan chain is open node, describe Pseudo.
            (ScanCells  (CELLNAME-LIST);)
            (ScanBranch (BRANCHNAME DECEMAL_INTEGER;)+;)
                                   // BRANCHNAME is the name for the branched point.
                                   // DECEMAL_INTEGER is the number of FFs from the Scan-In to the branched point.
            })*
}
```



[Format]

```
UserKeywords ScanBranch;
ScanStructures {
        ScanChain SC1{
            ScanIn si1;
            ScanOut so1;
            ScanCells A B C D E F G H;
            ScanBranch branch1 1;
         }
        ScanChain SC2{
            ScanIn branch1;
            ScanOut net1 Pseudo;
            ScanCells I J K;
            ScanBranch branch2 2;
         }
        ScanChain SC3{
            ScanIn branch2;
            ScanOut net2 Pseudo;
            ScanCells L M;
         }
 }
```

◆Problem of User-defined names

■IEEE Specification (Page59)

6.8 User-defined name characteristics

■Classification

Description: the contents should be recognized to describe or interpret STIL data.

■Problem

The description of User-defined name is defined as follows,

6.8 User-defined name characteristics (P59)

There are several categories of user-defined names in STIL: signal and group references, WaveformChar references, WaveformTable references, variable references, UserKeywords, labels, and domain names.

It is not clear how to handle the description not specified as User-defined names, such as ScanChain name of ScanStructures block, ScanCells name and UserFunction name.

domain names :
Domain names provide a mechanism to reference the data defined in a named block.
When a domain name is present for a SignalGroups, Procedures, MacroDefs, PatternBurst, Timing, Selector, Pattern or ScanStructures block, that domain name shall be specified in a “reference” statement in order to make use of the data in that block.

Also in P59, there is a description that the block name of Table 6 (P66) Spec, PatternExec etc. are User-defined name.

■Suggested Solution

The block name, put on the block statement becomes Named Block, is interpreted to be handled as User-defined name. Therefore the ScanChain name in the ScanStructures block is handled as User-defined name.

Also, the name put on the UserFunctions, as same as the UserKeywords, is interpreted to be handled as User-defined name.

ScanCells name in the ScanStructures block, as same as the Signals name, is interpreted to be handled as User-defined name.

Therefore User-defined name is interpreted as the collective designation of names of person creating STIL or of the names tools and designers can put.

It is interpreted the same way on the STIL extension spec (IEEE1450.1, 1450.2, 1450.3, 1450.6) where found the same problem. The detail should be refereed to the each STIL Usage Guide.

◆ Problem of BUS signal name description

■IEEE Specification (Page 60)

6.10 Signal and group name characteristics

■Classification

Description: It is the content that should develop a common recognition when describing or interpreting STIL data.

■Problem

BUS description can be described by ascending order and descending order of the Index Number. However, it is not clear how to handle the case described in the same Index Numbers.

Example: "BUSA" [0..31] //Index Number is ascending order,"BUSA"[0], "BUSA"[1], … the same as "BUSA"[31]

"BUSA" [31..0] //Index Number is descending order,"BUSA"[31], "BUSA"[30], …the same as"BUSA"[0]

"BUSA" [15..15] // Index Number is the same integer value, it is not clear how to handle this case. ,,, (*)

■Suggested Solution

For the BUS description, if the same integer value is described (*)in the Index Number, it is interpreted to show the same signal as the BUS signal of that integer value which is "signal name" (integer value).

However, it is recommended not to use the description (∗) which the same integer value is in the Index Number.

In addition, regarding the description of one Bus signal, if there is a mix of the description(∗) which the same integer value in the Index Number and the BUS signal description of that integer value, to avoid any confusion, that is fine to handle it as an error of the application side.

◆Problem of pattern repeated description scope

■IEEE Specification (Page 65)

6.15 WaveformChar characteristics(Page 65)

(cf. 21.1 Cyclized data (Page 100))

■Classification

Description: The contents should be recognized to describe or interpret STIL data

■Problems

On the list description of repeated Characters of vector flag ¥r, it is not clear if ¥r can be described or not and the interpretation for the case ¥r is described on the list description is vague.

<Format>
¥rN XXX

Example) sigref_expr=¥r3 00¥r2 11 0 01;

6.15 WaveformChar characteristics (P65)
The repeat flag ¥r is applied to a list until terminated by the end of the expression, or by the first whitespace after the ¥r and count fields.

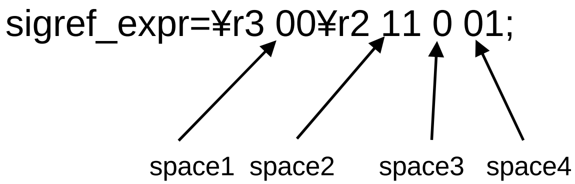
(cf. 21.1 Cyclized data (P100))
¥rN XXX: Repeat the next group of characters N times (e.g., ¥r60 01). N copies of the XXX characters are generated. The XXX characters are delimited by whitespace and may be WaveformChars, Hexadecimal characters, or Decimal characters.

■Proposed Solution

The space after the repeated times of ¥r is handled as a part of delimiter of the count fields which indicates the repeated times of ¥r. It is not handled as the first white space of ¥r repeated description and even if ¥r is described in the repeated list description, there is no contradiction. Therefore it is interpreted to be able to describe ¥r in the vector flag repeated list description of ¥r.

Example)

V { sigref_expr=¥r3 00¥r2 11 0 01; } is the same as below.
V { sigref_expr=0011110011110011110 01;}



sigref_expr=¥r3 00¥r2 11 0 01;

space1 space2 space3 space4

Space1, 2 are handled as a part of delimiter of count fields which indicates the repeated times of each ¥r3 and ¥r2, space 3 will be the first white space of both ¥r3 and ¥r2 list description.

Therefore, on the above case,
V { sigref_expr=0011110011110011110 01;} is the same as above.

■Proposed Solution (continued from the previous page)

Note1) On the STIL specification, WhiteSpace means space, tab, newline character.

Note2) On the repeated list description of ¥r, it cannot be described to switch the character types by using ¥h or ¥d. It is possible to switch the character types from hex description or decimal description by using ¥w.

```
 ¥h01 11 ¥r2 A¥wLH 0101
       00010001 1010 LH 1010 LH 0101
```

6.15 WaveformChar characteristics(P65)

Since the ¥r construct is terminated by whitespace, it is not possible to define a repeat construct around a list that specifies ¥h or ¥d options. For example, the list “¥r2 ¥hwW f0” attempts to apply the WaveformChars ‘w’and ‘W’ to the hex value f0. However, the whitespace required for ¥h also terminates the ¥r operation. This situation is resolved by putting the hex flag first as follows: “¥hwW ¥r2 f0.”

◆Problem of pattern substitution

•Proposed Solution: Complying with the STIL specification and Clarification, followings should be the behaviors.

In the SignalGroups names (sigref-exprs) substituted at Call, if there is a same name as SignalGroups names in the main body definition, substitute the pattern (argument) on the order of Vector statement of SignalGroups name matching the head side. This operation should be repeated until there is no SignalGroups name. However the substitution pattern which is at the Call of matched SignalGroups name can only be used for substituted Vector statement (# or % Vector statement) of following same name body and if it is remained, should be abandoned, and it never be used for the signal substitution of deploying response.

Next, if the substituted main body Vector statement is remained and substituted SignalGroups name at Call is remained, substitute the pattern described at responded signal of deploying SignalGroups on the order of description at Call. If there is no more main body Vector statement for substitution, the substitution operation is finished. It should be noted that if the substituted signals in the remained Signal Groups names are overlapped, since it cannot be specified which value should be substituted, it will be an error.

Even after above, if there is a signal remained in the main body substituted Vector statement, and there is a shortage of patterns to substitute at Call, the pattern specified just before the signal substituted Vector statement appeared first in the main body definition, should be substituted to the signal.

At the body definition, if there is no pattern (WFC) specified before the substituted Vector statement, and there is a shortage for substitution patterns, Procedure Call will be an error and to Macro, the pattern just before the Call will be succeeded and substituted.

•Proposed Solution (continued)

Meanwhile, the signal described at the Procedure body definition is requested to be described by head Vector statement or C statement at STIL, and if not, it will be an error. Especially if there is a substitution (# or %) or part substitution is conducted at first Vector statement, it is recommended to specify default status value by C statement right after the Procedure head W statement.

Procedure is an independent pattern block and does not influence or is not influenced by pattern of before or after the Call, however Macro is a pattern block deploys at the Call position and should be noted that Macro influence or is influenced by pattern of before or after the Call.

Therefore on the case there is a shortage of substitution pattern (argument) at the Call, it will be an error if Call of Procedure where there is no substitution pattern specified at the body definition against the substitution signal. However since the status value right before is succeeded and this status value will be substituted as implied specified pattern, it will not be an error.

However for the substitution operation including Shift block, the substitution operation to Shift block will be conducted in view of the substitution pattern number to Pre/Post pattern, it should be noted that when substitution pattern to Shift block is not remained as arguments, the Shift block behavior will be constrained.

To avoid confusion at the substitution operation, body definition and Call substitution operation are recommended to be conducted by substituting the same pin group names and the responding same signal numbers as much as possible.

■Problem and Solution Example 1: No substitution pattern (argument) case (SignalGroups)

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
Procedures {
   "proc1" {
                  W "tim_wf";
                  C { "gr2 "=¥r3 0 X; } // specified pattern
                  V1:V{ “gr2 ”=¥r4 # ; } // substitution pattern
                  V2:V { "gr1 "=¥r4 # ; } // substitution pattern
                  C { "gr2"=¥r4 1; }
                  V3:V { " gr3 "=¥r2 # ; } // substitution pattern
 }
}
Pattern "test1" {
                  W "tim_wf";
                  C {  " gr1 "  =¥r3 0 H ;  }

                  Call "proc1" {
                   " gr1 " =Z0ZL;  // substitute at Procedure body " gr1 "(V2)
                   " gr3 " =0H;   // substitute at Procedure body " gr3 "(V3)
                   }              // at “gr2 “(V1) with no substitution pattern, substitute C statement pattern specified before #
                   V4:V{" gr3 " =1L;}// can allocate C statement pattern of before Call statement as initial pattern
}
```

Solution>
If substituted Vector statement is remained at Procedure body and substitution pattern lacks at Call, substitute the pattern (WFC) specified just before the first Vector statement description which should be substituted to the signal at the Procedure body definition.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
            W "tim_wf";
            V1:V { "gr1"=000X ; }
            V2:V { "gr1"=Z0ZL ; }
            V3:V { "gr1"=110H ; }
            V4:V { "gr1"=001L ; }
}
```

■Problem and Solution Example 2: No substitution pattern (argument) case (Signals)

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
Procedure {
  "proc1" {
                    W "tim_wf";

                    V1:V {" gr1 "  =¥r4 0; }
                    V2:V {" A1 "= # ; }
                    V3:V {" A2 "= 1 ; }
                    V4:V {" A2"=  # ; }
                    V5:V {"gr3"= #  ; }
}}
Pattern "test1" {
                    W "tim_wf";
                    C { "gr1"=¥r4 X ; }

                    Call  "proc1" {
                      " gr3 "=1P; // Substitute at Procedure body " gr3 "(V4)
                    }  // For the “A1” (V1) with no substitution pattern, substitute pattern 0 of V0 specified just before #
                      // For the “A2”(V3) with no substitution pattern, substitute pattern 1 of V2 specified just before #
                    V6:V{ “ gr3 “ =1L;}// can allocate C statement pattern of before Call statement as initial pattern
}
```

Solution>
If substituted Vector statement is remained at Procedure body and substitution pattern lacks at Call, substitute the pattern (WFC) specified just before the first Vector statement description which should be substituted to the signal at the Procedure body definition.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                    W "tim_wf";
                    V1:V { "gr1"=0000 ; }
                    V2:V { "gr1"=0000 ; }
                    V3:V { "gr1"=0100 ; }
                    V4:V { "gr1"=0100 ; }
                    V5:V { "gr1"=011P ; }
                    V6:V { "gr1"=XX1L ; }
}
```

■Problem and Solution Example 3: No substitution pattern (argument) case (multipul SignalGroups)

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
Procedures {
  "proc1" {
                  W "tim_wf";
                  C { "gr2 "=¥r3 0 X; } // specified pattern V0
                  V1:V { "gr2 "=¥r4 # ; } // substitution pattern
                  V2:V { "gr1 "=¥r4 # ; } // substitution pattern
                  V3:V { "gr2 "=¥r4 # ; } // substitution pattern
                  C { "gr2"=¥r4 1; }
                  V4:V { " gr3 "=¥r2 # ; } // substitution pattern
 }
}
Pattern "test1" {
                  W "tim_wf";
                  C { "gr1"=¥r3 0 H ; }

                  Call "proc1" {
                   " gr1 " =Z0ZL;  // substitute at Procedure body " gr1 "(V2)
                   " gr3 " =1P;   // substitute at Procedure body " gr3 "(V3)
                   }            // for the " gr2 "(V1) with no substitution pattern, substitute the pattern of V0 specified before#.
                   V5:V{" gr2 " =1L;}// can allocate C statement pattern of before Call statement as initial pattern
}
```

Solution>
If substituted Vector statement is remained at Procedure body and substitution pattern lacks at Call, substitute the pattern (WFC) specified just before the first Vector statement description which should be substituted to the signal at the Procedure body definition.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                 W "tim_wf";
                 V1:V { "gr1"=000X ; }
                 V2:V { "gr1"=Z0ZL ; }
                 V3:V { "gr1"=000X ; }
                 V4:V { "gr1"=111P ; }
               V5:V { "gr1"=001L ; }
}
```

■Problem and Solution Example 4: (The case of consolidated SignalGroups, Signal)

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
Procedures {
  "proc1" {
                  W "tim_wf";
                  C { "gr2 "=¥r3 1 X; } // specified pattern
                  V1:V { "gr2 "=¥r4 # ; } // substitution pattern
                  V2:V { "gr1 "=¥r4 # ; } // substitution pattern
                  C { "gr2"=¥r4 0; }
                  V3:V { " gr3 "=¥r2 # ; } // substitution pattern
 }
}
Pattern "test1" {
                  W "tim_wf";
                  C { "gr1"=¥r3 0 X ;  }

                  Call "proc1" {
                   " gr1 " =Z0ZL;            // substitute at Procedure body " gr1 "(V2)
                 “ A1 ” =0;   “ A2” =0;  // substitute at Procedure body “ gr2 ”(V1) A1,A2. for places cannot be filled,
                                         // substitute the pattern of V1 specified before #.
                 “ gr3 ” =1P;               // substitute at Procedure body " gr3 "(V3)
                 }
                  V4:V{“ gr3 ” =1L;}// can allocate C statement pattern of before Call statement as initial pattern
}
```

Solution>
If substituted Vector statement is remained at Procedure body and substitution pattern lacks at Call, substitute the pattern (WFC) specified just before the first Vector statement description which should be substituted to the signal at the Procedure body definition.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                  W "tim_wf";
                  V1:V { "gr1"=001X ; }
                  V2:V { "gr1"=Z0ZL ; }
                  V3:V { "gr1"=001P ; }
                  V4:V { "gr1"=001L ; }
}
```

■Problem and Solution Example 5: The case of substitution pattern (argument) shortage

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
Procedures {
  "proc1" {
                  W "tim_wf";
                  C { "gr2 "=¥r3 0 X; } // specified pattern
                  V1:V { "gr2 "=¥r4 # ; } // substitution pattern
                  V2:V { "gr1 "=¥r4 # ; } // substitution pattern
                  C { "gr2"=¥r4 1; }
                  V3:V { " gr3 "=¥r2 # ; } // substitution pattern
 }
}
Pattern "test1" {
                  W "tim_wf";
                  C { "gr1"=¥r3 0 X ;   }

                  Call "proc1" {
                   “ gr1 ” =Z0ZL;  // substitute at Procedure body “ gr1 “(V2)
                   “gr2” =10;     // There are 2 substitution patterns for 4 arguments of “gr2” .  For 2 pins lack of argument,
                                 // substitute pattern V0 specified before #.
                   “ gr3 ” =1P;   // substitute at Procedure body “ gr3 “ (V3)
                  }
                  V4:V{ “ gr3 “  =1L;}// can allocate C statement pattern of before Call statement as initial pattern
}
```

Solution>
If substituted Vector statement is remained at Procedure body and substitution pattern lacks at Call, substitute the pattern (WFC) specified just before the first Vector statement description which should be substituted to the signal at the Procedure body definition.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                  W "tim_wf";
                  V1:V { "gr1"=100X ; }
                  V2:V { "gr1"=Z0ZL ; }
                  V3:V { "gr1"=111P ; }
                  V4:V { "gr1"=001L ; }
}
```

■Problem and Solution Example 6: The case of remained substitution pattern (argument)

Solution>
Substitution pattern (argument) at Call of matched SignalGroups name is only used for following same name substituted Vector statement body and if not used, is abandoned, and will not be used for the deploying responded signal substitution.

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
Procedures {
  "proc1" {
                  W "tim_wf";
                  C { "gr2 "=¥r3 0 X; } // specified pattern
                  V1:V { "gr2 "=¥r4 # ; } // substitution pattern
                  V2:V { "gr1 "=¥r4 # ; } // substitution pattern
                  C { "gr2"=¥r4 1; }
                  V3:V { " gr3 "=¥r2 # ; } // substitution pattern
 }
}
Pattern "test1" {
                  W "tim_wf";
                  C { "gr1"=¥r3 0 X ; }

                  Call "proc1" {
                   “ gr1 ” =Z0ZLZ1ZH;  // substitute at Procedure body “ gr1 ”(V2).  Remained patterns will be abandoned.
                   “ gr3 ” =1P;   // substitute at Procedure body " gr3 "(V3)
                  }            // for the " gr2 "(V1) with no substitution pattern, substitute the pattern of V0 specified before #.
                  V4:V{" gr3 " =1L;}// can allocate C statement pattern of before Call statement as initial pattern
}
```

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                  W "tim_wf";
                  V1:V { "gr1"=000X ; }
                  V2:V { "gr1"=Z0ZL ; }
                  V3:V { "gr1"=111P ; }
                  V4:V { "gr1"=001L ; }
}
```

■Problem and Solution Example 7: The case of no specified Vector before the first # (Procedure)

```
STIL 1.0;

Signals {
    "A1" InOut;
    "A2" InOut;
    "SI1" InOut { ScanIn; }
    "SO1" InOut { ScanOut; }
}
SignalGroups {
    "gr1" = '
        "A1"
        + "A2"
        + "SI1"
        + "SO1"
    ';
    "gr2" = '
        "A1"
        + "A2"
        + "SI1"
        + "SO1"
    ';
    "gr3" = '
        "SI1"
        + "SO1"
    ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
Procedures {
   "proc1" {
                W "tim_wf";
                // no specified pattern before gr2#
                V1:V { "gr2 "=¥r4 # ; } // substitution pattern
                V2:V { "gr1 "=¥r4 # ; } // substitution pattern
                C { "gr2"=¥r4 X; }
                V3:V { " gr3 "=¥r2 # ; } // substitution pattern
    }
}
Pattern "test1" {
                W "tim_wf";
                C { "gr1"=¥r4 X ;  }

                Call "proc1" {
                 "gr1 "=Z0ZL;    // substitute at V2
                 "gr3 "=1P; // substitute at V3
                  } // there is no specified pattern just before # of “gr2”, substitution pattern is not clear, therefore it is an error
                 V4:V{“ gr3 “ =1L;}// can allocate C statement pattern of before Call statement as initial pattern
}
```

Solution>
If there is no specified pattern just before the substituted Vector statement at the body definition and the substitution pattern lacks, it should be an error to Procedure.

Error

■Problem and Solution Example 8: The case of no specified Vector before the first # (Macro)

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
MacroDefs {
  " Macro1 " {
                  W "tim_wf";
                   // there is no specified pattern before " gr2 "# of V1
                  V1:V { "gr2 "=¥r4 # ; } // substitution pattern
                  V2:V { "gr1 "=¥r4 # ; } // substitution pattern
                  C { "gr2"=¥r4 1; }
                  V3:V { " gr3 "=¥r2 # ; } // substitution pattern
  }
}
Pattern "test1" {
                  W "tim_wf";
                  C { "gr1"=¥r3 0 X ;  }

                  Macro " Macro1 " {
                   "gr1 "=Z0ZL;    // substitute at V2
                   "gr3 "=1P;  // substitute at V3
                  }                // no substitution pattern "gr2 “’s
                                   // at V1, substitute the pattern “ gr2 ” =¥r4 X ; just before the Call
                   V4:V{ “gr3 “ =1L;} // for the shortage patterns, allocate the final pattern of Macro statement
}
```

Solution>
If there is no specified pattern (WFC) just before the substituted Vector statement at body definition and substitution pattern lacks, to Macro, substitute the succeeding pattern (WFC) just before the Call.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                  W "tim_wf";
                  V1:V { "gr1"=000X ; }
                  V2:V { "gr1"=Z0ZL ; }
                  V3:V { "gr1"=111P ; }
                  V4:V { "gr1"=111L ; }
}
```

■Problem and Solution Example 9: SignalGroups deployment

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr3" = '
      "SI1"
      + "SO1"
   ';
  "gr4" = '
      " A1"
      + " A2"
   ';
    "gr5" = '
      " A2"
      + " A1"
   ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
MacroDefs {
  “Macro1" {
                W "tim_wf";
                V1:V { "gr2 "=¥r4   # ; }    // substitution pattern
                V2:V { "gr1 "=¥r4   # ; }    // substitution pattern
                V3:V {" gr2 "=¥r4   #;  }     // substitution pattern
                V4:V { " gr3"=¥r2   # ; }    // substitution pattern
                V5:V{ “ gr4”=¥r2   # ; }     // substitution pattern (remained substitution pattern)
}}
Pattern "test1" {
                W "tim_wf";
                C { "gr1"=¥r3 0 X ;  }
                Macro " Macro1 " {
                 "gr2 " =00001111;    // substitute 0000 to V1, substitute111 to V3
                 "gr1 " =Z0ZL;    // substitute to V2
                 "gr3 "  =1P;  // substitute to V4
                 "gr5 "  =1P; //  no argument of “ gr4 ”, substitute “gr5” pattern to V5
                 }
                V6:V{“gr3 ” =1L;} // for the shortage patterns, allocate the final pattern of Macro statement
}
```

Solution>
If there is a same SignalGroups name of body definition in the SignalGroups name substituted at Call, substitute the pattern from Vector statement of head matched SignalGroups name in order.
This operation should be repeated until there is no SignalGroups name.
Next, at Call, if substituted SignalGroups name is remained, in order of description at Call, substitute the pattern described at responding signal deploying SignalGroups.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                W "tim_wf";
                V1:V { "gr1"=0000 ; }
                V2:V { "gr1"=Z0ZL ; }
                V3:V { "gr1"=1111 ; }
                V4:V { "gr1"=111P ; }
               V5:V { "gr1"=P11P ; }
                V6:V { "gr1"=P11L ; }
 }
```

■Problem and Solution Example 10: SignalGroups deployment

```
STIL 1.0;

Signals {
    "A1" InOut;
    "A2" InOut;
    "SI1" InOut { ScanIn; }
    "SO1" InOut { ScanOut; }
}
SignalGroups {
    "gr1" = '
        "A1"
        + "A2"
        + "SI1"
        + "SO1"
    ';
    "gr2" = '
        "A1"
        + "A2"
        + "SI1"
        + "SO1"
    ';
    "gr3" = '
        "SI1"
        + "SO1"
    ';
  "gr4" = '
        " A1"
        + " A2"
    ';
      "gr5" = '
        " A2"
        + " A1"
    ';
}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
MacroDefs {
   " Macro1 " {
                           W "tim_wf";
                           V1:V { "gr2"=¥r4   # ; }    // substitution pattern
                           V2:V { "gr1"=¥r4   # ; }    // substitution pattern
                           V3:V { “gr2”=¥r4   #;  }     // substitution pattern (substitute A2 data1 of gr5 remained at substitution
                                                                                                      pattern to A2)
                           V4:V { "gr3"=¥r2   # ; }    // substitution pattern
                           V5:V { “gr4”=¥r2   # ; }     // substitution pattern (substitute A1 dataP of gr5 remained at substitution
                                                                                                      pattern to A1)
   }                                                   // for the V3 and V5 shortage patterns, from the data of C statement just
}                                                      // before Macro, for SI1, S01 of V3, 0x and for A2 of V5, 0 will be substituted.
Pattern "test1" {
                           W "tim_wf";
                           C { "gr1"=¥r3 0 X ;  }
                           Macro " Macro1 " {
                            "gr2 " =00001;    // substitute 0000 to V1, substitute1 to “A!” of V3
                            "gr1 " =Z0ZL;    // substitute to V2
                            "gr3 "  =1P;  // substitute to V4
                            “gr5 ”  =1P; //  no argument of “ gr4 ”.  Substitute 1P to each “A2” of V3 and “A1” of V5.
                            }                  //  as a shortage pattern, substitute specified X
                            V6:V{“gr3 ” =1L;} // for the shortage pattern, allocate a final pattern of Macro statement
}
```

Solution>
If there is a same SignalGroups name of body definition in the SignalGroups name substituted at Call, substitute the pattern from Vector statement of head matched SignalGroups name in order.
This operation should be repeated until there is no SignalGroups name.
Next, at Call, if substituted SignalGroups name is remained, in order of description at Call, substitute the pattern described at responding signal deploying SignalGroups.

Above example, Pattern block is equal to the description below.

```
Pattern "test1" {
                 W "tim_wf";
                 V1:V { "gr1"=0000 ; }
                 V2:V { "gr1"=Z0ZL ; }
                 V3:V { "gr1"=110X ; }
                 V4:V { "gr1"=111P ; }
                 V5:V { "gr1"=P01P ; }
                 V6:V { "gr1"=P01L ; }
 }
```

■Problem and Solution Example 11: The case the substitution pattern is not identifiable

```
STIL 1.0;

Signals {
   "A1" InOut;
   "A2" InOut;
   "SI1" InOut { ScanIn; }
   "SO1" InOut { ScanOut; }
}
SignalGroups {
   "gr1" = '
      "A1"
      + "A2"
      + "SI1"
      + "SO1"
   ';
   "gr2" = '
      "A1"
   ';
     "gr3" = '
      "A1"
   ';
   "gr4" = '
      "A1"
   ';
  "gr5" = '
      " A2"
      + " SI1 "
  + " SI2 "
   ';

}
```

```
Timing "tim1" {
}
PatternBurst "test1" {
}
PatternExec {
}
MacroDefs {
 " Macro1" {
                W "tim_wf";
                V1:V { "gr1"=¥r4  # ; }     // substitution pattern
                V2:V { "gr2"=# ; }          // substitution pattern


 }
}
Pattern "test1" {
                W "tim_wf";
                C { "gr1"=¥r3 0 X ;  }
                Macro " Macro1" {
                 "gr3"=0;      // SignalGroups of "gr3" and "gr4 " reference "A1" signal
                 "gr4"=1;      //as a substitution pattern to "A1" of V1, it cannot be decided whether to put "gr3" or "gr4 "
                "gr5"=111;
                 }
                V3:V{"gr3" =1L;} // for the shortage patterns, allocate the final pattern of Macro statement
}
```

Solution>
At the Call of Prodedure or Macro, several SignalGroups cannot reference 1 signal. Put as an error.

Error

◆ Problem of comments

■IEEE Specifications (Page 58)

6.5 Comments （Page 58）

■Classification

Description: Indicates that those are contents which should be commonly understood to describe or analyze the STIL data.

■Problem

For comments, there are two types of description forms, //line comment and /*block comment*/. It is prescribed that comments in both forms can be described to any blank space and recognized as space (whitespace) , however the concrete treatment has been undefined.

Ex 1)

```
/* Comment */
(linefeed)
STIL 1.0;
  Header  {
      :   :
```

Ex 2)

```
STIL 1.0;
    :     :
Pattern “PAT1”{
  W/*comment1*/”TS1”
  V  {ALLPAT=0100HL;}
    Loop/*comment2*/2{
      V  {ALLPAT=0100LL;}
    }
}
```

Ex 3)

```
STIL 1.0;
    :     :
Pattern “PAT1”{
  W ”TS1”
  V  {ALLPAT=0100HL;}
  V  {ALLPAT=0100/* comment3 */LL;}
  V  {ALLPAT=0100LL;}
}
```

■Solution proposed

It is interpreted that the two types of comment description, //line comment and /*block comment*/, can be described in any place in STIL file. Handling these description as space (whitespace) is not particularly prescribed. It depends on each application.

It is recommended that //line comment and /*block comment*/ are interpreted as space of one character.

IEEE 1450.0 5.1.1 STIL grammatical constructs

There are two types of STIL statements, Simple Statement and Block Statement.

Simple Statement

Keyword **(OPTIONAL_TOKENS)*;**

Block Statement

Keyword **(OPTIONAL_TOKENS)* { (OPTIONAL_MORE_STATEMENTS)* }**

Therefore, //line comment and /*block comment*/ are not STIL statements.
They are treated as whitespaces, and can be described anywhere whitespace can be described.
"Whitespace" indicates a string that consists of one or more space character, tab character (¥t), or newline character.

IEEE 1450.0 6.5 Comments

There are two styles of comment in STIL:

// line comment	line comments are terminated by newline
/* block comment */	block comments may span multiple lines

Comments may appear at any legal whitespace location and are treated as whitespace.
Nested block comments shall not be allowed (e.g., "/* /* */ */"), but line comments may be contained in block comments. Comments defined using these constructs may not be preserved through STIL processes.
See Clause 13 for annotations, which are a type of comment that is preserved through processes.

[Description Example]

Ex 1)

```
/*  Comment  */
(linefeed)
STIL 1.0;
  Header  {
       :   :
```

Ex 1: //line comment and /*block comment*/ are not STIL statements. They can be described before STIL statement as shown left.

The comments are described in STIL1450.0 STIL statement (Page70).

8. STIL statement

The STIL statement shall be the first statement of a STIL file.
The version number refers to the revision of STIL that the writer is designed to support.

Ex 2)

```
STIL 1.0;
    :     :
Pattern  "PAT1"{
  W/*comment1*/"TS1"
  V  {ALLPAT=0100HL;}
    Loop/*comment2*/2{
      V  {ALLPAT=0100LL;}
    }
}
```

Ex 2): //line comment and /*block comment*/ are understood as space characters. After "W" and "Loop," whitespace is assumed to be described, and this is correct form of STIL.

Ex 3)

```
STIL 1.0;
    :     :
Pattern  "PAT1"{
  W "TS1"
  V  {ALLPAT=0100 HL;}
  V  {ALLPAT=0100/* comment3 */LL;}
  V  {ALLPAT=0100 LL;}
}
```

Ex 3: //line comment and /*block comment*/ are understood as space characters.

As shown in the example 2, the comment is treated as a whitespace. Whether or not to be the objective of Loop compression depends on each application.

- **Notes**

In STIL Usage Guide, using whitespace in user definition names is prohibited. However, there is the STIL data that has the user definition name that includes whitespace. When a comment string is described instead of whitespace to the user definition name that includes whitespace, it is not treated as a comment string, but another definition name.

Example) “abc d” and ”abc/* comment */d” are not the same definition name.

- **Location to describe Annotation**

The description of Annotation is defined in STIL1450.0 Annotation.

> 13. Ann statement (P74)
>
> Annotations are text strings that are maintained through a STIL process as appropriate for that process. (In particular, a STIL output shall contain any annotations that were present in the input.) Annotations may contain any desired user information or comments. The Ann statement may occur any place a STIL statement may occur after the initial STIL (version) statement. It may occur as a top-level statement or inside STIL block statements.
>
> The Ann statement uses two-character delimiters to identify an annotation block. The Ann statement block starts after the token “{*” and is terminated by the token “*}”. These delimiters shall be separated with whitespace from the annotation text.
>
> 13.1 Annotations statement syntax
>
> Ann {* ANNOTATION *}
>
> Ann: Keyword for the annotation statement.
>
> ANNOTATION : Text string of annotation.

The description of Annotation is positioned as the statement of STIL definition. Annotation can be described anywhere the statement can be described, but it cannot be described before the STIL statement (leading of the file).