6.8 Additions to timing expressions - time_expr

Additional symbols are added for referencing timing events in waveforms. These are additions to the @n symbol as defined in STIL.0 (subclause 6.13) which allows reference to the timing edges in a timing expression that are in the current period. The new symbols, as defined below (along with the other time referencing capabilities), allow access to period markers and events in subsequent periods.

a) @ => references the time of the prior event (STIL.0)

b) @n => references the time of the n-th event, where first event is @1 (STIL.0)

c) @@ => references the time of the current event (STIL.1)

d) @Tm => reference the time of the start of the m-th period, where the current period is numbered 0. The time value returned is relative to T0 of the current period. @T1 represents the end of the current period (i.e., the time relative to the current period for the start of the second period, called P1).

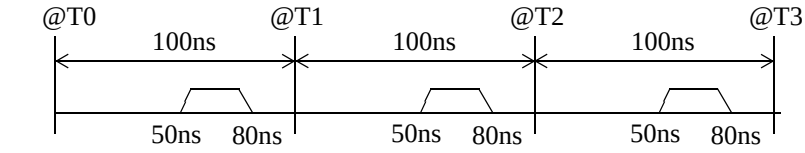
e) @Tm.n => reference the events of subsequent periods. The time value returned is relative to T0 of the current period. @T1.3 represents the third event of the second period. Note that events are numbered starting at 1, since the 0'th event is T0. (STIL.1)

f) EVENT_LABEL => references a label that is defined by EVENT_LABEL: anywhere in the current WaveformTable or WaveformDescriptions bloc (STIL.0)

g) SPEC_VARIABLE => references a variable defined in a Spec block (STIL.0)

h) VARIABLE => references an Integer or IntegerConstant defined in a Variables block (STIL.1).

The following diagram illustrates events across 3 periods.



@T0 = 0ns	@T1 = 100ns	@T2 = 200ns	@T3 = 300ns
@1 = 50ns	@T1.1 = 150ns	@T2.1 = 250ns	
@2 = 80ns	@T1.2 = 180ns	@T2.2 = 280ns	

Figure 1—Referencing timing edges

Example of clause 6.8 usage (not part of doc)

The following code example shows a usage of a timing expression using subsequent periods (cycles). Note that the concept of multiple periods (cycles) is only valid in the context of a pattern, macro, or procedure. This example uses the period timing of the two vectors within a loop to determine the loop count. Note: Other uses for this facility such as to specify timing edge rules are defined in the STIL extensions that use them.

```
1: STIL 1.0 { Design D19; }
2: Header {
3:      Source "IEEE P1450.1/D20 - Aug 26, 2004";
4:      Ann {* subclause 6.8 *}
5: }
6: Signals { ALLSIGS[1..100] In;
7: Timing {
8:      WaveformTable WFT1 { Period 75ns; }
9:      WaveformTable WFT2 { Period 500ns; }
10: }
11: Pattern PAT {
12:     C { }
13:     Loop '(10us/@T2)+1' {     // loop 18 time; trunc to int implied
14:       W WFT1; V { }    // period is 75ns
15:       W WFT2; V { }    // period is 500ns
16:     }
17: }
```

22. PatternFailReport

The PatternFailReport block is used to contain fail information resulting from the test of a device. As such, it is typically not part of a STIL test program file/stream, but contains references to the associated STIL test program file/stream to establish the context. Each PatternFailReport block shall contain the set of fails from the test of one device on the tester.

See clause 19 for the definition of the X (cross reference) pattern statement which is the basis for referencing fail data back to the pattern data.

22.1 PatternFailReport syntax

```
PatternFailReport (REPORT_NAME) {                                    (1)
    ( DeviceID "IDENTIFYING INFORMATION"; )                          (2)
    ( TestConditions { } )                                           (3)
    ( Pattern PAT_NAME; )                                            (4)
    ( Pattern PAT_NAME {
        ( LogStart (X_REF_ID) (CYCLE_COUNT) ; )
        ( LogStop (X_REF_ID) (CYCLE_COUNT) ; )
    } )
    ( PatternBurst PAT_BURST_NAME; )                                 (5)
    ( PatternBurst PAT_BURST_NAME {
        ( LogStart (X_REF_ID) (CYCLE_COUNT) ; )
        ( LogStop (X_REF_ID) (CYCLE_COUNT) ; )
    }
    ( PatternExec PAT_EXEC_NAME; )                                   (6)
    FailData {                                                       (7)
        ( X_REF_ID ( . X_REF_ID )*                                   (8)
            SIG_NAME
            <
                | ( CYCLE_OFFSET )* ;
                | 'observed_data' ;
                | ( CYCLE_OFFSET 'observed_data' )* ;
            >
         )*  // end fail data record
    }  // end FailData
}  // end PatternFailReport
```

(1) **PatternFailReport**: The block contains fail data as produced on a tester and is associated with a STIL test program stream.

(2) **DeviceID**: This statement contains a string for the purpose of identifying the device-under-test that produced the reported results.

(3) **TestConditions** { }: This block contains statatements that identify the conditions under which the test was made. The test conditions would typically identify things such as: environmental conditions like temperature; DC set up conditions like voltage and current; timing set up conditions; ATE type and system identification. These statements shall either be Ann statements, or else user keyword statements.

(4) **Pattern**: This statement specifies the name of the pattern in the associated STIL test program file/stream that is detecting the device failures. The block form of this statement allows for the specification of start and stop points of the logging information in this fail report. The X_REF_ID parameter refers to a X statement in the pattern. The CYCLE_COUNT parameter represents the number of vector cycles (i.e., periods) that transpire to the commencement or termination of the logging function. When only the CYCLE_COUNT is present, the cycle offset is from the beginning of the pattern. If both X_REF_ID and cycle count are present the cycle offset is from the specified X_REF_ID.

(5) **PatternBurst**: This statement specifies the name of the pattern burst in the associated STIL test program file/stream that is detecting the device failures. The block form of this statement allows for the specification of start and stop points. See the definition of these parameters in the Pattern statement, above.

(6) **PatternExec**: This statement specifies the name of the pattern exec in the associated STIL test program stream that is detecting the device failures.

(7) **FailData**: This begins the block containing the device failure data.

(8) **fail data record**: Each device failure record contains the fail data information as described below.

a) X_REF_ID: The first token of each record is used to identify the reference position within the pattern. It may be either a user defined name (according to the rules of STIL.0 subclause 6.8) or an integer. Reference shall always be to the last X statement encountered in the pattern execution sequence within the base pattern. If there are X references within called procedures and/or macros then the X_REF_ID is appended with dot separators to the X_REF_ID in the base pattern. For patterns with no X statements, the pattern name itself can be used as an X_REF_ID to specify the offset if from cycle 0 of the pattern.

b) SIG_NAME: This is the name of the signal that detected the failure. It may be either a name in a Signals block or a re-named single signal in a SignalGroups block. The signal name resolution follows the standard rules for domain name resolution. The signal names always refer to the names as used in the referenced pattern, however, a Signals/SignalGroups block may be included in the fail data file/stream for the purpose of defining the hex formatting of the fail data records.

c) CYCLE_OFFSET: This token indicates the cycle offset from the last the last X statement encountered in the pattern execution sequence. This is an optional field, and if not specified, a cycle offset of 0 is assumed. The cycle number for the labeled vector shall be 0. If the last last X statement encountered in the pattern execution sequence is within a Loop or Shift block, then the cycle offset is from the first occurence of the X reference in the loop or shift.

d) *observed_data*: This field is optional and specifies the observed state(s) of the failing signal. The observed data is enclosed in single quotes. In full data capture mode this is a string of observed states - with a state specified for each cycle (either H/L/T for failing cycles, or X for cycles for which no compare is done). Compaction of this observed data is supported by expressing the the data using the \h, \r, and \e constructs as defined in STIL.0 Clause 21.1.

If there is no Signals or SignalGroups blocks in the fail report file/stream then, by default, the observed_data is an event list comprised of the compare event characters L, H, X, and T (low, high, don’t-care, and tri-state) as defined in STIL.0 Clause 18.2 Table 10.
If there is a Signals or SignalsGroups block in the fail report that defines the observed signal name, then the "Base Hex;" statement can be used to specify hex formatting as the default. Note that the more compact hex logging (with 4 cycles represented by each hex character) can be done if only L and H output events are possible.

22.2 PatternFailReport example

The following are examples of syntax constructs. See annex N for a complete example of pattern fail report statements.

```
18: STIL 1.0 { Design D19; }
19: Header {
20:     Source "IEEE P1450.1/D20 - Aug 26, 2004";
21:     Ann {* subclause 22.2 *}
22: }
23:
24: Signals {
25:     PO1 Out;
26:     PO2 Out;
27:     SO1 Out; SO2 Out; SO3 Out; SO4 Out;
28:     SO5 Out { Base Hex LHT; }
29: }
30:
```

```
PatternFailReport {
    Pattern PAT;
    PatternBurst BRST;
    PatternExec EXE;
    FailData {
      13 PO1; // example with pattern-unit=13; signal=PO1
      PU14 PO1; // example with a pattern-unit identifier
    42 SO1 19; // example with cycle offset (i.e. scan cell in a shift)
      45.LU SO1 16; // example with reference inside a macro/proc (i.e the LoadUnload)
      46.M1.LU S01 16; // example of two levels of macro/proc
    59 PO2 23'H'; // example of fail state
      64 SO2 6 16 24 25 338; // example of multiple fails (5 in this case)
      72 SO3 13'H' 19'L' 45'L' 63'L'; // example of multiple fails with fail states
      88 S04 0'HHHLLLLHHH' 56'HHHL';  // example of data capture format
      Pattern PO1 19023'H'; // example referencing cycle 19023 from the beginning of the pattern
      "ABIST-30" SO5 22 '52 \r20 00 A \e L'; // @ cycle 22: HHLT ...80 L's... TTL
      } // end FailData
} // end PatternFailReport
```