Title: STIL 1450.1 Internal Resolution of 1450.1-D14

History:

- 04/02/2002 - issues carried over from D12
- 04/04/2002 - added GR issues.
- 04/11/2002 - updates from wg meeting
- 04/25/2002 - updates from wg meeting
- 05/23/2002 - updates from wg meeting
- 06/05/2002 - added some new issues - GR-22, GM-6, GM-7, GM-8
- 06/12/2002 - updates from wg meeting on 6/6
- 07/16/2002 - comments added on changes made in the document by tt
- 07/28/2002 - updates from wg meeting
- 08/01/2002 - updates from wg meeting
- 08/15/2002 - updates from wg meeting

The working document “P1450.1 Draft 14 is now undergoing review and update. The following are the individuals participating in the review:

1. (BC) billc@ims.com
2. (BR) bennyr@taux01.nsc.com
3. (CW) cww@ee.nthu.edu.tw
4. (DD) dave_dowding@agilent.com
5. (DK) david_kellerman@teseda.com
6. (DM) denism@synopsys.com
7. (DS) dsprague@btv.ibm.com
8. (GM) gmauston@qwest.net
9. (GR) gordon_robinson@3mts.com
10. (GW) gwilder@dal.asp.ti.com
11. (HR) hira_ranga@3mts.com
12. (IS) isikawa@eeclkg.eec.toshiba.co.jp
13. (JD) jason_doege@inovys.com
14. (JO) jim_oreilly@agilent.com
15. (JT) jim@galois.com
16. (KD) klaus-dieter_hilliges@agilent.com
17. (LM) larry.moran@teradyne.com
18. (PW) wohl@synopsys.com
### Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| DM-1  | Editorial - 15.2 | 4/25 - following decisions by wg:  
1. \m #; or \m##; is allowed, space optional  
2. \m 010 101; the first three wfc’s are mapped, up to the delimiting space character.  
3. \j\m is allowed, in which case the H is first mapped and then joined according to the WFCMap specification.  
4. \j\m\j is specifically NOT allowed as it could cause conflict.  
*sec 15.2 updated by tt on 6/17. Needs wg review* |
|       | The definition is missing with regard to \m and \j syntax in pattern data. For example consider to following cases:  
\m ##; <= is space required, not allowed, optional?  
\m 010 101; <= does mapping extend to the space or the ; ?  
\j\m H; <= is this allowed? |
| DM-2  | Technical - 15.2 | 4/25 - Decision by wg to be explicit and require the \j to be specified in the pattern data. No change required to the document. |
|       | Suggest removing the \j altogether. The join function would still be allowed, but changed as follows: If a signal definition appears multiple times within a pattern data block (i.e., V, C, etc.) and there exists a WFCMap for the signal-wfc, then the join is resolved, else error. |
### Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| DM-3  | Editorial - 12.3, Pattern Tiling and use of Wait/Extend  
1. A better explanation of the application rules for the Wait and Extend statements is needed. Probably should move Annex K into this section as way of explanation.  
2. The syntax in Annex K differs from clause 12.3. In clause 12.3 the Wait appears on the pattern. In Annex K it is after the PatternBurst. Which way should it be?  
3. Suggest the following semantic for Wait: “If there is no Wait statement following a ParallelPatternBurst, then all patterns or bursts shall complete before continuing. If there is a Wait following a ParallelPatternBurst, then processing shall continue as soon as all the patterns and bursts listed in the Wait block have completed.” | 4/25 -  
1. wg agreed that Annex K should be moved to the syntax example section.  
2. wg tended towards the form where the Wait and Extend are attributes of the pattern, rather than free-floating statements. However, wg decided to wait for more input from Jason on concurrent pattern execution.  
7/16 - sec 12.4 updated and Annex K removed by tt.  
7/18 - wg would like feedback from JD wrt his proposal on pattern merging. |
| DM-4  | 4/22/02 - Technical - 16.1  
The ScanStructures statement should be used as a domain definition that is used to specify the environment for a pattern. As such it should not be allowed as a pattern statement. | 4/25 - Peter is the primary proponent of this syntax. He will discuss this with Denis during the week of 5/6 when he is in sunnyvale. |
| DM-5  | 4/22/02 - Editorial - 12.4, If & While  
Need to define the semantics of statements in the run time environment. For example, what is the scope of “If ‘xxx’ { Fixed xxx; }” only within the braced block or everything after the block when xxx is true? | 4/5 - wg decided that the scoping of the Fixed statements does not change due to run time execution. i.e., a Fixed within a conditional block has effect up until the end of the pattern, if the condition evaluates to TRUE.  
7/16 - doc updated by tt |
| DS-1  | Editorial - Pg 11, Section 6.1, last paragraph.  
This paragraph is very confusing and tough to get a grip on. We believe it is because of the over usage of the word "enumeration" within that paragraph. For example, the sentence starting with "SignalVariableEnum and IntegerEnum variables shall..." is a real mouthful. | 01/03/2002 - Greg to rewrite this paragraph. |
<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| DS-2  | Editorial - Pg 12, Section 6.6  
A general statement about the confusion of the relationship between expressions. For instance, it appears that all boolean expressions, integer expressions, and real expressions are all logical expressions. Would it be helpful to have some kind of hierarchical tree or BNF type diagram to show this relationship. If this makes sense, maybe this would be best placed up front prior to going into expressions. Like maybe as section 6.2. Just a thought to maybe help clear up the explanation of expressions. | 01/03/2002 - Doug to rewrite this paragraph. |
| DS-3  | Editorial - Pg 35 Section 16.4.  
We still need to address the issue of X statements within procedures and macros and how fail data using X statements can uniquely identify references back to these points in the patterns. I suggested in an email putting verbage into this section to enforce a hierarchy across X statements from the mainline patterns down through the macros/procedures. Never got any replies so not sure how people feel about this. Need to discuss this at the next meeting. | The suggestion from Doug is included into section 16.4. See also GR-14.  
Needs WG review and approval. |
| DS-4  | Editorial - Pg 39, Section 18.1, next to last paragraph starting with "(12) Type...".  
It say the file type shall be "one of the specified types" but no types are specified anywhere. Also it says you can specify User followed by the type name but there is no provision for this in the syntax description on pg 38. I guess you could argue this second point is actually OK because it's covered in this definition of of file_type but it seems it would be clearer if the syntax said something like: Type (User) type_name ; | 01/03/2002 - Greg will distribute the list for group to review with the idea of including it in the “recommended usage document” on the STIL website. |
Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| GM-1  | Section 14 top of pg 31, the syntax:  
( ScanCells {
  (< cellname_list | cellname_list {

  is wrong, because it implies the interpretation of the all cells before the { } are part of the { }, e.g.:  
  
  ScanCells { a b c { CellIn hoozy; } } //hoozy is on cell 'c' only and not 'a' and 'b'  

  It's also the wrong syntax for the block construct. We have two forms of ScanCell statements. The first is the single-statement form defined in dot-zero, seen in the first example in 14.2 pg 33, and it uses cellname list:  
  
  ScanCells a b c;

  The second form is seen in the example in 14.3, which is the block-statement form. This form actually DOES REQUIRE STATEMENT DELIMI ERS BETWEEN EACH CELL NAME (as shown in that example):  
  
  ScanCells { a; b; c{} }  

  This form does NOT use the cellname-list (as currently defined). THEREFORE, I request the syntax description above change to:  

  ( ScanCells {
    ( cellname-ref ; )*  
    ( cellname-ref { ... } )*  

  ) )  

  We might need to define cellname-ref, possibly instead of cellname-list, to accomplish this. | 5/23/02 - PW to review and comment on GM-1, GM-2, GM-5. |
**Table 1: Summary of Issues with 1450.1-D13**

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| GM-2  | Section 14  
Definition of STATE-ELEMENT. It has been requested that each
STATE-ELEMENT item be defined in the same namespace as the
cellnames. The current definition states "State elements are internal
design nets of CELLNAME." The problem with this is that in my expe-
nience, shadow registers often appear at the same design-level as the
scan register itself, and are NOT internal elements of that cell, but I
would like to tag the presence of shadow registers with CellIn and Cell-
Out. I think the current definition is too limiting, and I propose chang-
ing the quoted sentence above to: "State element names are placed in
the same namespace of all scan cells." This does not disallow state-ele-
ments to be internal elements of cellname, but requires that the ele-
ments be 'uniqified', most commonly by placing the cellname as part
of the state-element name, e.g.,
   ScanCells { "a/b" { CellIn "a/b/c"; }} | 5/23/02 - PW to review and comment on GM-1, GM-2, GM-3. |
| GM-3  | section (6), the If (boolean_expr) definition. The sentence "The value of
boolean_expr is evaluated during pattern operation" has been ques-
tioned, as being too limiting. This is one context where the value may
be evaluated, but I don’t think it’s necessary to limit it to this context...
so I propose: "The value of boolean_expr is evaluated as necessary by
the application, for example it may be evaluated during sequencing
through Vectors during Pattern operation, to establish a value if neces-
| GM-4  | CellOut definition, pg 34. The phrase "is to be unloaded" caused some
confusion. I would recommend the wording "is to be unloaded into the
cellname-ref when the boolean_expr is true...". Is the boolean_expr
actually optional for the CellOut statement -what’s it mean if it’s not
present? That the cell’s state is always overwritten by the state of the
CellOut reference? | 7/16/2002- doc updated by TT |
This is the most complicated one (maybe?). A single cellname-ref may identify multiple elements, e.g., "mbr[0..7]". In support of the fact that we allow this construct now, I request a uniform extension of the STATE-ELEMENT references in both CellIn and CellOut statements to both support STATE-ELEMENT-LISTS, with the following additional elaboration:

If the cellname-ref is a single item, then each CellOut statement present shall specify only a single STATE-ELEMENT (the STATE-ELEMENT-LIST shall contain one reference). If the cellname-ref contains a reference to multiple elements, then there shall be a one-to-one correspondence in declaration order, of STATE-ELEMENTS in all CellIn and CellOut statements to individual cellnames referenced.

I don’t know if we need an example of what this is trying to support, but here it is:

```c
// In the scan chain definition of the ScanCells.
c[0..4] {
    If clk_p CellIn u[0..4];
}
```

where the dependent state element u[0] corresponds to scan element c[0], etc...

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| GM-5  | This is the most complicated one (maybe?). A single cellname-ref may identify multiple elements, e.g., "mbr[0..7]". In support of the fact that we allow this construct now, I request a uniform extension of the STATE-ELEMENT references in both CellIn and CellOut statements to both support STATE-ELEMENT-LISTS, with the following additional elaboration: If the cellname-ref is a single item, then each CellOut statement present shall specify only a single STATE-ELEMENT (the STATE-ELEMENT-LIST shall contain one reference). If the cellname-ref contains a reference to multiple elements, then there shall be a one-to-one correspondence in declaration order, of STATE-ELEMENTS in all CellIn and CellOut statements to individual cellnames referenced. I don’t know if we need an example of what this is trying to support, but here it is:  
// In the scan chain definition of the ScanCells.  
c[0..4] {  
    If clk_p CellIn u[0..4];  
}  
where the dependent state element u[0] corresponds to scan element c[0], etc... | 5/23/02 - PW to review and comment on GM-1, GM-2, GM-5. |
| GM-6  | Additional semantic definition for ParallelPatList Lockstep email 6/3/02. Specifically in support of cores or modules that have connected scan chains.  
<< text too long for this summary doc >> | 6/12/02 - Updated the document with GM’s input as modified by TT and with input from RK.  
8/1 JD: Jason identified two additional behaviors to support pattern merging: ‘dead-cycle insertion’ to wait through capture operations, and pattern reordering to maximize potential signal overlap (minimize non-overlapping patterns that need to be executed separately). Jason is concerned that the tiling/sequencing and lockstep operations ought to be merged. |
GM-7 Clarification is needed to 1450.0 wrt the Shift block:

1) Within a Shift block, the parameters to be shifted should be determined by the presence of a ScanIn or ScanOut attribute in the Signal block. This is as opposed to implying the shift from the presence of # characters.

2) Foreach ScanIn/ScanOut signal within a Shift block, the shift data can be defined in two ways:
   a) a parameter defined by # notation
   b) an in-line series of wfc’s

Note: This clarification becomes more clear when one looks at new capabilities needed in 1450.1 in support of SignalVariables.

GM-8 Need clarification on the processing of SignalVariables within a Shift block:

si = 10101 ‘ABC’ 10101; // ABC is a SignalVariable
si = 10101 ‘ABC’ ‘DEF’ 10101; // with two SignalVariable’s
si = 10101 ‘ABC[1]’ 10101; // with indexed SignalVariable
si = 10101 ‘XYZ[1..10]’ 10101; // with multi-bits of a SignalVariable
si = ‘w10101 ABC’ ‘w10101’; // illegal
si = 10101 ‘ABC DEF’ 10101; // illegal

GM-9 Editorial - Suggest moving Variables block AFTER the modification clauses.

GR-1 The extension specification really needs the equivalent of the 1450 orientation clause (which is about a third of IEEE Std 1450. The current document dives directly into syntax changes with no motivation and background. One day we would hope to see a "merged" standard, and we would expect the tutorial material to cover the full merged scope.

GR-1 The extension specification really needs the equivalent of the 1450 orientation clause (which is about a third of IEEE Std 1450. The current document dives directly into syntax changes with no motivation and background. One day we would hope to see a "merged" standard, and we would expect the tutorial material to cover the full merged scope.

Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM-7</td>
<td>Clarification is needed to 1450.0 wrt the Shift block:</td>
</tr>
</tbody>
</table>

1) Within a Shift block, the parameters to be shifted should be determined by the presence of a ScanIn or ScanOut attribute in the Signal block. This is as opposed to implying the shift from the presence of # characters.

2) Foreach ScanIn/ScanOut signal within a Shift block, the shift data can be defined in two ways:
   a) a parameter defined by # notation
   b) an in-line series of wfc’s

Note: This clarification becomes more clear when one looks at new capabilities needed in 1450.1 in support of SignalVariables. |
<table>
<thead>
<tr>
<th></th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7/16 - doc updated by TT</td>
</tr>
<tr>
<td>GM-8</td>
<td>Need clarification on the processing of SignalVariables within a Shift block:</td>
</tr>
</tbody>
</table>

si = 10101 ‘ABC’ 10101; // ABC is a SignalVariable
si = 10101 ‘ABC’ ‘DEF’ 10101; // with two SignalVariable’s
si = 10101 ‘ABC[1]’ 10101; // with indexed SignalVariable
si = 10101 ‘XYZ[1..10]’ 10101; // with multi-bits of a SignalVariable
si = ‘w10101 ABC’ ‘w10101’; // illegal
si = 10101 ‘ABC DEF’ 10101; // illegal |
|       | 7/16 - doc updated by TT |
| GM-9  | Editorial - Suggest moving Variables block AFTER the modification clauses. |
| GR-1  | The extension specification really needs the equivalent of the 1450 orientation clause (which is about a third of IEEE Std 1450. The current document dives directly into syntax changes with no motivation and background. One day we would hope to see a "merged" standard, and we would expect the tutorial material to cover the full merged scope. |
|       | 4/11/2002 - see TT1 |
GR-2 There is already a clash between the extensions: P1450.1 is making changes in some places being changed by P1450.2. One example is expressions: P1450.1 in Table 2 specifies that operators are allowed in 'timing and logical' expressions, whereas P1450.2 extended expressions to cover voltages, currents, slew rates etc.

4/11/2002 - All STIL dot Working Groups coordinate on all extensions to the base language; all extensions ARE consistent for the application of each dotted effort. No change identified.

GR-3 Back in an early 1450 ballot I expressed dislike for the density of single quotes in STIL and its seeming irregularity. Many of the examples show just how pernickety these rules now are. Is it feasible to fix whatever issues force this nastiness on users.

4/11/2002 - single-quotes are consistently used to identify expressions in the language. No change identified.

GR-4 STIL is obviously turning into an ever-extending language, using a "curly" syntactic style, yet the extension mechanism (UserKeywords) is still not capable of specifying true schema-like structuring rules (such as can be specified in XML schema or DTDs). Specifying the mechanisms for such flexible extensibility really belongs in the 1450 standard, with all these extensions then being defined within that extension framework. I'll be unhappy with all of these extensions until we tackle this problem.

4/11/2002 - UserKeywords in local contexts generates the need for many repeated definitions. Working Group expects two usage contexts: global user definitions, or local user definitions. A specific application (that has need of Userkeywords) would operate in one of these two contexts, and if the local context was at a level that required repeated definitions then the tool should consider moving the definition of that context to a higher level. See example at the end of 4/11 minutes. The notion of a "global definition that has local consequences" is excessive support for this construct which is intended to have MINIMAL application.

The Working Group discussed the reference of XML and notions of a general language. The context of the Working Group is that STIL is NOT intended to be a general language, and that extending the language toward generality is not the goal of the efforts. However, the Working Group understands that *something* might be done here but does not understand the requirement in order to address it.

---

**Table 1: Summary of Issues with 1450.1-D13**

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR-2</td>
<td>There is already a clash between the extensions: P1450.1 is making changes in some places being changed by P1450.2. One example is expressions: P1450.1 in Table 2 specifies that operators are allowed in 'timing and logical' expressions, whereas P1450.2 extended expressions to cover voltages, currents, slew rates etc.</td>
<td>4/11/2002 - All STIL dot Working Groups coordinate on all extensions to the base language; all extensions ARE consistent for the application of each dotted effort. No change identified.</td>
</tr>
<tr>
<td>GR-3</td>
<td>Back in an early 1450 ballot I expressed dislike for the density of single quotes in STIL and its seeming irregularity. Many of the examples show just how pernickety these rules now are. Is it feasible to fix whatever issues force this nastiness on users.</td>
<td>4/11/2002 - single-quotes are consistently used to identify expressions in the language. No change identified.</td>
</tr>
<tr>
<td>GR-4</td>
<td>STIL is obviously turning into an ever-extending language, using a &quot;curly&quot; syntactic style, yet the extension mechanism (UserKeywords) is still not capable of specifying true schema-like structuring rules (such as can be specified in XML schema or DTDs). Specifying the mechanisms for such flexible extensibility really belongs in the 1450 standard, with all these extensions then being defined within that extension framework. I'll be unhappy with all of these extensions until we tackle this problem.</td>
<td>4/11/2002 - UserKeywords in local contexts generates the need for many repeated definitions. Working Group expects two usage contexts: global user definitions, or local user definitions. A specific application (that has need of Userkeywords) would operate in one of these two contexts, and if the local context was at a level that required repeated definitions then the tool should consider moving the definition of that context to a higher level. See example at the end of 4/11 minutes. The notion of a &quot;global definition that has local consequences&quot; is excessive support for this construct which is intended to have MINIMAL application.</td>
</tr>
</tbody>
</table>

The Working Group discussed the reference of XML and notions of a general language. The context of the Working Group is that STIL is NOT intended to be a general language, and that extending the language toward generality is not the goal of the efforts. However, the Working Group understands that *something* might be done here but does not understand the requirement in order to address it.
## Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR-5</td>
<td>The detailed contents seems to have drawn lines around a scope that many could disagree with. For example the BistStructures name suggests generality, yet the contents are very LBIST-specific.</td>
<td>4/11 - &quot;contents of the spec identify a scope that can be disagreed with&quot;. While this effort (under the umbrella of a standards definition) must support sufficient definition to be used in the industry, the definition of needs and requirements is determined by participation in the effort. If specific issues can be identified requiring enhancements to constructs defined, then the Working Group is always overwilling to augment the existing definitions. The definitions currently present represent the Working Group's efforts at defining meaningful usable constructs.</td>
</tr>
<tr>
<td>GR-6</td>
<td>Glossary entries should be in the Definitions clause.</td>
<td>4/11 - All definitions in clause 3 become part of the IEEE list of standard definitions, and the previous ballot process of 1450.0 resulted in moving concepts that are local to a spec, into an Annex. If there are specific recommendations of definitions that ought to be part of the IEEE context then we're more than willing to move them into clause 3.</td>
</tr>
<tr>
<td>GR-7</td>
<td>&quot;Integers are not SignalGroups&quot; is penciled in large characters on the first page of Annex E. This is evidence of a serious gap in the type/concept structures of the language.</td>
<td>4/11 - This issue is well-understood by the Working Group, which has moved these constructs all around various areas of the language. While the concern is appreciated, the lack of a recommendation makes the Working Group unable to address this concern. 7/16/2002 - A new block named “Variables” has been created for defining integer variables and constants. 7/18 - wg agreed with the move to a separate “Variables” block. 8/1 - Further changes to the Variables block: Change Constant to Integer-Constant (to leave option for RealConstant). Elaborate in sec 10 on the use of Variables block for control vs. the Spec block for test parameters 8/15 - Change the Usage statement so that it only applies to Integer variables and has only “Test” as the attribute.</td>
</tr>
<tr>
<td>GR-8</td>
<td>Fail data is a different concept from Pattern data, and should not use the identical syntactic form. A new keyword expresses the fact that this is different information.</td>
<td>5/23 - JD is setting up a task force to discuss fail feedback (GR-8, GR-21)</td>
</tr>
</tbody>
</table>
### Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR-9</td>
<td>There is (to me at least) confusion about which parts of the language are declarative and which are obeying statements that can change the meaning of other expressions. My belief has always been that STIL is a purely declarative language.</td>
<td>5/23/2002 - JT and TT to review GM-3 and GR-9.</td>
</tr>
<tr>
<td>GR-10</td>
<td>Clause 9: This is an improvement on 1450, but doesn’t go far enough in defining extension mechanisms with full schema capability. This is also ugly as such local statements will get repeated many many many many many times.</td>
<td>4/11/2002 - see GR4</td>
</tr>
<tr>
<td>GR-11</td>
<td>Clause 12.1: &quot;LockStep&quot; needs to be differentiated from a single Pattern across the larger set of pins. For instance can the two Lockstep Patterns have different Loop structures provided that each Vector obeys Period rules?</td>
<td>LockStep means the the two patterns are doing the same things in each vector of the respective patterns. Loops, Calls, Macros, Shifts must all coincide.</td>
</tr>
<tr>
<td>GR12</td>
<td>Clause 14.1: These structures are growing and growing. And yet they can’t handle the latest types of scan structure from Mentor and IBM presented at ITC 2001 and TRP01.</td>
<td>What else is needed?</td>
</tr>
<tr>
<td>GR-13</td>
<td>&quot;Fixed&quot;, what’s the meaning if the WFC used to specify the value is one that implies changes? Even worse, what WaveformTable is used to say what the value means?</td>
<td>Does Fixed repeat the wfc, hold the last state, change with the W selection?</td>
</tr>
<tr>
<td>GR-14</td>
<td>The &quot;X&quot; statement with Offsets and Iterations is not rich enough to express the complexity of real vector execution contexts (think of &quot;the 15th vector after &quot;foo&quot; in the subroutine called from the 3rd vector after &quot;bar&quot; on the 99th time round that loop&quot;).</td>
<td>See DS-3</td>
</tr>
<tr>
<td>GR-15</td>
<td>18.1Numbering of items (e.g. (12)Type) is out of sync with syntax annotations.</td>
<td>5/23/2002 - TT to fix.</td>
</tr>
<tr>
<td>GR-16</td>
<td>18.2 CTL sneaks in again</td>
<td>Need to search/remove all CTL from this doc.</td>
</tr>
<tr>
<td>GR-17</td>
<td>19.1 Too specific for being called &quot;BistStructures&quot;. Useful stuff (e.g. cell types used in CA designs) is missing. In (12) replace &quot;offstate&quot; by &quot;offset&quot;,</td>
<td>Ask PW to review and comment.</td>
</tr>
</tbody>
</table>
### Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Identi</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR-18</td>
<td>19.3 Looked for [1] and eventually found it. Get all references together at one point in the Std.</td>
<td>5/23/2002 - TT to fix</td>
</tr>
<tr>
<td>GR-19</td>
<td>Annex E: Integers are not SignalGroups.</td>
<td>Integers are defined in the Signals block for convenience. Is this the best thing to do? See GR7.</td>
</tr>
<tr>
<td>GR-20</td>
<td>Annex I: What is meant by &quot;non-failure&quot; environments?</td>
<td>5/23/2002 - Intro to this annex re-worded by PW. Annex now called “Block Data Collection”.</td>
</tr>
<tr>
<td>GR-21</td>
<td>Annex N The &quot;tag&quot; mechanism is fundamentally ambiguous when &quot;vector splitting&quot; occurs. Earlier comments show how it fails to adequately specify many contexts. In general I want to see clear syntactic entries identifying information as fail data.</td>
<td>5/23/2002 - JD is setting up a task force to discuss fail feedback (GR-8, GR-21)</td>
</tr>
</tbody>
</table>
Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR22</td>
<td>1) Some of the things that look to me like &quot;operators&quot; aren’t mentioned as such in any of the tables. Specifically: [ and ] used in signal references. @ used in event time expressions , used in function parameter lists e.g. max(3,5). . used in hierarchy navigation 2) The precedence of the modulus (%) operator is very unconventional. Almost all languages give it the same precedence as multiply and divide, but P1450.1 has it separated from those by + and -. 3) P1450.1/6 make some strong distinctions between the various &quot;types&quot; of expression, and what operators are allowed in each type. But there seem to be many situations where there is no syntactic clue as to which of the expressions is allowed (e.g. in the examples in P1450.1, we see &quot;sigref&quot; expressions and &quot;logical&quot; expressions in exactly the same context. 4) This starts to get nasty when we add the lexical ambiguities that STIL has (e.g. is H1 an identifier or a sequence of WFCs). It’s plausible to argue that we should use operand type information from earlier in the expression, or symbol table info. to guide this, but that means that some operator could only be used &quot;one way round&quot;. For example, is fred == hll OK when fred’s a signalgroup, but not when its hll as the signalgroup name and fred as the “unusual” sequence of WFCs. 5) I’ve also got some lurking suspicions that the meaning of some of these might change (in obfuscated STIL contest situations) if a Pattern is used in contexts with different sets of group names, macro names etc. active. 6) Then of course we’ve got an old issue of mine that I believe that the ‘ticks’ around the expressions shouldn’t be necessary.</td>
<td></td>
</tr>
</tbody>
</table>
Table 1: Summary of Issues with 1450.1-D13

<table>
<thead>
<tr>
<th>Ident</th>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR-23</td>
<td>Technical - Need all keywords be reserved? See Appendix 1 for elaboration on this issue.</td>
<td></td>
</tr>
<tr>
<td>RK-1</td>
<td>Editorial - 22.6 I do not see any place in 1450.1 where you upgrade Clause STIL0-22.6 to allow an integer_expr in the place of loopcnt. (label:) Loop integer_expr { (pattern-statements)*}</td>
<td>The example in STIL1-12.4 shows an example of using a ‘count’ expression. In addition, a new sub-clause STIL1-16.7 is added to explicitly allow define such. Needs WG review and approval.</td>
</tr>
<tr>
<td>RK-2</td>
<td>Editorial - Annex F This example of tied scan chains running in lock-step can be simplified and improved.</td>
<td>4/23/02 - Rohit has reviewed his ideas with Tony &amp; Greg. The idea looks good. Rohit has agreed to re-work Annex F for the wg to review.</td>
</tr>
<tr>
<td>TT-1</td>
<td>Editorial - 1. Need a better intro section that sets the reader’s expectations as to what is in this standard. Need this since we don’t have a clause 5 - tutorial.</td>
<td>3/19/2002 - Tony has added an overview writeup to beginning of clause 1. 7/18/2002 - WG approved the change.</td>
</tr>
<tr>
<td>TT-2</td>
<td>Technical - BistStructures to be removed from the document. Can be handled as user-keyword in tools that need it.</td>
<td>7/16 - need wg discussion 7/18 - wg decided that a general email should be sent out to see what level of support there is for this construct before it is removed. 7/28 - email sent to stil-reflector and ccl-working-group 8/1 - Doug still checking for need within IBM; decision postponed two more weeks.</td>
</tr>
<tr>
<td>WG-1</td>
<td>Editorial - general Need to add references in the body of the document to the appropriate Annexes.</td>
<td>4/25 - tt - update the doc</td>
</tr>
<tr>
<td>WG-2</td>
<td>Need an explanation of \e</td>
<td>4/25 - Greg to provide a writeup</td>
</tr>
</tbody>
</table>

Appendix 1 - Reserved words

From: Gordon Robinson <Gordon_Robinson@3mts.com>
To: "Tony Taylor (E-mail)" <Tony.Taylor@synopsys.COM>,
   "Greg Maston (E-mail)" <g.a.maston@ieee.org>
Subject: Need all keywords be reserved?
Looking ahead to the set of dots under development, I’m getting concerned that our set of reserved keywords will grow too much.

As an example we can see many "enumerations" in CTL, each of which has many possible values.

The STIL style up until now has been to make all of these keywords reserved, and so make them unavailable to the user. Worse, every time we add such words we can invalidate things the user already has. That is why many programming languages have been reluctant to add further reserved words, and some have even reserved from first release a number of words they anticipate possibly using in later versions.

Many (most) of the keywords in STIL are used in constrained contexts, and so need not be reserved for any language design and parse reasons. For instance the values of enumerations could be treated syntactically as Identifiers, with the semantic rule that the value is one of a predefined set. If, however, the language definition states that they are "reserved", implementations are obliged to treat their use as an error, even if they use the "just an identifier" mechanism to implement the enumerations.

Back in the HILO and HITEST days we went to great lengths to avoid all reserved words. Even strong structural words like WHEN and RESET could be used as signal names (I leave you to guess what sort of monstrosity examples we wrote to check those pieces out).

STIL seems strange because there’s this great list of reserved words, but any of them can actually occur as a non-reserved token in vector data, so they’re not as reserved as they might appear.

I’d like to suggest that at the very least we make enumeration values not be reserved
in extensions to STIL, and sound the community out about whether we should "unreserve" some of the others.

Gordon