1450.4 meeting minutes – 11/20/09

Attendees: Jim O'Reilly, Ernie Wahl, Bruce Parnas, Markus Seuring, Ajay Khoche
Not present:
Agenda:

- IEEE Meeting Preamble (No discussion of proprietary information).
- Discussion items.
  - SigGroup data type (cont. from last week).
    - To allow passing Signal lists (signals, lists of signals, SignalGroups) into Tests or Flows.
    - Discussion issues: syntax, semantics, capabilities
    - **SigGroup** SIG_GROUP_NAME = *sigref_expr;*
    - Issues:
      - What about signal groups, individual signals (DataBus0), or bussed signals (DataBus[0])? How are these handled?
        - Per review of dot0, dot1, and discussions with Greg Maston:
          - Signal groups are expanded into the list of signals that make them up. No hierarchy of signal groups is retained if, for instance, a signal group contains another signal group. See Examples below.
          - Bracketed signals (i.e., data[0..7]) are treated as a list of individual signals (data[0], data[1], . . ., data[7]). The brackets and index are part of the signal name.
          - Further, per IEEE-1450.1-2005 (section 10, p32, last sentence), a bracketed signal automatically has a group created, named after its base name. For instance, with a bracketed signal (list of signals) data[0..7], a SignalGroup named data is automatically created. Therefore, it is illegal to define another signal group by that name, or to define a signal called data following definition of a signal data[0..7]. Likewise, if a signal called data is defined, it is illegal to define a bracketed signal called data[0..7], as the autocreated SignalGroup data will conflict with the previously defined signal named data.
      - How do we figure out how many individual signals are present in a *sigref_expr*? How do we walk through a *sigref_expr*, signal by signal?
        - Use .**Size** operator, which returns the number of individual signals in a sigref_expr.
          - SIG_GROUP_NAME.**Size**
      - How do you access individual elements of a *sigref_expr*?
        - Use index notation.
          - SIG_GROUP_NAME[*integer_expr*] – *integer_expr* must evaluate to >=0.
      - Additional operator .Name is available for individual elements of **SigGroup**. SIG_GROUP_NAME[*integer_expr*].**Name** returns in a string the name of the specified element of SIG_GROUP_NAME
  - Examples:

```
Signals {
    data[0..7] { };  // Per dot1, auto-created SignalGroup data is created
    addr[7..0] { }; // Per dot1, auto-created SignalGroup addr is created
    data { };  // Not allowed per dot1- conflicts with auto-created SignalGroup data
    clk { };
    rst { };
    tack { };
}
```

```
SignalGroups {
    grp1 = data[0..7] + clk;      // grp1 = data[0], data[1], . . ., data[7], clk1
    grp2 = rst + tack;
    grp3 = grp1 + rst;            // grp3 = data[0], data[1], . . ., data[7], clk1, rst
    grp4 = rst + addr[7..0];      // grp4 = rst, addr[7], addr[6], . . ., addr[0]
    grp5 = rst + addr;            // grp5 = rst, addr[7], addr[6], . . ., addr[0]
    grp6 = data + rst + addr[7..0]          // grp5 = data[0], data[1], . . ., data[7], rst,
                                               addr[7], addr[6], . . ., addr[0]
}

SigGroupVar sigref_1 = grp1;
SigGroupVar sigref_2 = grp6;


sigref_1.Size = 9 (8 from data[0], data[1], . . ., data[7], and 1 from clk)
sigref_1[0] yields data[0]
sigref_1[7] yields data[7]
sigref_1[8] yields clk

sigref_2.Size = 17 (8 from data[0], data[1], . . ., data[7], 1 from clk, 8 from addr[7], . . ., addr[0])
sigref_2[0] yields data[0]
sigref_2[7] yields data[7]
sigref_2[8] yields clk
sigref_2[9] yields addr[7]
sigref_2[16] yields addr[0]

sigref_2[7].Name returns "data[7]" (including the brackets).
```

- SpecVariable type.
    - To allow passing spec variables into Tests or Flows
    - Behavior as specified by dot0.
        - If only Typ is specified, no selector is needed. Do Min and Max exist or not in this case?
            - No. If only Typ is used, Min and Max do not exist, and cannot be used. An error is generated if they are used.
        - Does Meas exist, even if not specified in Spec block?. What value does it have if not initialized?
            - Yes, .Meas ALWAYS exists. The mechanism for assigning a value to .Meas is not specified in dot0. Further, the value of .Meas prior to the time it is assigned a value is not specified by dot0, and is likely to be implementation dependent..
        - Are the only options to define Typ only, or Min, Typ, Max? i.e., you can't define Min and Typ only, or Typ and Max only, or Min and Max only? (Syntax shows that when using block format - specifying Min, Typ, Max - each element is optional . . .)
            - The syntax is correct. When using the assignment form, only the Typ value is assigned. If using the block form, any combination of Min, Typ, and/or Max can be assigned.
            - As mentioned above, if a Min, Typ or Max value is not specified, it doesn't exist and can't be used; however, the .Meas value ALWAYS exists even if not mentioned, and can be used – though its value is indeterminate until a value is actually assigned.
        - Do we need an both an initial and a current value for Meas? (Meas.original and Meas.current, as is done for bin counters).

        - No. This was discussed, and we determined that allowing a data value of NaN (Not a Number) which can be assigned anywhere a numerical value or expression which evaluates to a numerical value would be sufficient. The expected usage of this is initially for the .Meas field.

    - **Proposed behavior for dot4**.
        - Whether specified or not, all values of a spec variable (Min, Typ, Max, and Meas) are defined. If not assigned a value in the spec block, the value NaN is assigned.
            - If using the assignment form, only Typ is defined, and Min, Max, and Meas have the value NaN
            - If using the block form, any selector fields specified (Min, Typ, Max, and/or Meas) have a value (however, that value CAN be NaN, if the user chooses!). Any selector fields NOT specified have the value NaN.
            - If a variable evaluates to NaN, any expression using this variable also evaluates to NaN. It is an error to use a variable which evaluates to NaN to specify values of a hardware resource (timing edges, DC levels, etc.).
            - All Selector subfields for a variable (i.e.,Min, Typ, Max, Meas) MUST have the same units (Seconds, Volts) – but these units can be scaled differently (i.e., Typ can be mV, Meas can be uV).
            - Two operators are defined for spec variables:
                - .**Name** (i.e., <SPEC_VAR_NAME>.**Name**. This operator returns the spec variable name as a string.
                - .**Units** (i.e., <SPEC_VAR_NAME>.**Units.** This operator returns the spec variable units (i.e., Seconds, Volts) as a string. The units returned are in canonical form. For instance, if a variable of type Power was calculated as I^2R, the units will be Watts. Likewise, Coulombs/Seconds will return the units of Current, which is Amperes (A). This behavior is especially important for Compound types. If the (compound) Units of a variable do not reduce to one of the variable types defined in STIL, .Units will return a string containing the literal representation (.i.e., Volts/Watt, which would be V/W. **NOTE: We may need to discuss this in more depth!!!**

- Other issues:
    - Conversions from one data type to another – what are the rules? These need to be explicitly defined.
    - Limits as a specific data type?
        - How to specify an open limit on one end of a comparison?
        - Specify >=, >, <=, < - what does the syntax look like?
        - Limits x 0nS >= n <= 3nS
    - List of reserved words by tester type as an integral part of dot4? (should this be a dot3 addition?). Does NameChecks in dot3 cover this, or could it cover it with some extensions.
    - Added issue #44 (extensions to signal types In, Out, InOut, Pseudo) are needed – for instance, to permit specification of unused package pins.
        - Perhaps a signal type called NC (no-connect) would be useful in these cases.

- Open issues - are there other open issues that should be considered? A review of the open issues list can guide us here.
    - http://spreadsheets.google.com/ccc?key=pEI1-gPUmt2ZTw_kcCTgnKw&inv=jim_oreilly@ieee.org&t=9330484534885518 71&guest).
    - If logged into your google account, can edit. If not, can only view.
- Next Meeting 12/03/09.

For reference STIL .4 information can be found at the IEEE STIL website: http://grouper.ieee.org/groups/1450/ (select the P1450.4 link from the table) or use the direct link http://grouper.ieee.org/groups/1450/dot4/index.html