1450.4 meeting minutes - 12/06/07

Attendees:Doug Sprague, Jim O'Reilly, Jose Santiago, Bruce Parnas, Ernie Wahl

Not present: Ajay Koche

Agenda:

- Preamble:
    - Record Meeting (*2)
        - To listen to the meeting recording, do the following:
            - Call the (US) dial-in numbers 1-877-421-0003 (toll free) or 1-770-615-1374 (toll)
            - Enter the passcode code 747464
            - Once dialed in with the proper access code, enter *3 (star 3)
            - Then enter the file number 67497501 for this conference (this number will change each week).
            - Press 1 to listen to the conference.
    - IEEE Meeting Preamble (No discussion of proprietary information).
- Review high level todo list (below), assign more names??
    - Binning (Ernie)
    - Runtime Variables (Doug)
    - PatternExec constructs (Jim)
    - Bypass (in PreActions) syntax and semantics (Jim)
    - TestBase (Jim)
    - Input/Output Interface? (Ernie??)
        - Writing/Reading to/from screen, files
        - Interfacing to Shmoo Plotting
    - MultiSite (??)
    - Standard Predefined Test Methods (Ernie)
    - Datalogging Interface? (??)
    - Process (Jose, Doug, Jim)
    - Framemaker Documentation (Jose)

- Continue review of binning document from Ernie, starting in section 1.5
    - Sent on 11/15 by Ernie, also attached to minutes from last meeting sent today
    - Reviewed proposed syntax changes to the BinDefs and BinMaps block (see below).
        - The names of the BinDefs and BinMap blocks are no longer optional, but required.
        - Pass and Fail subblocks within the BinDefs block are no longer optional, but required. The contents of these blocks, however, are optional (either the Pass or the Fail subblock can be empty).
        - Within the Pass and Fail subblocks of the BinDefs block, one can use either the non-axis notation or the axis notation (see below), but the non-axis notation and the axis notation cannot be mixed within the Pass or the Fail subblock. It is allowable, however, to use the axis notation in one subblock, and the non-axis notation in the other.
        - Within the BinMap block, it is allowable to mix the non-axis notation and the axis notation (this is required if one uses, for example, the axis notation for the Pass subblock of the BinDefs block, and the non-axis notation for the Fail subblock). In reviewing these change for the BinMap block after the meeting, however, I believe that what we want instead of:

```
BinMap BIN_MAP_NAME {   // soft to hard bin mapping
        ( SOFTBIN_NAME -> integer; )* |
        ( [ (BIN_AXIS.SOFTBIN_NAME)* ] -> integer; )*
}
```

is:

```
    bin_map_stmt =
        SOFTBIN_NAME -> integer; |
        ( [ BIN_AXIS.SOFTBIN_NAME] )* -> integer;

    BinMap BIN_MAP_NAME {   // soft to hard bin mapping
        (bin_map_stmt )*
    }
```

The revised syntax shows that a binmap can include both axis and non-axis bin_map_stmts, and fixes the BNF notation so that a binmap statement which is the intersection of two or more axes can be specified.

The actual syntax for the bin statement needs further clarification. In particular, that statement now includes only a softbin name and optional integer. Other existing SW systems include additional information, such as bin counters, retest or reprobe flags, override on fail (on a global or per-bin basis) flags, and in some cases, even the color to use for that bin in a GUI wafermap or result display. Our model and syntax probably needs to include some or all of these.

- BinAndStop in postactions of testbase
    - Issue discussed if we need BinAndStop or just Bin and Stop as separate actions
    - Ernie feels that BinAndStop is needed in order to use that action in TestBase. The semantics are as follows: If the Bin parameter for a test is not set, no action happens; otherwise, if the Bin parameter for a test is set, then the appropriate binning takes place and the test stops. It is functionally equivalent to:

      ```
      if (Bin != NULL) {
          SetBin Bin;
          Stop;
      }
      ```

      Since it is functionally equivalent to the above, the BinAndStop keyword is not absolutely required, but provides a convenient shorthand.
    - Conclusion – will include both
- Indexing over all the bins on all axes of a multi-axis bin definitions.
    - This issue is tied up with a number of other issues, such as:
        - Do we have a notion of a test loop?
            - No explicit loop construct, but can construct one with branching from flownode output back to previous point in the flow.
        - How does one iterate a test over all categories of a particular spec, repeating the test for each category?
        - How does iteration over categories of a spec tie in with iteration over bins in a multi-axis bin definition?
        - These issues need to be resolved in order to resolve the complete bin
    - Iterate using numerical index, or by name?
    - Ernie wants to do it by index example
      AC test, at least 3 specs, setup time, 3.00ghz, 2.93ghz, 2.66ghz
      Iteration over all the bins

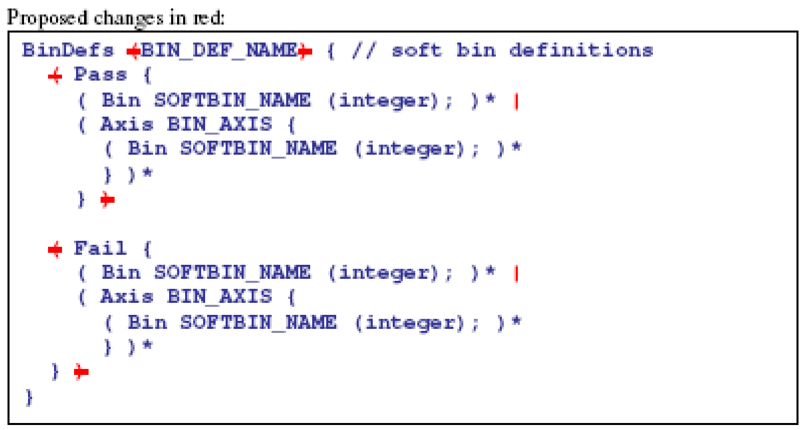
      ```
      for 0 to upper speed index
          do  AC test[1] and bin appropriately
      end loop
      ```
    - If adding or removing specs, would like the ability to enable or disable each axis of a bin definition so that one can easily add or remove specs without modifying the bin definitions.
    - Would like to have Ernie provide an example from his previous work as to how the binning and specs tie together in such a scenario.

Next meeting:

- Next meeting 12/13/2007. Will poll next week regarding the meeting on 12/20/2007. The meeting for 12/27/2007 is cancelled; meetings to resume after that on 01/03/2008.

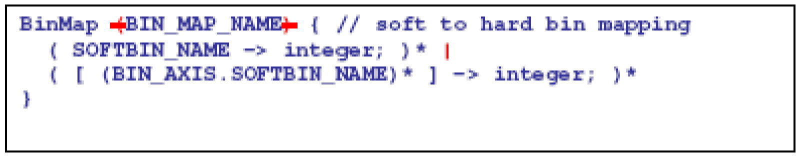
For reference STIL .4 information can be found at the IEEE STIL website: http://grouper.ieee.org/groups/1450/ (select the P1450.4 link from the table) or use the direct link http://grouper.ieee.org/groups/1450/dot4/index.html

Proposed syntax changes to BinDefs block

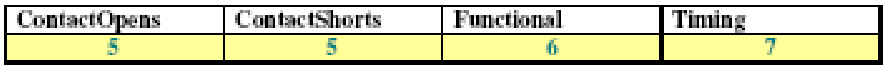

Proposed changes in red:

```
BinDefs ~~(~~BIN_DEF_NAME~~)~~ { // soft bin definitions
  ~~(~~ Pass {
    ( Bin SOFTBIN_NAME (integer); )* |
    ( Axis BIN_AXIS {
      ( Bin SOFTBIN_NAME (integer); )*
      } )*
    } ~~)~~

  ~~(~~ Fail {
    ( Bin SOFTBIN_NAME (integer); )* |
    ( Axis BIN_AXIS {
      ( Bin SOFTBIN_NAME (integer); )*
      } )*
  } ~~)~~
}
```

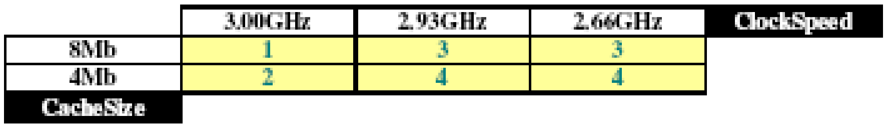


```
BinMap ~~(~~BIN_MAP_NAME~~)~~ { // soft to hard bin mapping
  ( SOFTBIN_NAME -> integer; )* |
  ( [ (BIN_AXIS.SOFTBIN_NAME)* ] -> integer; )*
}
```



ContactOpens	ContactShorts	Functional	Timing
5	5	6	7

Table 1: Fail bin group with anonymous axis



	3.00GHz	2.93GHz	2.66GHz	ClockSpeed
8Mb	1	3	3	
4Mb	2	4	4	
CacheSize				

Table 2: Pass bin group with two labeled axes

Fig. 1: Single-axis fail bin and multi-axis pass bin definitions example from Ernie's conceptual document.

```
BinDefs MyBinDefs {
        Pass {
                BinAxis ClockSpeed {
                        Bin 3_00GHz 1;
                        Bin 2_93 GHz 2;
                        Bin 2_66 GHz 3;
                }
                BinAxis CacheSize {
                        Bin 8MB 1;
                        Bin 4MB 2;
                }
        }
        Fail {
                Bin ContactOpens 1;
                Bin ContactShorts 2;
                Bin Functional 3;
                Bin Timing 4;
        }
}

BinMap MyBinMap {   // soft to hard bin mapping
        ContactOpens -> 5;
        ContactShorts -> 5;
        Functional -> 6;
        Timing -> 7;
        [ClockSpeed.3_00GHz][CacheSize.8MB] -> 1;
        [ClockSpeed.3_00GHz][CacheSize.4MB] -> 2;
        [ClockSpeed.2_93GHz][CacheSize.8MB] -> 3;
        [ClockSpeed.2_93GHz][CacheSize.4MB] -> 4;
        [ClockSpeed.2_66GHz][CacheSize.8MB] -> 3;
        [ClockSpeed.2_66GHz][CacheSize.4MB] -> 4;
}
```

Proposed syntax for mapping bins at the intersection of two or more axes to a hardware bin #

Fig. 2: Syntax example implementing binmapping from Fig 1. above