STIL P1450.4 Binning

Summarized below are the key points of binning from several different commercially available testers.

Agilent 93000

Bin architecture:

A bin is defined as

```
struct {
        char hardbin_string[2];
        char *softbin_string;
        int hardbin_number;     // 0-999
        bool pass_fail;         // good | bad
        bool reprobe_flag;      // reprobe | noreprobe
        int color;              // 0-7 or black | white | red | yellow |
                                //      green | cyan | blue | magenta
        bool over_on_flag;      // over_on | not_over_on
} Bin;
```

There are two types of test execution flow-nodes

- run
- run-and-branch

and two types of bin flow-nodes:

- Stop bin (can be good or bad)
- Otherwise bin (if not defined by user, a system default is used).

The display icons for test execution and bin flow-nodes are shown below in Fig. 1.

Bin map(s):

List of bin structures described above. Only one bin map available in the program. As far as I know, there are no limits to the number of bins a binmap can contain - though, with the exception of softbin_string, the range of values each structure member can take on would seem to limit the number of permutations. Nonetheless, as each unique bin is defined (and used), it is simply placed in the binlist.

Containment hierarchy (relationship to flow):

- A bin is essentially a terminal flow-node.
- A flow does not need to contain bin assignment flow-nodes; if not, a default "otherwise" bin is predefined.
- No other flow-nodes can follow a bin flow-node (i.e, a bin node has an input but no output).
- The input to a bin flow-node is from the output of one (and only one) flow-node (this is really more a function of the testflow, rather than the binning – the bin just happens to be a flow-node).
- Binning is tied to a bin flow-node. Bin assignment (via a bin flow-node) selects one of N bins in the binlist (i.e., hard binning and soft binning happen together, not separately).
- Two types of test execution flow-nodes, each of which can be followed by another flow-node, or a bin node.
    - run
    - run-and-branch

Bin/stop relationship:

Two modes:

- Stop on fail:
  Disables/powers down, skips over remaining tests and testblocks, sets bin based on the bin flow-node connected to the the fail port of the failing test. If no bin flow-node is present, the flow continues to the next execute flow-node. At the end of the test flow, if no bin has been assigned, the otherwise bin is assigned.
- Override on.
  Flow continues past the fail bin to the flow-node immediately following the failing test (as indicated by a dotted line in the testflow – see Fig. 2 below). If there are no more tests after the stop bin, the otherwise bin is assigned.

Bin strategies:

- Bin flow nodes are used in testflow:
  Bin based on the bin assigned from a bin-flow-node (assign fail-bin, pass, bin, or otherwise bin, depending on pass/fail results and overon setting)
- Bin flow nodes are not used in testflow:
  Only the otherwise bin is used, and always assigned at the end of the testflow.

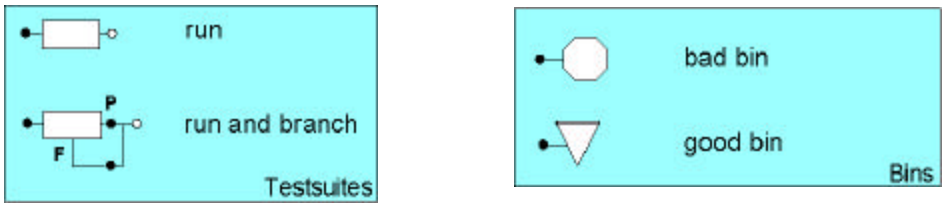

Fig. 1 Agilent 93000 - Display icons for test execution and bin flow-nodes

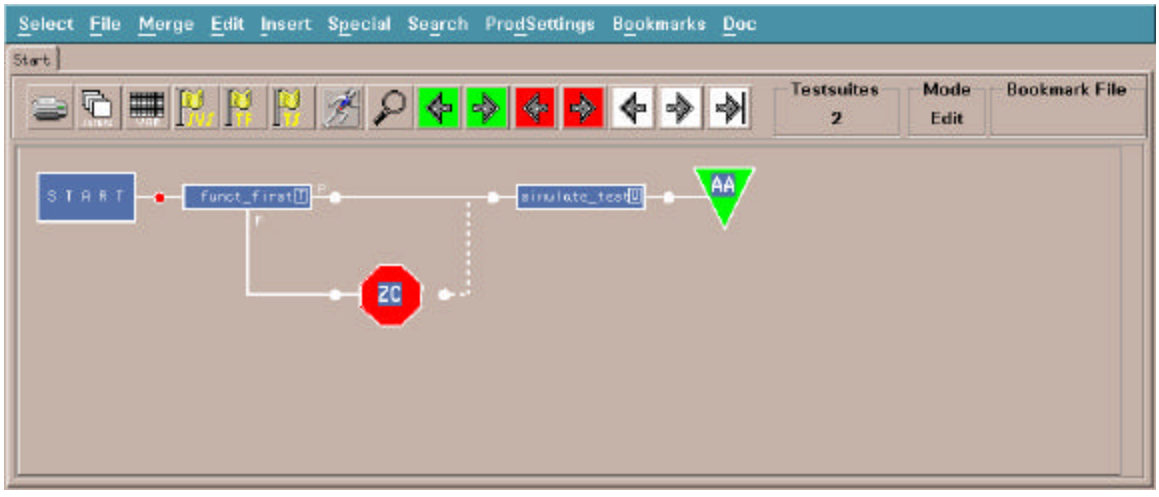

Fig. 2 Agilent 93000 - Graphical representation of test flow including pass/fail bins and run/run-and-branch flow nodes

```
test_flow

  run_and_branch(funct_first)    then
  {
  }
  else
  {
       stop_bin "ZC", "fp_cont_fail", , bad,noreprobe,red, 7, not_over_on;  // Fail bin
  }
  run(simulate_test_time__2_);
  stop_bin "AA", "p_good_part", , good,noreprobe,green, 1, not_over_on;  // pass bin
end
-----------------------------------------------------------------
binning

otherwise bin = "DB", "otherwise_bin", , bad,noreprobe,cyan, 99, not_over_on;  // otherwise bin
end
```

Fig. 3 Agilent 93000 - Text representation of graphical test flow shown in Fig. 2

Teradyne Binning:

Bin Architecture:

A bin is defined as a structure containing softbin number, softbin string, hardbin number, hardbin string, and a pass/fail/error classification, e.g.,

1 "GRADE_1" hbin =1 "GRADE_1" pass,

Bin maps:

Array of 256 instances of bin struct described above in “Bin Architecture”. Only one bin map allowed per program.

Containment hierarchy (relationship to flow):

soft binning is tied to a test, however, a test may be an otherwise empty container. Soft binning is separate from hard binning.

Bin/stop relationship:

3 modes

a) stop on fail: disables and powers down affected site, skips over remaining tests and testblocks, and proceeds to hardware binning
b) continue on fail: registers the first bin failed, no intra-chiptest count (maintains chip to chip summary, e.g. failed bin 15 twice)
c) continue on error: not sure what happens

Bin strategies: there are 3

a) pass binning: set soft bin on test pass
b) fail binning: set soft bin on test fail
c) disqualify binning: unset soft bin(s) on test fail

Credence Binning:

Bin Architecture:

A bin is defined as a structure containing softbin number, and hardbin number.

Bin maps:

Array of 128 instances of bin struct described above in “Bin Architecture”. Only one bin map allowed per program.

Containment hierarchy (relationship to flow):

soft binning is tied to a test, however, a test may be an otherwise empty container. Soft binning is separate from hard binning.

Bin/stop relationship:

there is 1 mode, stop on fail. There is a per test ignore fail option but then no binning takes place.

Bin strategies: there are 2

a) fail binning: set soft bin on test fail
b) unconditional binning: set soft bin unconditionally

Schlumberger ITS9000

Bin Architecture:

A bin is defined as a structure containing three elements

- virtual (soft) bin number
  The virtual bin number will be mapped to a hardware bin number in the binmap. The virtual bin number is used to set the working value of the virtual bin number as the flow passes through the port of a flow node. The soft bin number or a device is usually set to the value of the working virtual bin number at the end of test.
- Bit bin number.
  Each bin object has a bit bin number (which normally defaults to 0). The bit bin number is used when it is necessary to calculate the soft bin based on the path of the flow through the test. At key places in the test flow, the bit bin number is assigned to a bit mask. As the flow passes through a flow node with a bit bin number assigned, this mask is logically ored into a working bit bin mask. At the end of test, the working value of the bit bin number is added to the working virtual bin number to calculate the soft bin number for the device. The final soft-bin number is then mapped to a hardware bin in the bin map.
- Flow id bit number
  The Flow id is used when it is necessary to replace the normal binning mechanism with a user defined mechanism. To track the flow, a unique Flow Id Bit Number [0 .. 4095] set at the input or output ports of each flow node. As the flow passes through a port of a flow now with a Flow id Bit number assigned, a bit is ored into into the Flow id bit array (4096 bits) at the bit number specified. This array can be access by a user routine at end of test to override the normal binning mechanism.

Bin maps:

The bin map has two sections – a software bin section and a hardware bin section.

- Hardware bin section
  - Defines the available hardware bins, including
  - Hardware bin name
  - PASS or FAIL
  - Reprobe this bin or not (if so, the maximum allowable reprobe count)
  - bin number (to be sent to the prober or handler).
- Software bin section
  - Assigns a name to each soft bin number (specified in each bin object).
  - Maps each soft bin to a hardware bin (any number of soft bins can be mapped to a single hard bin).

Multiple binmaps can be used in the test program. Only one at a time, of course, can be active.

Containment hierarchy (relationship to flow):

- A bin can be attached to either the input or output of a testflow node.
- The virtual bin number and bit bin number of each bin must be selected so that soft bin calculated at the end of test matches a soft bin in the selected bit map.
- When a bin is assigned at a flow-node port, only the soft-bin is assigned. At the end of the flow, the most-recently-assigned soft bin is mapped to a hard bin via the currently-selected bin map.
- A flow does not need to contain a bin assignment or a bin map to run. If no binmap exists or no bin assignment is done in the flow, then the test program's "current bin" (i.e., the most-recently assigned bin) is a null bin object.
- A test program can have more than one bin map. However the only one bin map can be selected at a time.

Bin/stop relationship:

- Bin assignment and test flow stop/continue execution are separate. The soft bin assignments are made as the flow passes through flow node ports with a bin attached. The device is binned according to the soft bin assignment at the end of testing (when a terminal node is reached).

- Stop on fail is not supported. The flow always continues until it reaches a terminal flow node. In the typical case, however, a test will have one fail node and at least one pass node. The flow is constructed so that a test-flow node's failing port leads directly to a terminal node (a "Stop" segment, in ITS9000 terminology). The ITS9000 software does allow setting breakpoints or pauses at various points in the test flow (including breakpoints if a specific test fails).

- Overide on Fail is supported.

Bin strategies:

- Bin Assignment Strategy
  The virtual bin number is used to assign the working soft bin number as the flow passes through the port of a flow node. At the end of test, the last soft bin number assigned is used to look up the soft bin and hard bin information in the selected bin map.

- Path Dependent Strategy
  This strategy is used when it is necessary to calculate the soft bin based on the path of the flow through the test. At key places in the test flow, the bit bin number is assigned to a bit mask.

  As the flow passes through a flow node with a bit bin number assigned, this mask is logically ored into a working bit bin mask. At the end of test, the working value of the bit bin number is added to the working virtual bin number to calculate the soft bin number for the device. This soft bin number is used to look up the soft bin and hard bin information in the selected bin map.

- User Defined Strategy
  This strategy is used when it is necessary to replace the native binning strategy with a user defined bining strategy. On popular way to do this is by tracking the complete flow path by assigning unique Flow Id Bit Number [0 .. 4095] set at the input port or at all output ports of each flow node. As the flow passes through a port of a flow now with a Flow id Bit number assigned, a bit is ored into into the Flow id bit array (4096 bits) at the bit number specified. This array can be access by a user routine at end of test to calculate the soft bin number based on the complete path of execution.

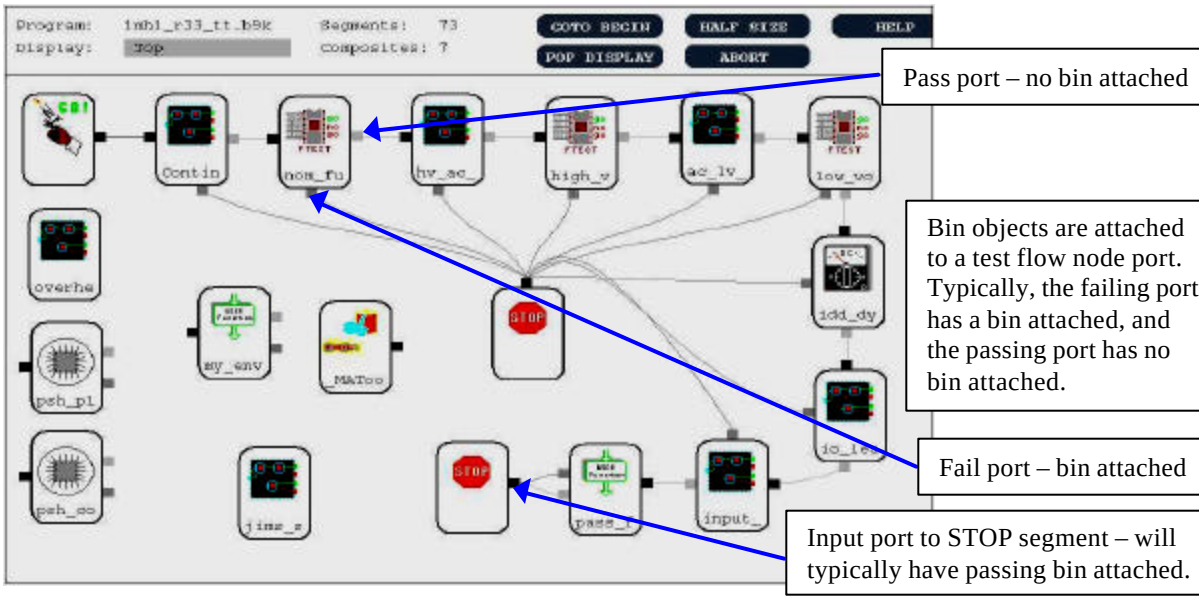

Fig. 4 Schlumberger ITS9000 – Graphical representation of test flow

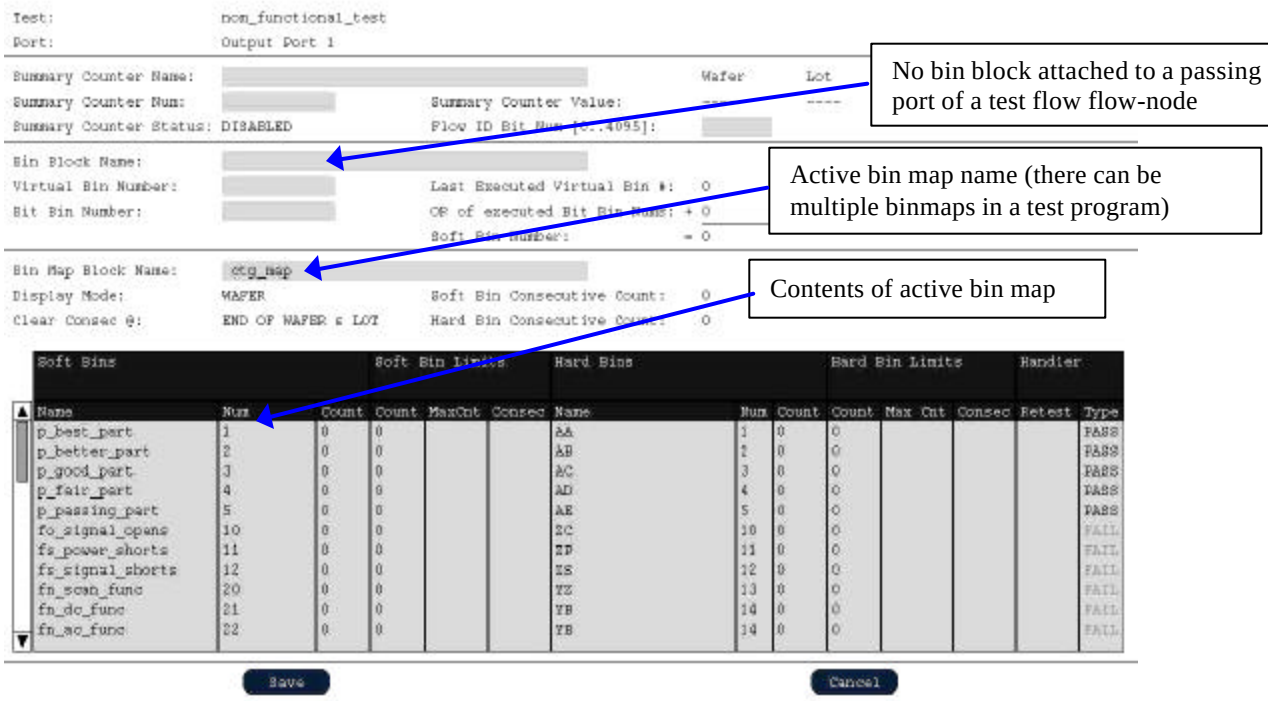

Fig. 5 Schlumberger ITS9000 – Bin/Binmap window opened at passing port – no bin attached

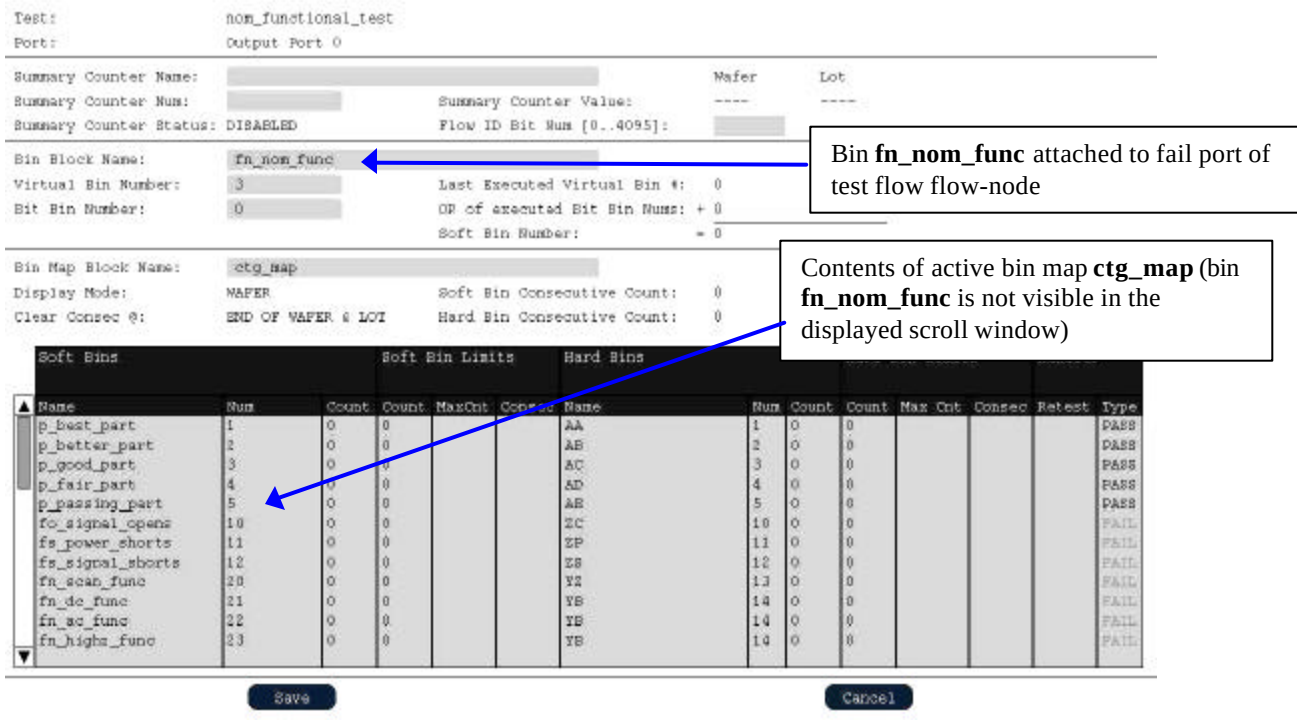

Fig. 6 Schlumberger ITS9000 – Bin/Binmap window opened at failing port – bin fn_nom_func attached

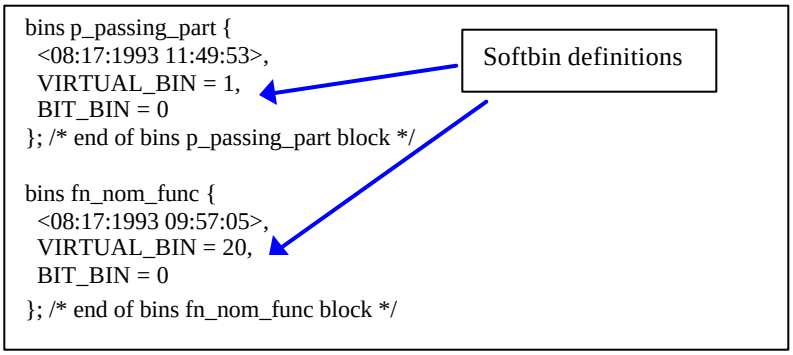

Fig. 7 Schlumberger ITS9000 – Test program syntax example for bin blocks

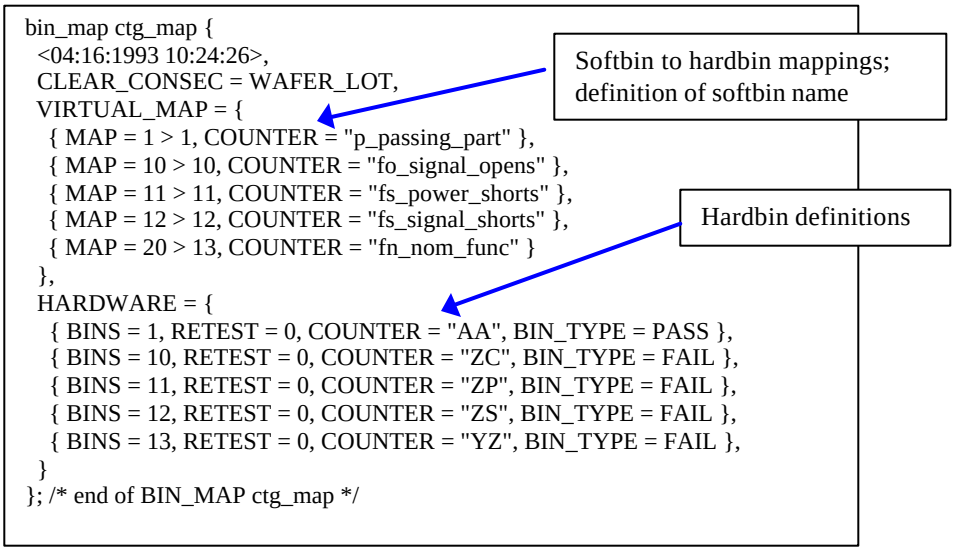

Fig. 8 Schlumberger ITS9000 – Test program syntax example for bin map

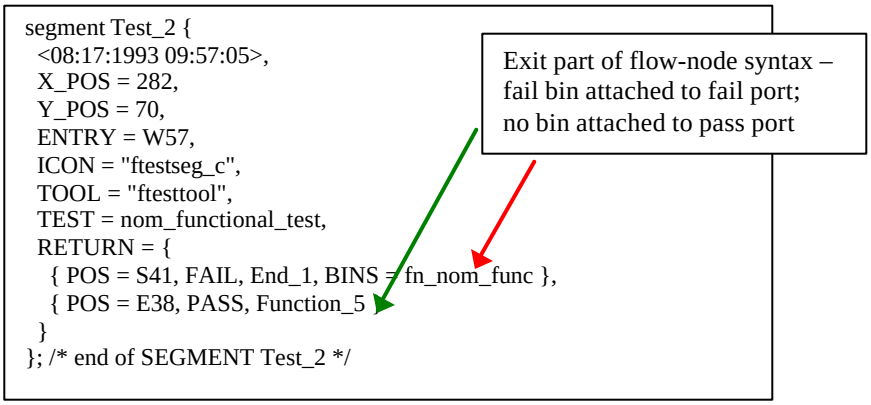

Fig. 9 Schlumberger ITS9000 – Test program syntax example for flow-node