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.

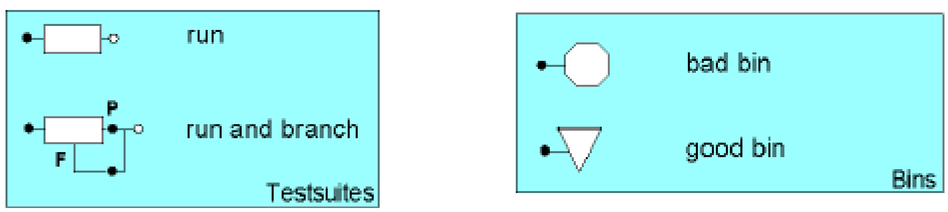

Fig. 1 Agilent 93000 - Display icons for test execution and bin flow-nodes

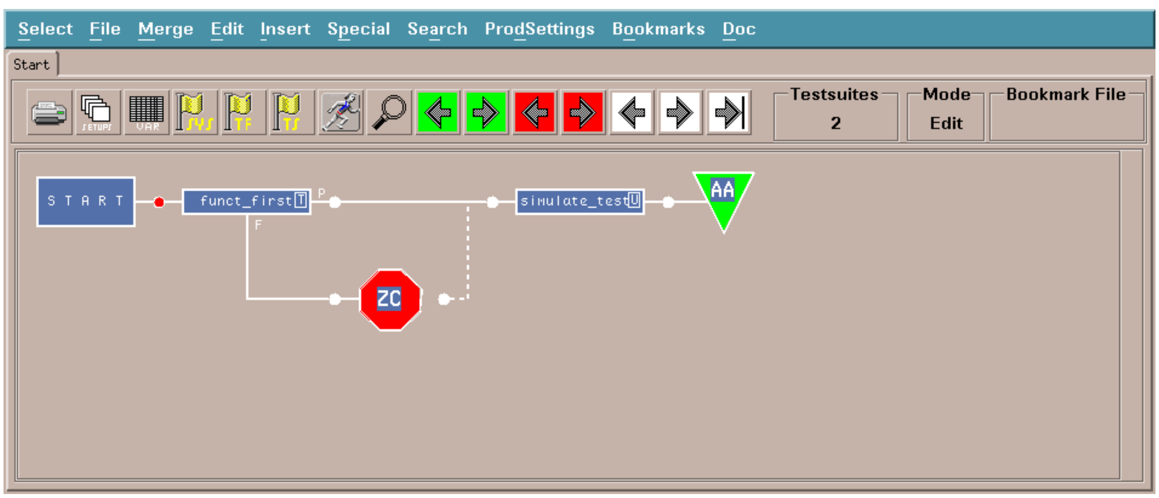

Fig. 2 Agilent 93000 - Graphical representation of test flow including pass/fail bins and run/run-and-branch flow nodes

```
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 two elements

- virtual (soft) bin number
  The virtual bin number will be mapped to a hardware bin number in the binmap.
- Bit bin number.
  Each bin object has a bit bin number (which normally defaults to 0). When the flow passes through a flow-node port which contains a bin object, the bit bin number is OR'ed into a global bit-bin variable. At the conclusion of the test flow, the global bit-bin number is OR'ed with the current soft-bin number, to generate a final soft-bin number, which is then mapped to a hardware bin.

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 bins available for attachment to a flow-node input or output port are selected from a bin map. A test program can have multiple bin maps (but obviously, only one can be selected at any given time).
- All bins used in a testflow must be contained in the currently selected bin map.
- A flow does not need to contain any bin assignment at the flow-node input or output ports; if no bin assignment is done in the flow, then the test program's "current bin" (i.e., the most-recently assigned bin) is an empty bin object.
- When a bin is assigned at a flow-node port, only the soft-bin is assigned. At the end of the flow, the soft bin is mapped to a hard bin via the currently-selected bin map

There is also a 4K flow ID array, which can be used to uniquely identify the path taken through a flow. The flow ID number is assigned through the binmap/bin assignment tool.

Bin/stop relationship:

Two modes:

- Stop on fail
- Override on

Bin assignment and test flow stop/continue execution are separate. Test flow execution does NOT stop just because a bin has been assigned. Rather, it continues until it reaches a terminal flow-node (a predefined "STOP" segment, in ITS9000 terminology). Or, put another way, a bin is assigned ONLY if and when the flow transits a flow-node port to which a bin has been attached.

Bin strategies:

Set soft bin and bit bin (which defaults to 0) only when the flow transits a flow-node port (input or output, pass or fail) to which a bin is attached. The soft bin can be changed as the flow passes throught different flow-node ports. At end of flow, the current soft bin is OR'ed with the global bit-bin variable; that result is then mapped to a hard bin via the currently selected bin map. **(HELP: Dan Fan or Jim Moseley, can you supply some additional details regarding binning strategies – as defined for Credence or Teradyne, or as defined by Schlumberger?)**

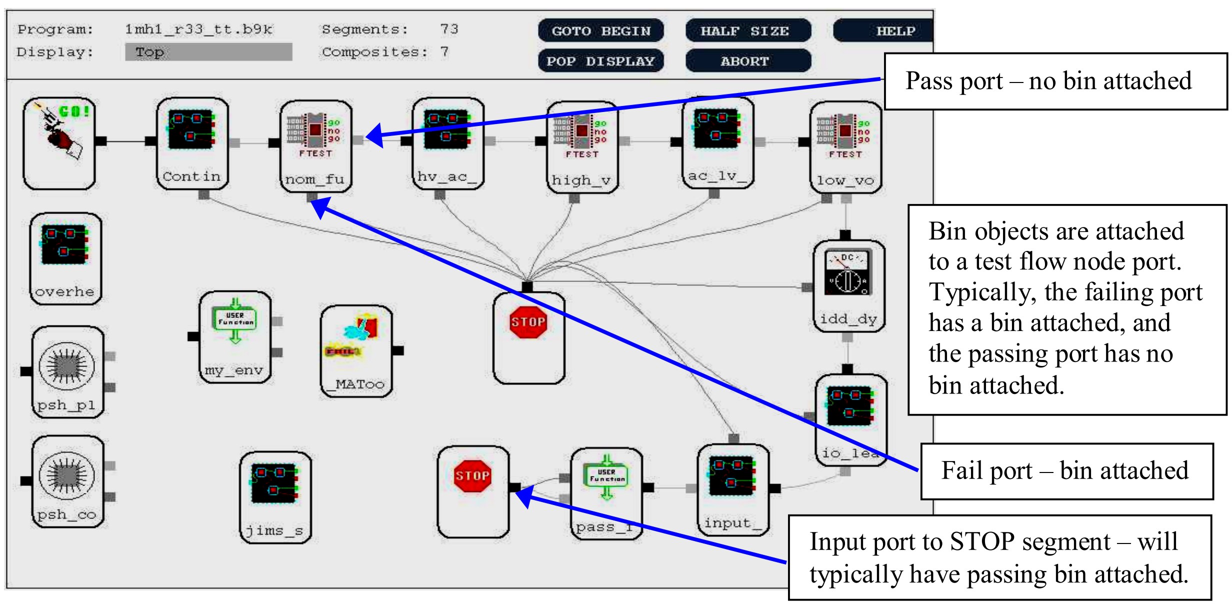

Fig. 4 Schlumberger ITS9000 – Graphical representation of test flow

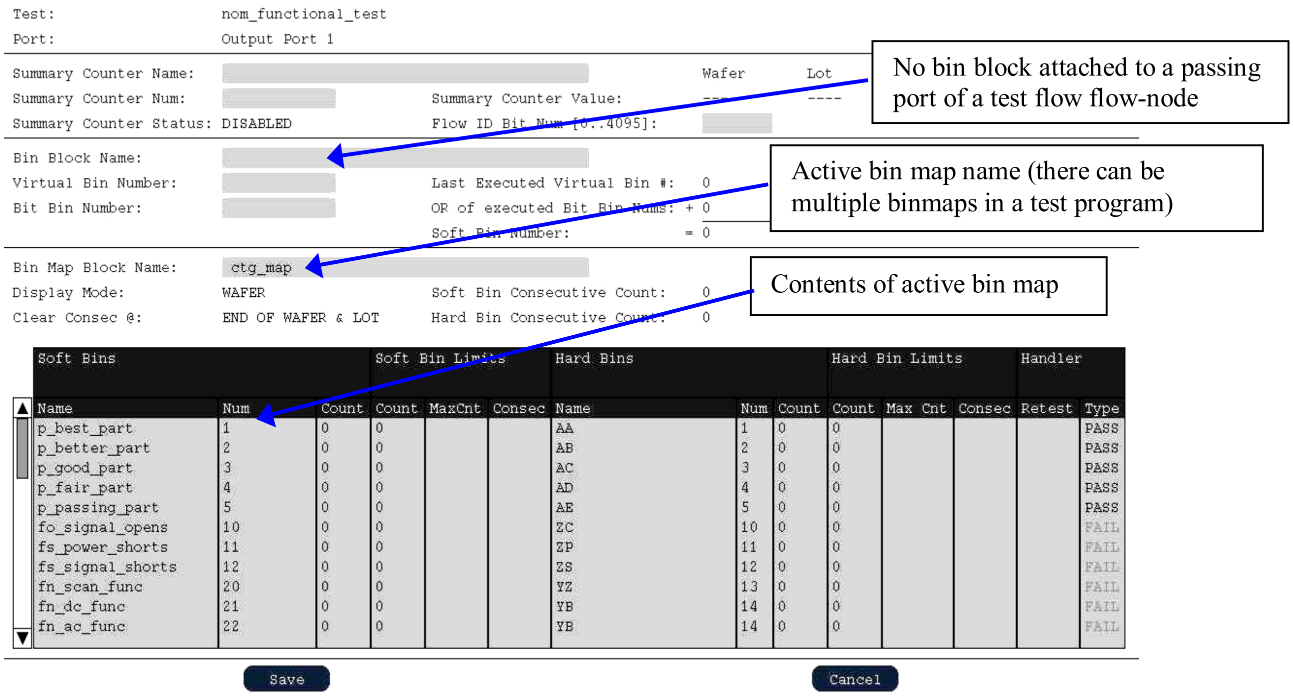

Fig. 5 Schlumberger ITS9000 – Bin/Binmap window opened at passing port – no bin attached

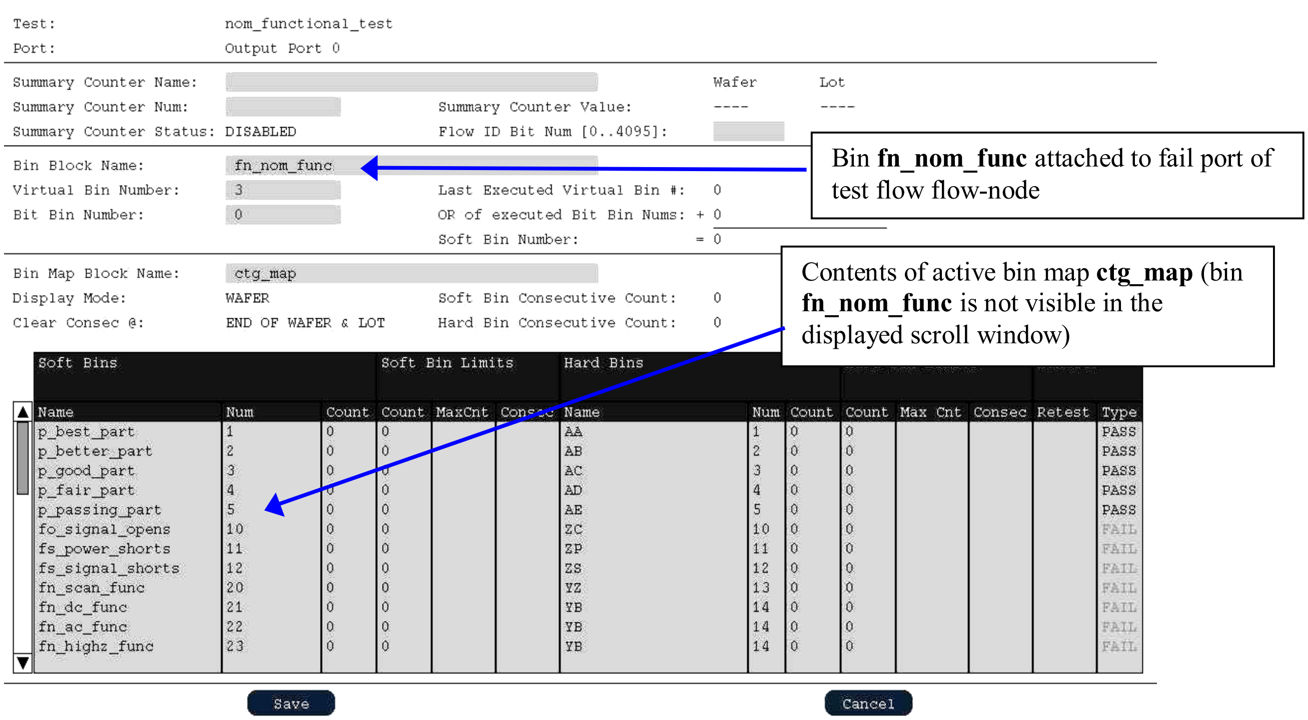

Fig. 6 Schlumberger ITS9000 – Bin/Binmap window opened at failing port – bin fn_nom_func attached

```
bins p_passing_part {
  <08:17:1993 11:49:53>,
  VIRTUAL_BIN = 1,
  BIT_BIN = 0
}; /* end of bins p_passing_part block */

bins fn_nom_func {
  <08:17:1993 09:57:05>,
  VIRTUAL_BIN = 20,
  BIT_BIN = 0
}; /* end of bins fn_nom_func block */
```

Softbin definitions

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

```
bin_map ctg_map {
  <04:16:1993 10:24:26>,
  CLEAR_CONSEC = WAFER_LOT,
  VIRTUAL_MAP = {
    { MAP = 1 > 1, COUNTER = "p_passing_part" },
    { MAP = 10 > 10, COUNTER = "fo_signal_opens" },
    { MAP = 11 > 11, COUNTER = "fs_power_shorts" },
    { MAP = 12 > 12, COUNTER = "fs_signal_shorts" },
    { MAP = 20 > 13, COUNTER = "fn_nom_func" }
  },
  HARDWARE = {
    { BINS = 1, RETEST = 0, COUNTER = "AA", BIN_TYPE = PASS },
    { BINS = 10, RETEST = 0, COUNTER = "ZC", BIN_TYPE = FAIL },
    { BINS = 11, RETEST = 0, COUNTER = "ZP", BIN_TYPE = FAIL },
    { BINS = 12, RETEST = 0, COUNTER = "ZS", BIN_TYPE = FAIL },
    { BINS = 13, RETEST = 0, COUNTER = "YZ", BIN_TYPE = FAIL },
  }
}; /* end of BIN_MAP ctg_map */
```

Softbin to hardbin mappings; definition of softbin name

Hardbin definitions

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

```
segment Test_2 {
  <08:17:1993 09:57:05>,
  X_POS = 282,
  Y_POS = 70,
  ENTRY = W57,
  ICON = "ftestseg_c",
  TOOL = "ftesttool",
  TEST = nom_functional_test,
  RETURN = {
    { POS = S41, FAIL, End_1, BINS = fn_nom_func },
    { POS = E38, PASS, Function_5 }
  }
}; /* end of SEGMENT Test_2 */
```

Exit part of flow-node syntax – fail bin attached to fail port; no bin attached to pass port

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