

OTPL:

DUTFlow FlowMain

```
{
  DUTFlowItem FlowMain_Type(DiagPBFunctionalTestTyp)
  {
    Result 0
    {
      Property PassFail = "Pass";
      IncrementCounters PassCount;
      GoTo MarginFlow;
    }
    Result 1
    {
      Property PassFail = "Fail";
      IncrementCounters FailCount;
      SetBin SoftBins.FailLeakage3GHz;
      Return 1;
    }
  }
}
```

Flow Node

Test Name

Test FunctionalTest DiagPBFunctionalTestTyp

```
{
  PListParam = DiagPBPat;
  TestConditionParam = TC1Typ;
}
```

TestCondition TC1Typ

```
{
  TestConditionGroup = DiagPBTTCG;
  Selector = typ;
}
```

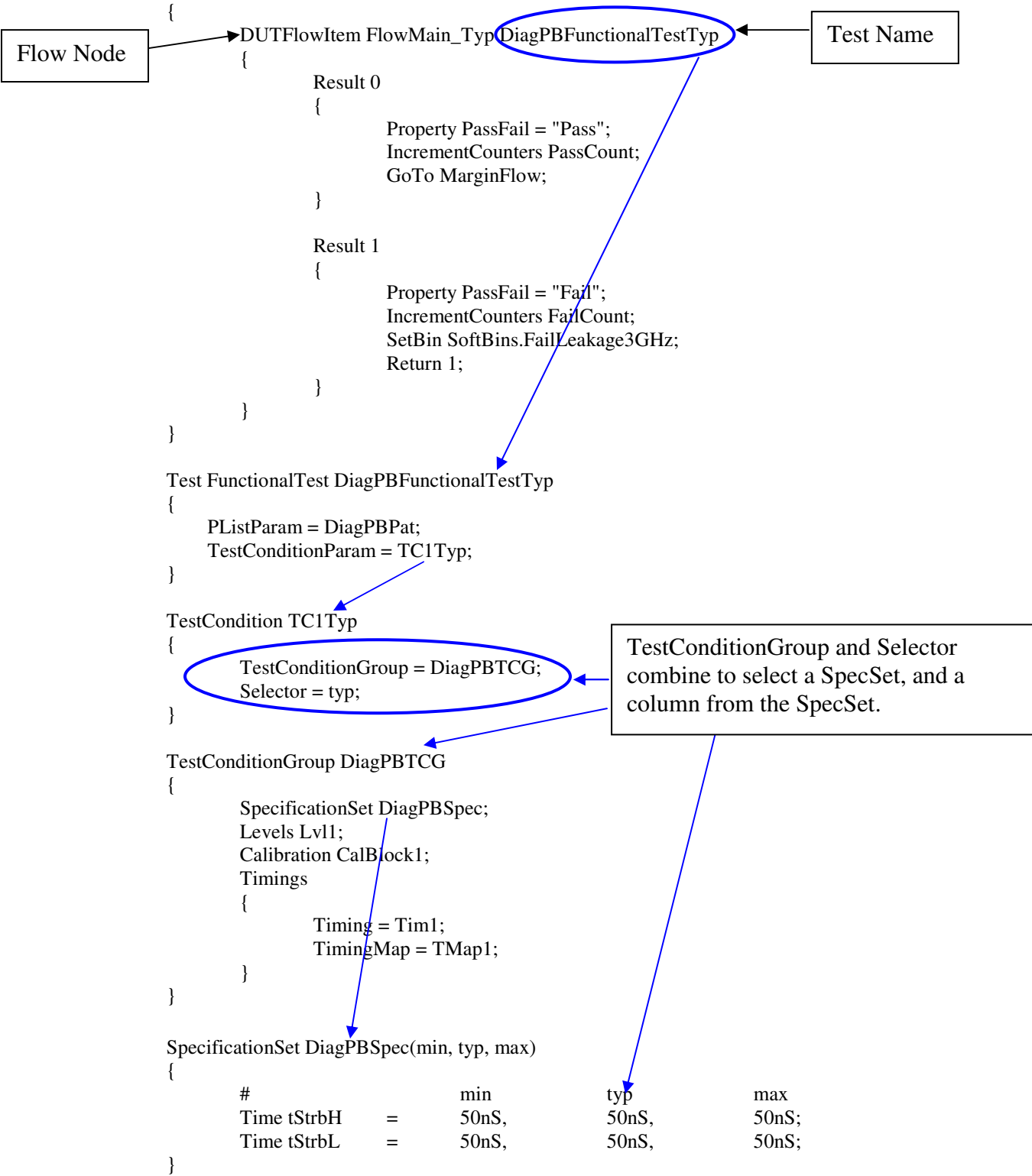
TestConditionGroup and Selector combine to select a SpecSet, and a column from the SpecSet.

TestConditionGroup DiagPBTTCG

```
{
  SpecificationSet DiagPBSpec;
  Levels Lvl1;
  Calibration CalBlock1;
  Timings
  {
    Timing = Tim1;
    TimingMap = TMap1;
  }
}
```

SpecificationSet DiagPBSpec(min, typ, max)

```
{
  #           min           typ           max
  Time tStrbH = 50nS,      50nS,      50nS;
  Time tStrbL = 50nS,      50nS,      50nS;
}
```



Envision:

Spec and Category are specified in FlowNode

```
SubFlow MainFlow_OnStart {  
  Node[17] = FlowNode_ {  
    Port[0] { To = 7; }  
    Port[1] { To = 18; }  
    SpecPairs {  
      ACSpec = Expr { String = "ACSpec.5MHz" Type = INTEGER; }  
      DC_Spec = Expr { String = "DC_Spec.T_25dC"; Type = INTEGER; }  
    }  
    PortSelect = "Gross_Functional.ExecResult";  
    Exec = Gross_Functional;   
  }  
}
```

Flow Node

Test Name

```
Test Gross_Functional {  
  Result = Expr { String = "#"; Mode = Output; }  
  Mask[0] = ACspeed;   
  Entry[0] = Functional_Levels;  
  Entry[1] = Functional_PatSeq;  
  PortExpr[0] = Expr { String = ".Result = tm_rslt:PASS"; }  
  PortExpr[1] = Expr { String = ".Result = tm_rslt:FAIL"; }  
  Title[0] = FuncTest;  
  TestMethod = Ftest;  
  Test_enable[0] = Expr { String = "Seq_en:DEFAULT_EXECUTION"; }  
  Test_pins[0] = Expr { String = "all_pins"; }  
  Test_result[0] = Expr { String = "#"; Mode = Output; }  
  Pattern_index[0] = Expr { String = "Functional_PatSeq.Thread.Single"; }  
}
```

Mask (Selector) specified in Test

```
Mask AC_Speed {  
  Freq = Typ;  
  Per = Typ;  
}
```

Mask selects a set of min/typ/max values

```
Spec ACSpec {  
  Category[0] = AC_5MHz;  
  Category[1] = AC_10MHz;  
  Param[0] = Freq;  
  Param[1] = Per;  
}
```

Spec block defines categories and parameters

```
Param Freq {  
  Type = Hz;  
  Spec = ACSpec;  
  AC_5MHz { Typ = Expr { String = "5MHz"; Type = Hz; } }  
  AC_10MHz { Typ = Expr { String = "10.0MHz"; Type = Hz; } }  
}
```

Parameter definition specifies spec block to which this parameter belongs, and this parameter's values for each category within the spec block

```
Param Per {  
  Type = s;  
  Spec = ACSpec;  
  AC_5MHz { Typ = Expr { String = "1/Freq"; Type = s; } }  
  AC_10MHz { Typ = Expr { String = "1/Freq"; Type = s; } }  
}
```

SpecPairs (in <spec_block_name.category> notation)

| Spec Name | AC_spec_table | Analog_Verification | DC_spec_table |
|-------------------|-------------------------|---------------------|-----------------------|
| Selected Spec.Cat | AC_spec_table.High_5MHz | | DC_spec_table.StdSpec |
| Selected Value | 2 | <none> | 0 |
| Category 0 | LowSpeed | DS_Specs | StdSpec |
| Category 1 | High_7MHz | | |
| Category 2 | High_5MHz | | |
| Category 3 | High_4Mhz | | |

OK Clear All

Figure 16.13: Flow Tool Category Selection Display

ASAP:

```
segment BEGIN {  
  TEST_TYPE = 230,  
  RETURN = {  
    { POS = E42, OTHERWISE, Test_2 }  
  }  
}; /* end of SEGMENT BEGIN */
```

Flow node selects test

```
segment Test_2 {  
  TEST ← nom_functional_test,  
  RETURN = {  
    { POS = S41, FAIL, End_1, BINS = fn_nom_func },  
    { POS = E38, PASS, Function_5 }  
  }  
}; /* end of SEGMENT Test_2 */
```

Test Name

```
ftest nom_functional_test {  
  <12:10:1991 18:11:34>,  
  PINDEF_TABLE = mh1ps,  
  TIMING = mh1_loose,  
  LEVELS = sh1_nomv,  
  PATTERN_LIST = mh1_nom,  
  PRIOR_EVENT_INIT = FALSE,  
  TIMEOUT = 100.000ms  
}; /* end of TEST nom_functional_test */
```

Test selects timing with pre-selected category

```
timing_type mh1_func {  
  <08:09:1993 13:53:06>,  
  PINDEF_TABLE = mh1ps,  
  
  CALCULATION {  
    per_ts0 = spec_per*1.0  
  },  
  
  CATEGORIES { loose, spec, srch },  
  
  PARAMETERS {  
    spec_per = {1.5us,150ns,150ns}  
  },  
  
  TIMING {  
    PERIOD = {  
      "period0", per_ts0  
    }  
  }  
}; /* End of TIMING_TYPE mh1_func */
```

Timing_type block defines equations, parameters, and categories. Timing and levels specs and equations are kept separate (i.e., can't reuse a timing spec for an level). No real concept of Selector (min/typ/max), only category.

```
timing mh1_loose {  
  TIMING_TYPE = mh1_func(loose)  
}; /* end of TIMING: mh1_loose */
```

Timing block selects timing template and category

SmarTest:

```
testfunctions
tf_1:
testfunction_description = "functional";
testfunction_parameters = "functional";
end
test_suites ← Test definition
functional:
  override = 1;
  override_tim_equ_set = 1;
  override_lev_equ_set = 1;
  override_tim_spec_set = 2;
  override_lev_spec_set = 1;
  override_timset = 2;
  override_levset = 1;
  override_seqlbl = "fast";
  override_testf = tf_1;
end
```

```
test_flow
run_and_branch(functional) then Test Name
{
}
else
{
  stop_bin "1", "Functional", , bad,noreprobe,red, 1, over_on;
}

  stop_bin "6", "Good", , good,noreprobe,green, 6, over_on;
end
-----
binning

otherwise bin = "XX", "*** Otherwise Bin *** ", , bad,noreprobe,black, , not_over_on;
end
```

```
EQSP TIM,EQN,#9000002784 EQNSET 1 "gross_func_eqn"
```

“override_tim_equ_set = 1”
statement in test_suite
“functional” selects timing
equation set 1

```
SPECS
```

```
f      [MHz]  
tpdIOx [ns]
```

```
EQUATIONS
```

```
p= 1/f*1000
```

```
TIMINGSET 1 "20MHz" # period and edge delays for a 20MHz test
```

```
period= p
```

```
PINS CP
```

```
e1= 20
```

```
e2= 45
```

```
PINS io_pins
```

```
e1= 2
```

```
e4= 45
```

```
TIMINGSET 2 "50MHz" # period and edge delays for a 50MHz test
```

```
period= p
```

```
PINS CP
```

```
e1= 7
```

```
e2= 17
```

```
PINS io_pins
```

```
e1= 1
```

```
e4= 7 + tpdIOx
```

“override_timset = 2”
statement in test_suite
“functional” selects timing
set 2 within equation set 1

```
# -----  
EQNSET 2 "speed_eqn"
```

```
SPECS
```

```
f      [MHz]  
ts_IOx_CP [ns]  
th_CP_IOx [ns]  
tw_CP    [ns]  
tpd_CP_IOx [ns]
```

```
EQUATIONS
```

```
p= 1/f*1000
```

```
CP_ref= p/2
```

```
TIMINGSET 1 "std"
```

```
period= p
```

```
PINS CP
```

```
e1= CP_ref
```

```
e2= CP_ref + tw_CP
```

```
PINS io_pins
```

```
e1= CP_ref - ts_IOx_CP
```

```
e2= CP_ref + th_CP_IOx
```

```
e4= CP_ref + tpd_CP_IOx
```

EQSP TIM,SPS,#9000002361

EQNSET 1 "gross_func_eqn"

SPECSET 1 "gross_func_specs"

| # | SPECNAME | ****ACTUAL**** | ****MINIMUM**** | ****MAXIMUM**** | UNITS | COMMENT |
|--------|----------|----------------|-----------------|-----------------|-------|---------|
| f | 50 | 30 | 70 | | [MHz] | |
| tpdIOx | 2.5 | | | | [ns] | |

SPECSET 2 "speed_func_specs"

| # | SPECNAME | ****ACTUAL**** | ****MINIMUM**** | ****MAXIMUM**** | UNITS | COMMENT |
|--------|----------|----------------|-----------------|-----------------|-------|---------|
| f | 20 | 15 | 25 | | [MHz] | |
| tpdIOx | 2.5 | | | | [ns] | |

EQNSET 2 "speed_eqn"

SPECSET 1 "speed_specs"

| # | SPECNAME | ****ACTUAL**** | ****MINIMUM**** | ****MAXIMUM**** | UNITS | COMMENT |
|------------|----------|----------------|-----------------|-----------------|-------|---------|
| f | 80 | 50 | 95 | | [MHz] | |
| ts_IOx_CP | 4 | 1.5 | 5 | | [ns] | |
| th_CP_IOx | 1.2 | -1 | 2 | | [ns] | |
| tw_CP | 4 | 0.1 | 5 | | [ns] | |
| tpd_CP_IOx | 8 | 4 | 15 | | [ns] | |

@

Actual: real spec value
Min/Max: used as range for spec searches

"override_tim_spec_set = 2"
statement in test_suite
"functional" selects spec set
2 within equation set 1

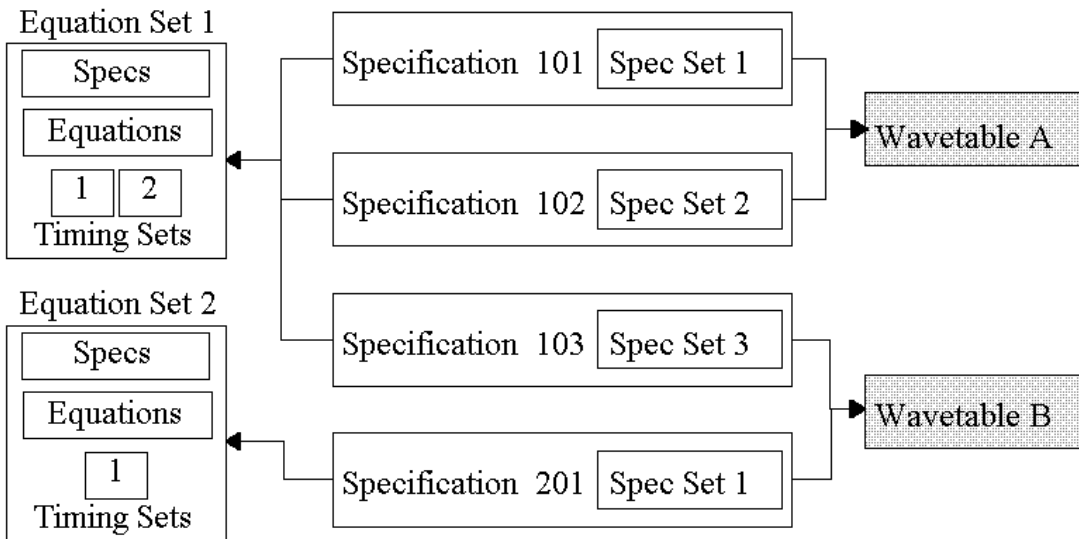


Figure 11: Timing Specifications

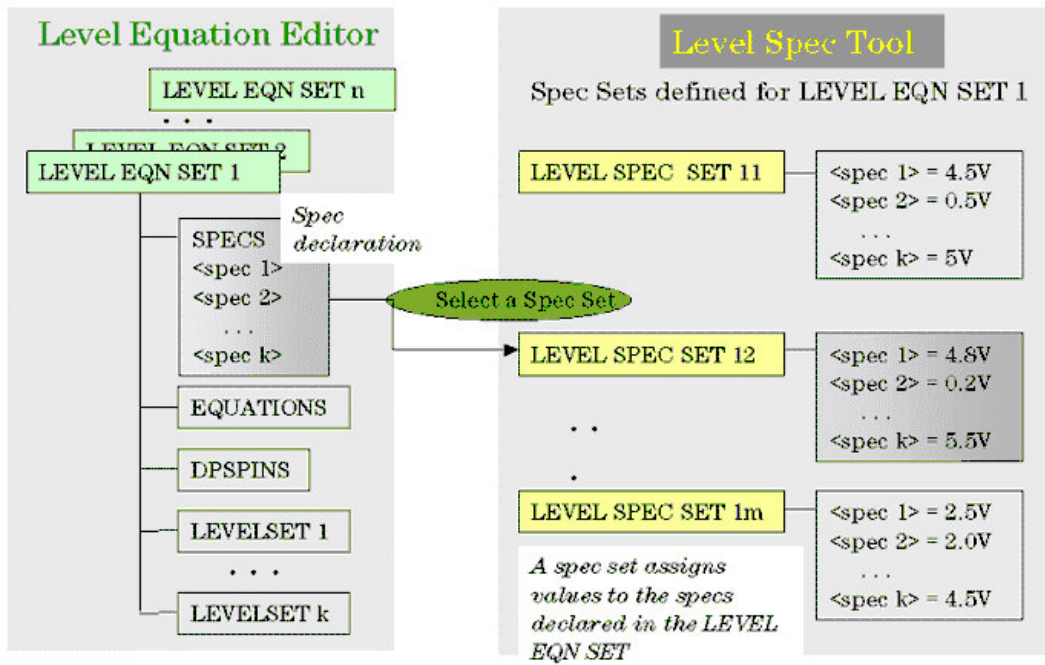


Figure 5: Level Setup Process

Stylus:

```
ProgSeq "Main"{
  Segment 31{
    Type TEST;
    Test acPathTests;
    Title "Path Delay Tests";
    Position 320,320;
    Action { 'Result==Fail'; Bin=31; }
    Action { 'Result==Pass'; Next=51; }
  }
}

TestParams "acPathTests"{
  Method "SpecContext" {"CategorySelectorExpression" Engineering+SpeedGrade;}
  Method "Search"{
    "SignalRefExpr" '_po_-'SO[7]'"-'SO[10]";
    "PatternExec" "acPathExec";
    "ResourceSignals" ""CLK";
    "Search Start" '1/30MHz';
    "Search Stop" '1/130MHz';
    "Upper Limit" '1/InternalFreq';
    "SearchExpression" "acPathDelay";
    "ResultVariable" #;
  }
}
```