Inovys's Stylus submission to P1450.4

Don Organ
Inovys Corporation
don.organ@inovys.com
March 11, 2004

Program Sequence

The Program Sequence is a high level grouping construct. It organizes a collection of Segments into what might be considered a "flow". Each segment generally represents a "flownode" - conceptual a test step. A ProgSeq may be one of the special high-level entry points which are activated under identified situations (such as when the test program is started).

ProgSeq <*SequenceName*> **{**
    (**Segment** <*SegmentNumber*> {
        (**Type TEST|BLOCK|UNCONDITIONAL|BRANCH**;)
        (**Test** <*TestParamsName*>;|
        SequenceName <*SequenceName*>;)
        (**Title** "TitleString";)
        (**Start** True|False;)
        (**Position** <x-offset>,<y-offset>;)
        (**Override** OFF|IGNORE|CONTINUE;)
        (**Execute** ON|OFF|ON_DATALOG;)
        (**Action** (<*ActionTitle*>) {
            ('boolean-expr';)
            (**Next**=<SegmentNumber>;|
            (**Bin**=<BinNumber>;|
            (**Exit**=<ExitStatus>;)
        })* // end of Action
    })+ // end of Segment
} // end of ProgSeq

ProgSeq: A Program Sequence is block that contains a collection of Segments. If given one of certain names (Main, Start, Finish, Load, Unload), then this Program Sequence will be invoked at specific situations as indicated by that start-point.
Only one Segment in a ProgSeq may be identified as Start=True. If none are identified, then the first Segment becomes the Start by default. The Start segment is the first segment executed when the ProgSeq is executed.

A ProgSeq may contain an unlimited number of Segments. The Start one is the only one guaranteed to be executed when the ProgSeq is executed. Subsequent Segments are selected by the Action constructs in the Segment.

Segment: An individual block in a ProgSeq. There are four different types of segments - as indicated by the required Type field.

TEST: Indicates the Segment is a test block. A TEST segment is the only segment type that may contain the Test field - and a TEST segment must contain a Test field.

BLOCK: Indicates that this segment references a ProgSeq. This provides a "sub-flow" capability. A BLOCK segment is the only segment type that may contain the SequenceName field - and a BLOCK segment must contain the SequenceName field.

BRANCH: Indicates a branch block. A BRANCH block has no special semantics - it normally contains several Actions to branch the program flow based on previous results or test options. (The BRANCH segment is displayed with a special shape in the GUI tool.)

UNCONDITIONAL: Indicates an unconditional block. The UNCONDITIONAL segment has no special semantics - but normally has only a single ACTION segment. (The UNCONDITIONAL segment is displayed with a special shape in the GUI tool.)

Override: Default value=OFF.The value of IGNORE indicates that the execution of this segment should ignore any failures; the TestParams will not stop on first failing TestMethod (as is otherwise documented), and a PASS value will be returned even if one or more TestMethods failed.
The value of CONINUE again indicates that the Test Params will not sop on the first failing TestMethod, however a FAIL value will be returned if any TestMethod fails (even though other TestMethods may pass).

Execute: Default value=ON. OFF skips execution of the entire segment. ON_DATALOG will cause the segment to be executed only if datalogging is enabled.

Action: Identifies an Action block in a Segment. A Segment's Action blocks determine the flow sequencing. After the other Segment operations occur (such as executing the test or the sub-block), the first Action block listed is visited. Its associated boolean-expr is evaluated. If there is no boolean-expr, then it defaults to "True". If the boolean-expr evaluates to true, then that Action is selected (and no subsequent Action blocks will be visited in this segment). If the boolean-expr evaluate to false, then that Action block is skipped and the next Action block listed is visited.
In the True case, the associated action (Next, Bin or Exit) is executed. If there is no action, then the default is Next (to the next Segment) - or Exit 1 if this is the last segment in the ProgSeq.
The Next is a goto - it identifies the next Segment to be visited. This next segment must be in the same ProgSeq.
The Bin identifies a softbin. Program execution terminates at that bin.
The Exit is legal only in a ProgSeq that is called from a BLOCK of another Segment (i.e. it is not legal for a topic level ProgSeq). The Exit indicates

that the nested ProgSeq terminates and returns control to the calling BLOCK segment.

TestParams

A TestParams identifies the calling arguments associated with the invocations of one or more TestMethods. A TestParams executes in order, halting if any TestMethod returns a FAIL. The TestParams returns a status to the calling TEST Segment block of PASS if no TestMethods failed, or FAIL if any failed.

TestParams <*TestParamsName*> **{**
    (**Method** <MethodName> **{**
        (<formal-argument-name> <actual-argument>;)*
    })* // end of Method call
} // end of TestParams

TestParams: Identifies the TestParams block. The TestParams block may be invoked from a TEST Segment in a ProgSeq.
A TestParams block has special semantics with regards to the Categories and Selectors of the Spec. By default, all Categories and Selectors are deselected immediately before the TestParams execution begins. Typically, the first TestMethod in the TestParams will be one that sets the Categories and Selectors. Once set, the Categories and Selectors remain set for all subsequent TestMethods in a TestParams. However, if a TestMethod changes a Category or Selector, then that change remains in effect for the remainder of the TestParams - or until changed by a subsequent TestParams.
Note: although not strictly a formal-argument to the TestMethod, there is an pseudo argument named “Enabled” for each TestMethod, with default value of “True”. If Enabled==False, then that TestMethod is not invoked.

Method: Indicates a TestMethod call. The <MethodName> identifies the specific TestMethod. TestMethods are “built-in” to the system and are not definable via this language. A given TestMethod may have a number of parameters. Each parameter must have a formal-argument-name and a type (such as Volts or PatternExec). The Method block indicates the actual argument values (such as after an expression is evaluated) that will be supplied for each identified formal-argument-name.

Bin

There are two related “bin” concepts. A soft-bin - represented by the **Bin** construct, is an abstraction invented in the test program. A soft-bin has a name, a result type (Pass or Fail) and an associated BinNumber which may be used in lot-summary reports. A hard-bin is defined by the **BinMap** construct.

Bin <SoftBinName> **{**
    ProgramResult = “**Pass**|**Fail**”;
    BinNumber = <soft-bin-number>;

}

BinMap

The BinMap provides a mapping from a soft-bin to a hardware bin number. There may be multiple BinMaps in a test program, but only one is selected at a time.Typically, handler equipment operate on hardware bin numbers, and different binmaps may be associated with different types of handler equipment.

BinMap <BinmapName> **{**
    (<SoftBinName> **->** <hard-bin-number>;)*
}

Example Syntax

```
Bin "PassAll"{
     ProgramResult = "Pass";
     BinNumber =    1;
}
Bin "Fail_Opens"{
     ProgramResult = "Fail";
     BinNumber =   11;
}
Bin "Fail_Shorts"{
     ProgramResult = "Fail";
     BinNumber =   12;
}
Bin "Fail_Icc"{
     ProgramResult = "Fail";
     BinNumber =   13;
}
Bin "Fail_InputLeakage"{
     ProgramResult = "Fail";
     BinNumber =   14;
}
Bin "Fail_dcStuckCore"{
     ProgramResult = "Fail";
     BinNumber =   21;
}
Bin "Fail_acPathCore"{
     ProgramResult = "Fail";
     BinNumber =   31;
}
Bin "Fail_Functional"{
     ProgramResult = "Fail";
     BinNumber =   41;
```

```
}
Bin "Fail_PLL"{
     ProgramResult = "Fail";
     BinNumber =   51;
}
Bin "Fail_Iddq"{
     ProgramResult = "Fail";
     BinNumber =   15;
}

ProgSeq "Main"{
     Segment 11{
          Type TEST;
          Test ContactTest;
          Start true;
          Title "Opens / Shorts Test";
          Position 120,80;
          Action FailOpens { 'Result==Fail'; Bin=11; }
          Action FailShorts { 'Result==Fail'; Bin=12; }
          Action  { 'Result==Pass'; Next=7; }
     }
     Segment 7{
          Type BLOCK;
          Title "Device Programming";
          Position 360,80;
          Action 1 { ''; Next=14; }
          SequenceName "DeviceConfig";
     }
     Segment 14{
          Type TEST;
          Test LeakageTests;
          Title "Leakage Tests";
          Position 200,200;
          Action  { 'Result==Fail'; Bin=14; }
          Action  { 'Result==Pass'; Next=21; }
     }
     Segment 21{
          Type TEST;
          Test dcStuckTest;
          Title "DC Stuck At Tests";
          Position 120,320;
          Action  { 'Result==Fail'; Bin=21; }
          Action  { 'Result==Pass'; Next=31; }
     }
     Segment 31{
          Type TEST;
          Test acPathTests;
          Title "Path Delay Tests";
          Position 320,320;
          Action  { 'Result==Fail'; Bin=31; }
          Action  { 'Result==Pass'; Next=51; }
     }
     Segment 41{
          Type TEST;
```

```
          Test FunctionalTests;
          Title "Functional Tests";
          Position 320,440;
          Action  { 'Result==Fail'; Bin=41; }
          Action  { 'Result==Pass'; Next=15; }
     }
     Segment 51{
          Type TEST;
          Test PllTests;
          Title "Phase Lock Loop Tests";
          Position 120,440;
          Action  { 'Result==Fail'; Bin=51; }
          Action  { 'Result==Pass'; Next=41; }
     }
     Segment 15{
          Type TEST;
          Test IddqTest;
          Title "Iddq Tests";
          Position 320,560;
          Action  { 'Result==Pass'; Bin=1; }
          Action  { 'Result==Fail'; Bin=15; }
     }
}
ProgSeq "Start"{
     Segment 1{
          Type UNCONDITIONAL;
          Test ConnectResources;
          Start true;
          Title "Segment <>";
          Position 120,80;
     }
}
ProgSeq "Finish"{
     Segment 2{
          Type UNCONDITIONAL;
          Test RemoveResources;
          Start true;
          Title "Segment <>";
          Position 120,80;
     }
}
ProgSeq "Load"{
     Segment 3{
          Type UNCONDITIONAL;
          Test OnLoad;
          Start true;
          Title "Spec Selection";
          Position 120,80;
     }
}
ProgSeq "Unload"{
}
ProgSeq "DeviceConfig"{
     Segment 101{
```

```
          Type UNCONDITIONAL;
          Test Device_1;
          Title "Stuck Fail";
          Position 120,200;
          Action  { ''; Exit=1; }
     }
     Segment 102{
          Type UNCONDITIONAL;
          Test Device_2;
          Start true;
          Title "Good Device";
          Position 120,320;
          Action  { ''; Exit=1; }
     }
     Segment 103{
          Type UNCONDITIONAL;
          Test Device_3;
          Title "ac Fault";
          Position 120,440;
          Action  { ''; Exit=1; }
     }
     Segment 104{
          Type UNCONDITIONAL;
          Test Device_4;
          Title "Chain Race";
          Position 320,200;
          Action  { ''; Exit=1; }
     }
     Segment 105{
          Type UNCONDITIONAL;
          Test Device_5;
          Title "Stuck Chain";
          Position 320,320;
          Action  { ''; Exit=1; }
     }
     Segment 106{
          Type UNCONDITIONAL;
          Test Device_6;
          Title "Stuck & Broke";
          Position 320,440;
          Action  { ''; Exit=1; }
     }
}
TestParams "ConnectResources"{
     Method "Connect"{"SignalRefExpr" 'all_pins';"Resource" PE;"Action"
Apply;"Delay" '0s';}
}
TestParams "ContactTest"{
     Method "Contact"{"SignalRefExpr" 'all_pins';"Diodes" VssDiode;"Current"
'300uA';"ShortsLimit" '200mV';"OpensLimit" '1V';"Delay" '300us';}
}
TestParams "RemoveResources"{
     Method "Connect"{"SignalRefExpr" 'all_pins';"Resource" PE;"Action"
Remove;"Delay" '0s';}
```

```
     Method "SpecContext"{"CategorySelectorExpression" Engineer-
ing+SpeedGrade;}
     Method "Setup"{"DCLevels" "Zero";}
}
TestParams "Load_dcFault1"{
     Method "Setup"{"DCLevels" "NomLevels";}
     Method "Delay"{"Delay" '50ms';}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Apply;"Delay" '1ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Low;}
     Method "Delay"{"Delay" '2ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Off;}
     Method "Delay"{"Delay" '2ms';}
     Method "Pattern"{"SignalRefExpr" 'JTAG_PINS-TDO';"PatternExec"
FPGA_dcFault;}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Remove;"Delay" '0s';}
}
TestParams "dcStuckTest"{
     Method "Pattern"{"SignalRefExpr" '"_so"';"PatternExec" "dcStuckAtExec";}
}
TestParams "LeakageTests"{
     Method "Current"{"SignalRefExpr" '"_in"';"Force Voltage" '0V';"Min I" '-
10uA';"Max I" '10uA';"Delay" '800us';"Measurement Sequence" MBB;"Measurement
Mode" Static;"Max Expected Current" '100uA';}
     Method "Current"{"SignalRefExpr" '"_in"';"Force Voltage" '3V';"Min I" '-
10uA';"Max I" '10uA';"Delay" '400us';"Measurement Sequence" MBB;"Measurement
Mode" Static;"Max Expected Current" '100uA';}
}
TestParams "Device_2"{
      ExecutionMode IgnoreAllFails;
     Method "Setup"{"DCLevels" "NomLevels";}
     Method "Delay"{"Delay" '50ms';}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Apply;"Delay" '1ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Low;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Off;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "Voltage"{"SignalRefExpr" '"FPGA_OK"';"Force Current" '10uA';"Min
V" '2V';"Delay" '2ms';}
     Method "Delay"{"Delay" '2ms';}
     Method "Pattern"{"SignalRefExpr" 'JTAG_PINS-TDO';"PatternExec"
"Exec_GoodConfig";"BurstMode" True;}
     Method "Delay"{"Delay" '2ms';}
     Method "Voltage"{"SignalRefExpr" '"FPGA_OK"';"Force Current" '10uA';"Min
V" '-1V';"Max V" '1V';"Delay" '2ms';}
     Method "Delay"{"Delay" '2ms';}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Remove;"Delay" '0s';}
}
TestParams "Device_5"{
```

```
     Method "Setup"{"DCLevels" "NomLevels";}
     Method "Delay"{"Delay" '50ms';}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Apply;"Delay" '1ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Low;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Off;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "Pattern"{"SignalRefExpr" 'JTAG_PINS-TDO';"PatternExec"
"Exec_scstuck";"BurstMode" True;}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Remove;"Delay" '0s';}
}
TestParams "Device_4"{
     Method "Setup"{"DCLevels" "NomLevels";}
     Method "Delay"{"Delay" '50ms';}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Apply;"Delay" '1ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Low;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Off;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "Pattern"{"SignalRefExpr" 'JTAG_PINS-TDO';"PatternExec"
"Exec_scrace";"BurstMode" True;}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Remove;"Delay" '0s';}
}
TestParams "Device_1"{
     Method "Setup"{"DCLevels" "NomLevels";}
     Method "Delay"{"Delay" '50ms';}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Apply;"Delay" '1ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Low;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Off;"Comparator-
Mode" NoChange;}
     Method "Delay"{"Delay" '2ms';}
     Method "Pattern"{"SignalRefExpr" 'JTAG_PINS-TDO';"PatternExec"
"Exec_stuck";"BurstMode" True;}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Remove;"Delay" '0s';}
}
TestParams "Device_6"{
     Method "Setup"{"DCLevels" "NomLevels";}
     Method "Delay"{"Delay" '50ms';}
     Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Apply;"Delay" '1ms';}
     Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Low;"Comparator-
Mode" NoChange;}
```

```
    Method "Delay"{"Delay" '2ms';}
    Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Off;"Comparator-
Mode" NoChange;}
    Method "Delay"{"Delay" '2ms';}
    Method "Pattern"{"SignalRefExpr" 'JTAG_PINS-TDO';"PatternExec"
"Exec_stuck_broke";"BurstMode" True;}
    Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Remove;"Delay" '0s';}
}
TestParams "acPathTests"{
    Method "SpecContext"{"CategorySelectorExpression" Engineer-
ing+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" "acPathDe-
lay";"ResultVariable" #;}
}
TestParams "Device_3"{
    Method "Connect"{"SignalRefExpr" 'DELAYENABLE';"Resource" PE;"Action"
Apply;"Delay" '0s';}
    Method "PinMode"{"SignalRefExpr" 'DELAYENABLE';"DriverMode" High;"Compar-
atorMode" NoChange;}
    Method "Setup"{"DCLevels" "NomLevels";}
    Method "Delay"{"Delay" '50ms';}
    Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Apply;"Delay" '1ms';}
    Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Low;"Comparator-
Mode" NoChange;}
    Method "Delay"{"Delay" '2ms';}
    Method "PinMode"{"SignalRefExpr" 'PROGRAM';"DriverMode" Off;"Comparator-
Mode" NoChange;}
    Method "Delay"{"Delay" '2ms';}
    Method "Pattern"{"SignalRefExpr" 'JTAG_PINS-TDO';"PatternExec"
"Exec_GoodConfig";"BurstMode" True;}
    Method "Connect"{"SignalRefExpr" 'JTAG_PINS+PROGRAM';"Resource"
PE;"Action" Remove;"Delay" '0s';}
}
TestParams "FunctionalTests"{
    Method "SpecContext"{"CategorySelectorExpression" Engineer-
ing+SpeedGrade;}
    Method "Pattern"{"SignalRefExpr" 'fpins';"PatternExec"
"Exec_functional";}
}
TestParams "PllTests"{
    Method "SpecContext"{"CategorySelectorExpression" Engineer-
ing+SpeedGrade;}
    Method "Frequency"{"SignalRefExpr" '"ClkOut"';"Minimum Freq" '100MHz*(1-
0.0002)';"Maximum Freq" '100MHz*(1+0.0002)';"Measure Mode" Continuous;"Pat-
ternExec" "Exec_pll";"Output Mode" Average;"ResultVariable" #;}
}
TestParams "OnLoad"{
    Method "SpecContext"{"CategorySelectorExpression" Engineer-
ing+SpeedGrade;}
}
```

```
TestParams "IddqTest"{
     Method "SpecContext"{"CategorySelectorExpression" Engineer-
ing+SpeedGrade;}
     Method "IDDQ_PMU"{"Test Pins" 'fpins';"Meas Pin" 'IddqPin';"DPS Pin"
'VCCINT';"User Bits" 'Kidd';"PatternExec" "Exec_Iddq";"Measurement Mode"
Dynamic;"Delay" '2ms';"Max I" '80uA';}
}
```