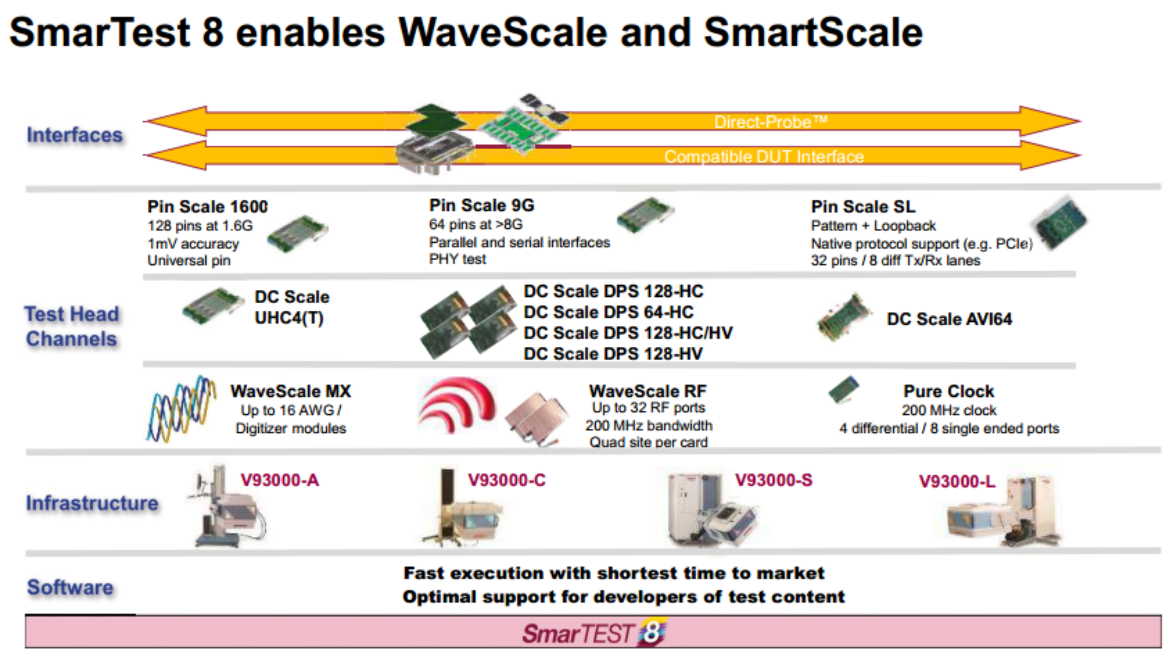
V93000 SmarTest 8 Basic User Training

Version 8.2.5

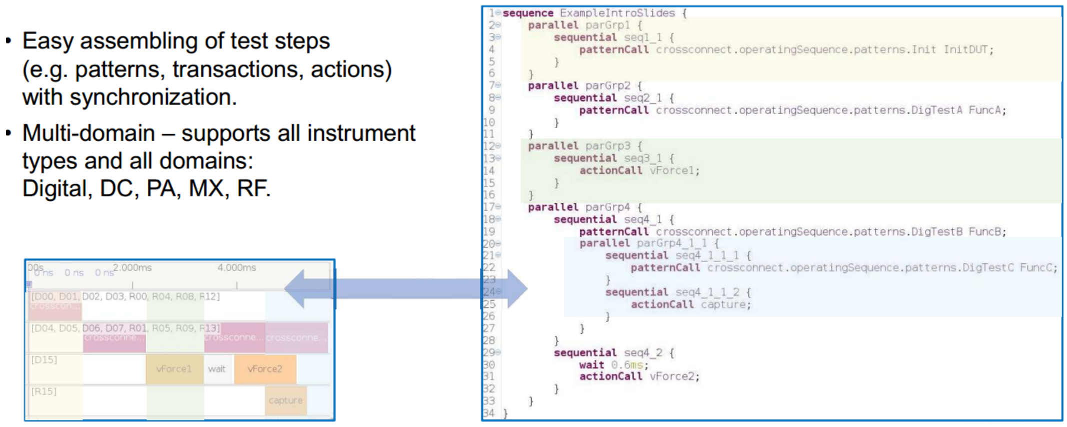
Introduction/Agenda

Lecture 01: Introduction, Concepts and Software overview

- Key Objectives of SmarTest 8
    1. Enable the **shortest** time from **tape-out to volume production**.
    2. Enable **best test program development efficiency**.
    3. Offer the **fastest test throughput** for manufacturing test of complex SOC/SiP/TSV and also small low complexity devices.



- **SmarTest 8 Values**
    1. Simple synchronization across domains



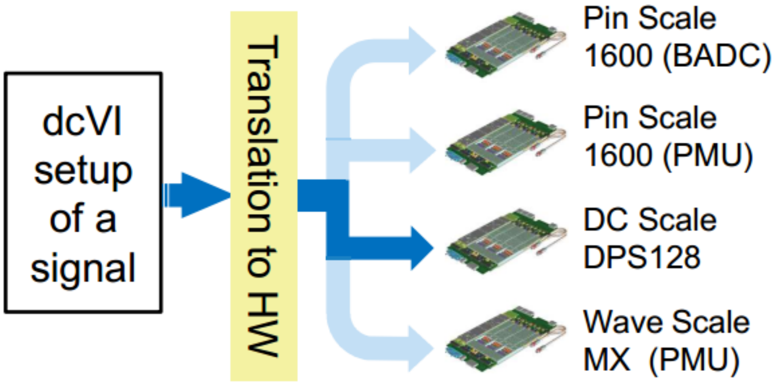
- Easy assembling of test steps (e.g. patterns, transactions, actions) with synchronization.
- Multi-domain – supports all instrument types and all domains: Digital, DC, PA, MX, RF.

```
sequence ExampleIntroSlides {
    parallel parGrp1 {
        sequential seq1_1 {
            patternCall crossconnect.operatingSequence.patterns.Init InitDUT;
        }
    }
    parallel parGrp2 {
        sequential seq2_1 {
            patternCall crossconnect.operatingSequence.patterns.DigTestA FuncA;
        }
    }
    parallel parGrp3 {
        sequential seq3_1 {
            actionCall vForce1;
        }
    }
    parallel parGrp4 {
        sequential seq4_1 {
            patternCall crossconnect.operatingSequence.patterns.DigTestB FuncB;
            parallel parGrp4_1_1 {
                sequential seq4_1_1_1 {
                    patternCall crossconnect.operatingSequence.patterns.DigTestC FuncC;
                }
                sequential seq4_1_1_2 {
                    actionCall capture;
                }
            }
        }
        sequential seq4_2 {
            wait 0.6ms;
            actionCall vForce2;
        }
    }
}
```

2.000ms
4.000ms
[D00, D01, D02, D03, R00, R04, R08, R12]
[D04, D05, D06, D07, R01, R05, R09, R13]
[D15]
[R15]
vForce1
wait
vForce2
capture

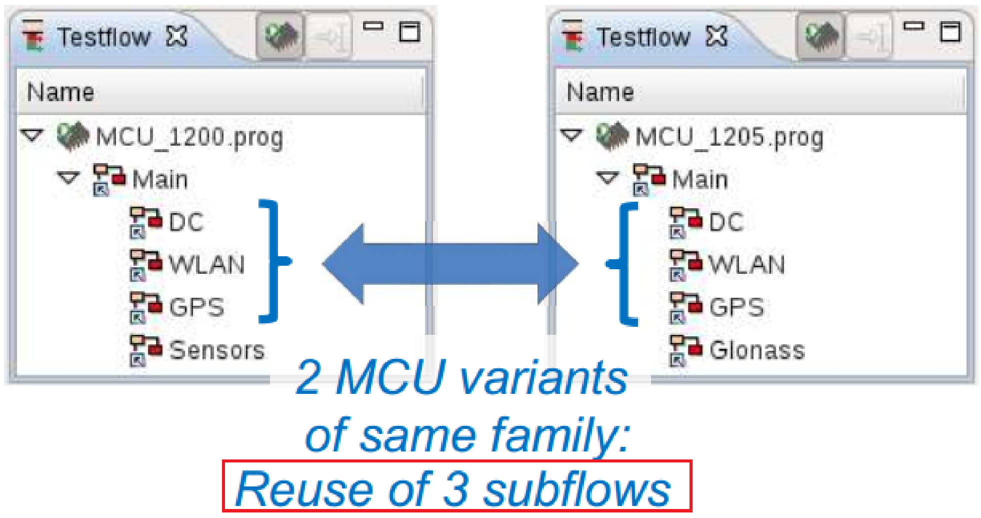
2. Unified setups and tools

> **Hardware Abstraction**: awg, clock, dcVI, digInout...
>
> **Multi Domain Tools**: Timing Debug, digital/analog/RF/DC/PA...



3. Reuse

> Every node of a testflow can be a test suite or another testflow.
>
> Every testflow can have input and output parameters.



4. Collaborative development

> • Flexible file structure for setup files • Allows a dedicated file for a setup entity • Integrated support for SVN and GIT • Links to external setup files

5. Assisted setup generation

> SmarTest Setup Format (SSF)
>
> 快捷键：ctrl-space, F3

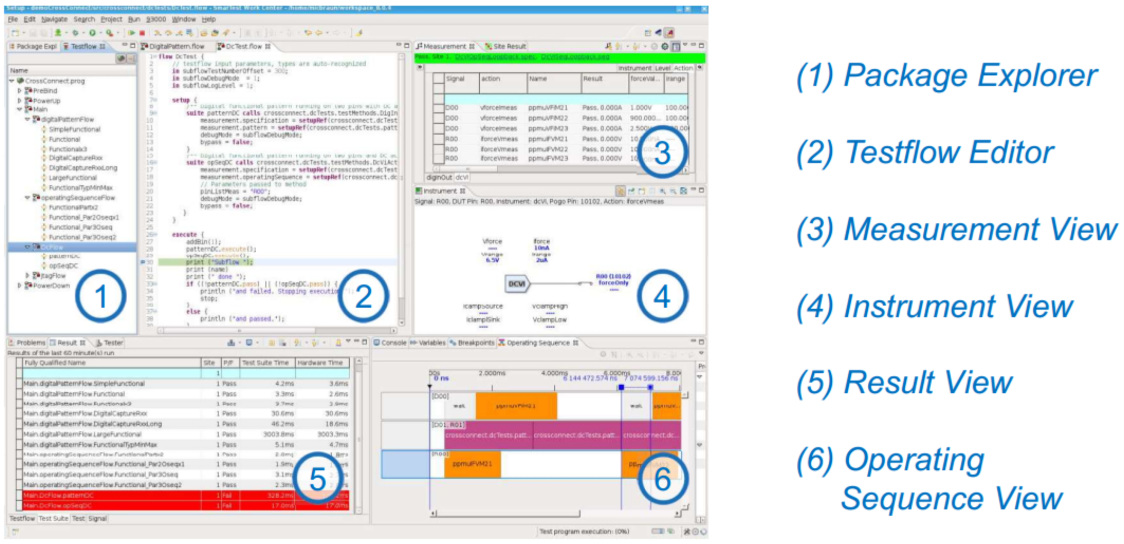
> 视图切换：例如 pin定义

6. Smart test methods

> 单site扩展成多site容易
>
> 支持特定site的数据处理
>
> 结果支持后台运算

7. Java

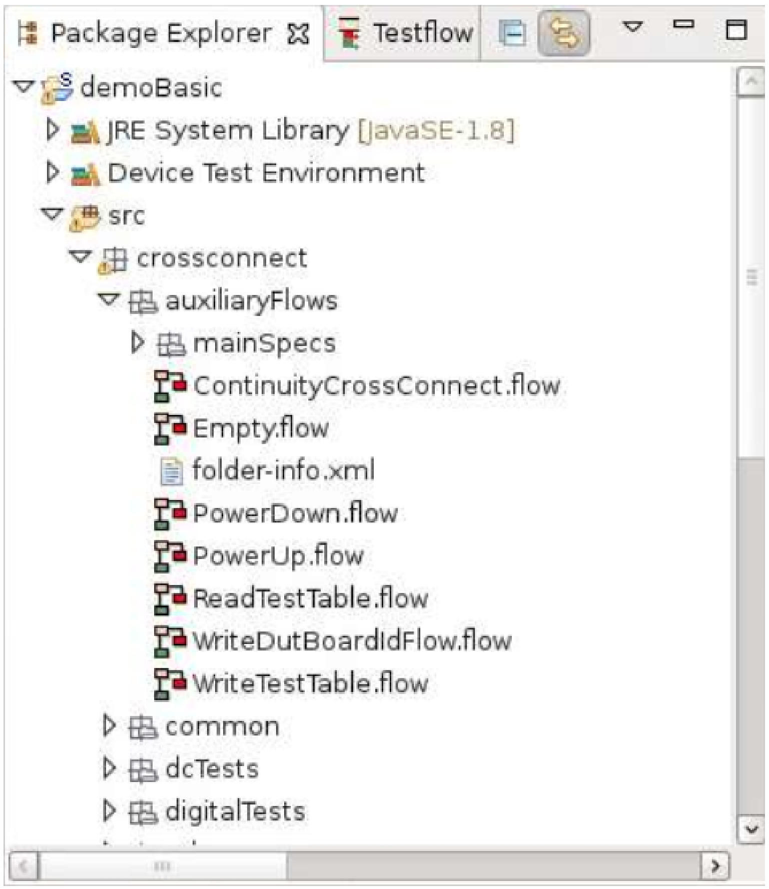
- graphical tools



Lecture 02: SmarTest Projects

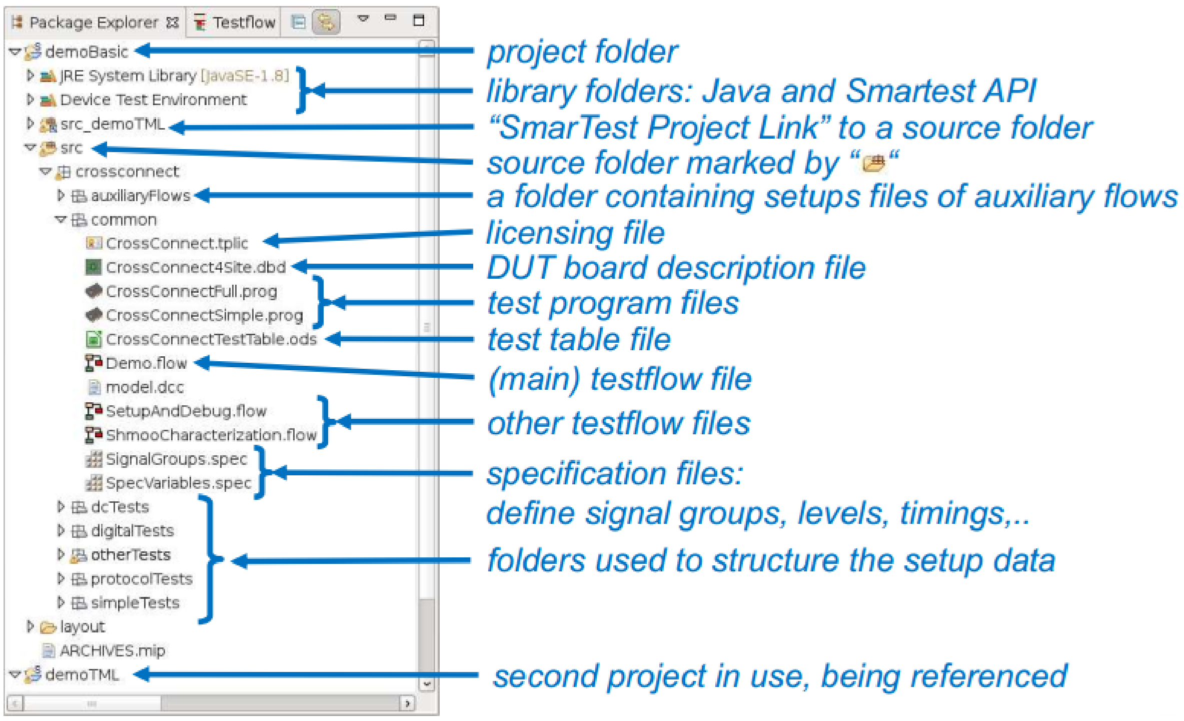
- SmarTest 8 Project

> – Java Libraries – V93000 System Libraries – Source Folders – Configuration Files – Compile/Build Directories



- Project Setup

> 1. Creat new SmarTest Project
> 2. Import SmarTest Project

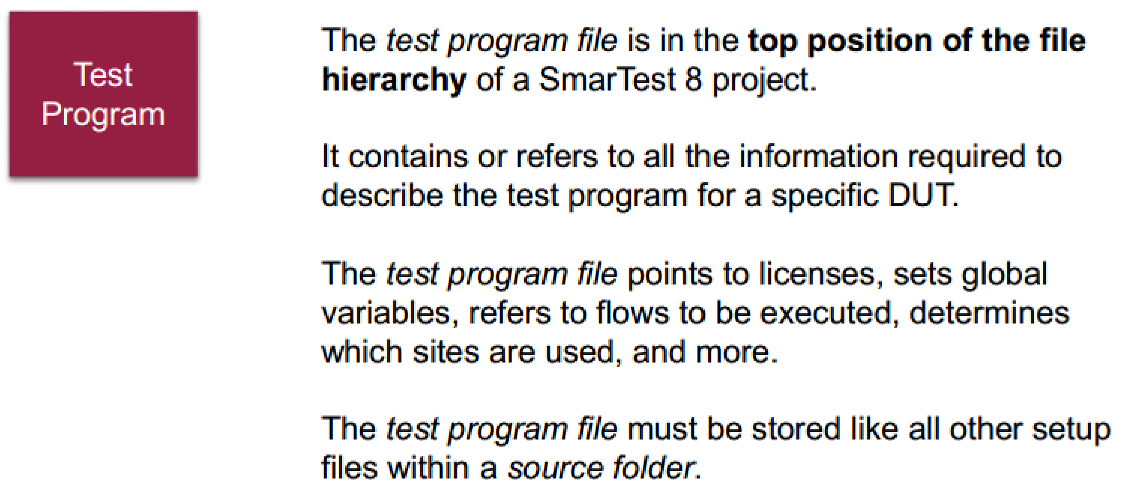


Lab 01: Preparation and Prerequisites

• Start SmarTest 8 with the correct model file. • Create a workspace • Change the view • Import the SmarTest 8 project (i.e. test program) used in the lab

Lecture 03: Test Program File

- **Test Program File**


Test Program

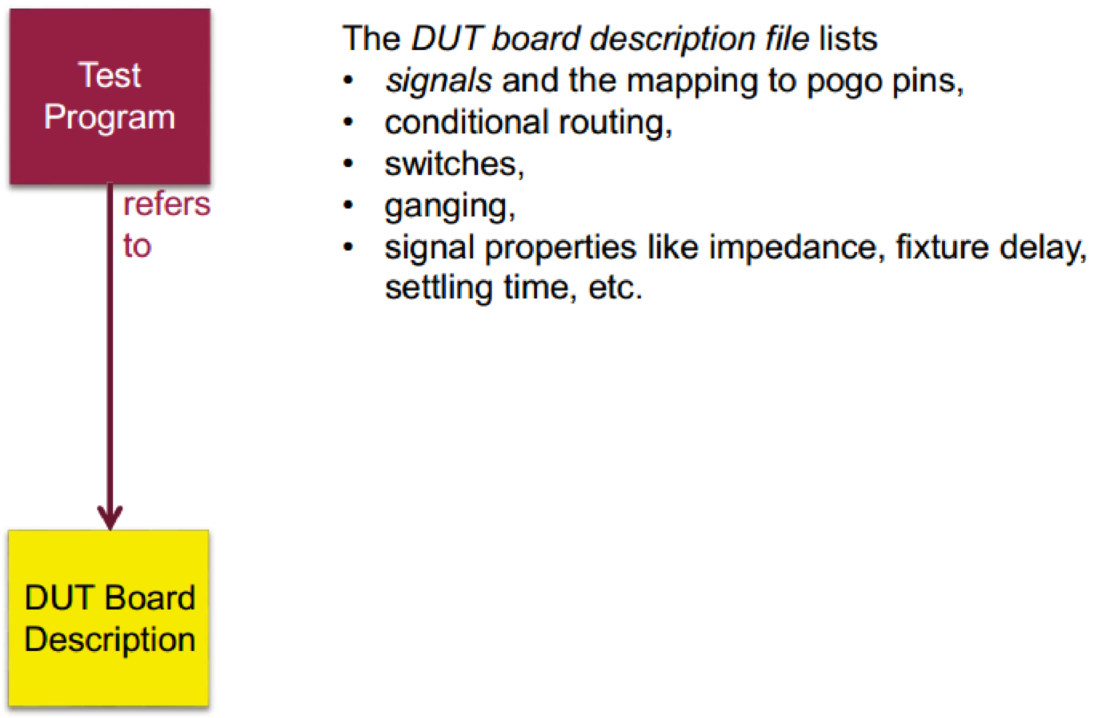
The *test program file* is in the **top position of the file hierarchy** of a SmarTest 8 project.

It contains or refers to all the information required to describe the test program for a specific DUT.

The *test program file* points to licenses, sets global variables, refers to flows to be executed, determines which sites are used, and more.

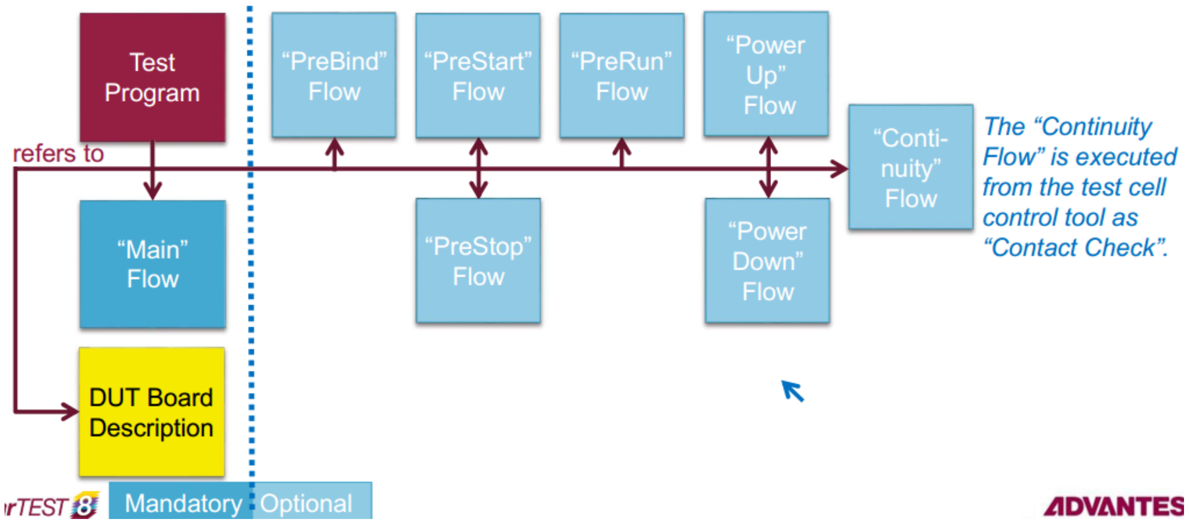
The *test program file* must be stored like all other setup files within a *source folder*.

- **DUT Board Description File**

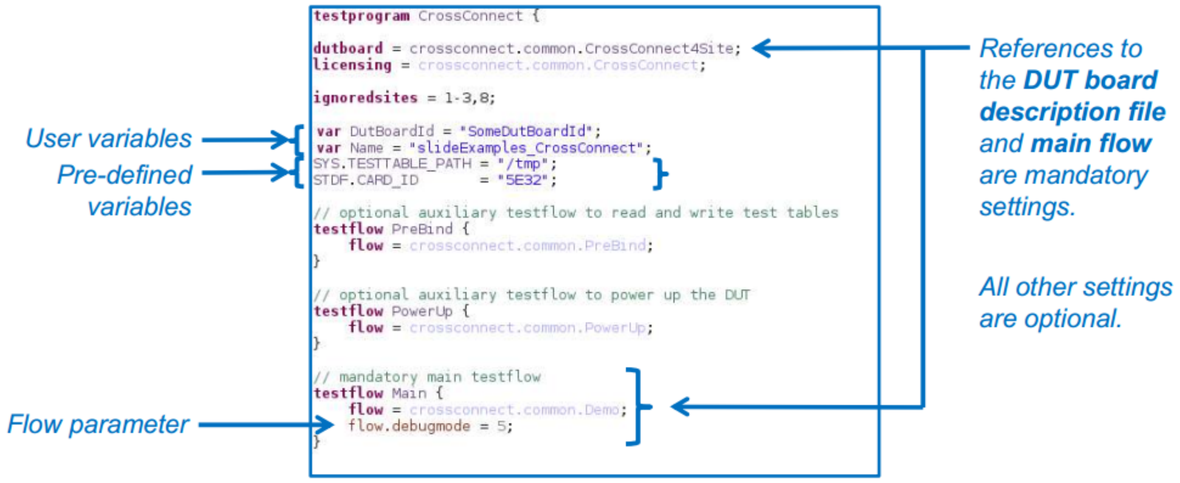


The *DUT board description file* lists
- *signals* and the mapping to pogo pins,
- conditional routing,
- switches,
- ganging,
- signal properties like impedance, fixture delay, settling time, etc.

- **Main Flow File**
- **Auxiliary Flow Files**



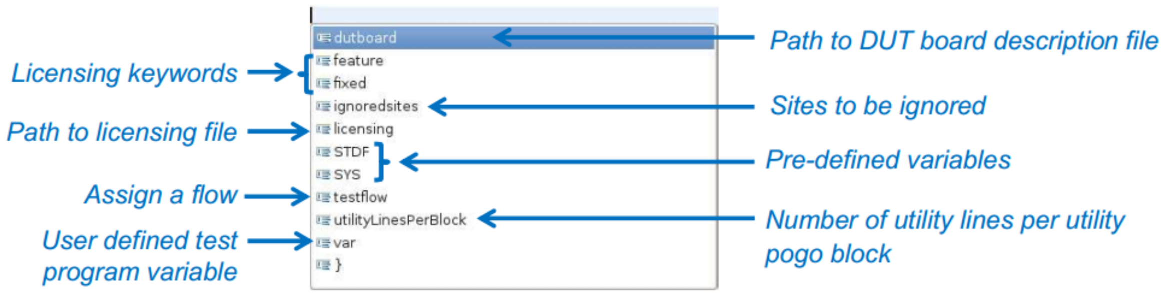
example


User variables
Pre-defined variables
Flow parameter
References to the DUT board description file and main flow are mandatory settings.
All other settings are optional.

```
testprogram CrossConnect {

dutboard = crossconnect.common.CrossConnect4Site;
licensing = crossconnect.common.CrossConnect;

ignoredsites = 1-3,8;

 var DutBoardId = "SomeDutBoardId";
 var Name = "slideExamples_CrossConnect";
SYS.TESTTABLE_PATH = "/tmp";
STDF.CARD_ID       = "5E32";

// optional auxiliary testflow to read and write test tables
testflow PreBind {
    flow = crossconnect.common.PreBind;
}

// optional auxiliary testflow to power up the DUT
testflow PowerUp {
    flow = crossconnect.common.PowerUp;
}

// mandatory main testflow
testflow Main {
    flow = crossconnect.common.Demo;
    flow.debugmode = 5;
}
```

Ctrl+Space

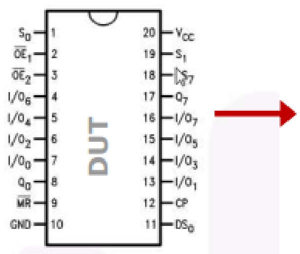
Available Key Words in the Test Program File



Lecture 04: DUT Board Description File

- example

DUT data sheet excerpt



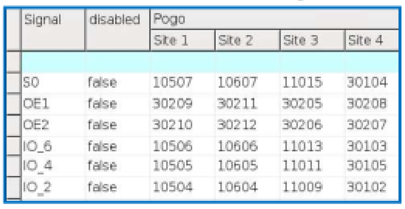
Pin Description

Pin Names	Description
CP	Clock Pulse Input
DS_0	Serial Data Input for Right Shift
DS_7	Serial Data Input for Left Shift
S_0, S_1	Mode Select Inputs
$\overline{MR}$	Asynchronous Master Reset
$\overline{OE}_1$, $\overline{OE}_2$	3-STATE Output Enable Inputs
I/O_0–I/O_7	Parallel Data Inputs or 3-STATE Parallel Outputs
Q_0, Q_7	Serial Outputs

DUT board Signal list: 4 sites

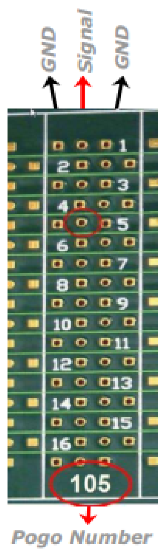
	Name	DUT1	DUT2	DUT3	DUT4
1	S0	D10507	D10607	D11015	D30104
2	OE1	D30209	D30211	D30205	D30208
3	OE2	D30210	D30212	D30206	D30207
4	I/O_6	D10506	D10606	D11013	D30103
5	I/O_4	D10505	D10605	D11011	D30105
6	I/O_2	D10504	D10604	D11009	D30102

SmarTest
DUT board description



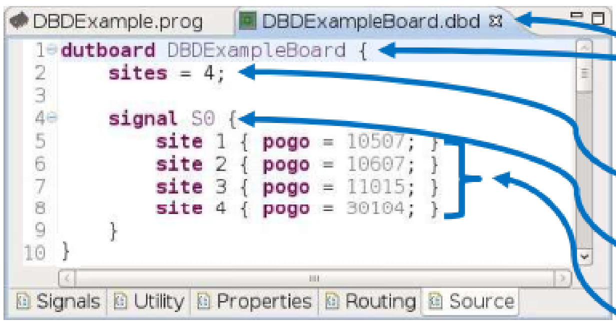
Signal	disabled	Pogo			
		Site 1	Site 2	Site 3	Site 4
S0	false	10507	10607	11015	30104
OE1	false	30209	30211	30205	30208
OE2	false	30210	30212	30206	30207
IO_6	false	10506	10606	11013	30103
IO_4	false	10505	10605	11011	30105
IO_2	false	10504	10604	11009	30102

DUT board allocation



Trace on interface-PCB from Pogo-block to device pin

> dutboard = crossconnect.common.BasicLab;



```
dutboard DBDExampleBoard {
    sites = 4;

    signal S0 {
        site 1 { pogo = 10507; }
        site 2 { pogo = 10607; }
        site 3 { pogo = 11015; }
        site 4 { pogo = 30104; }
    }
}
```

Name of the DUT board description and its file name must match

The total number of sites

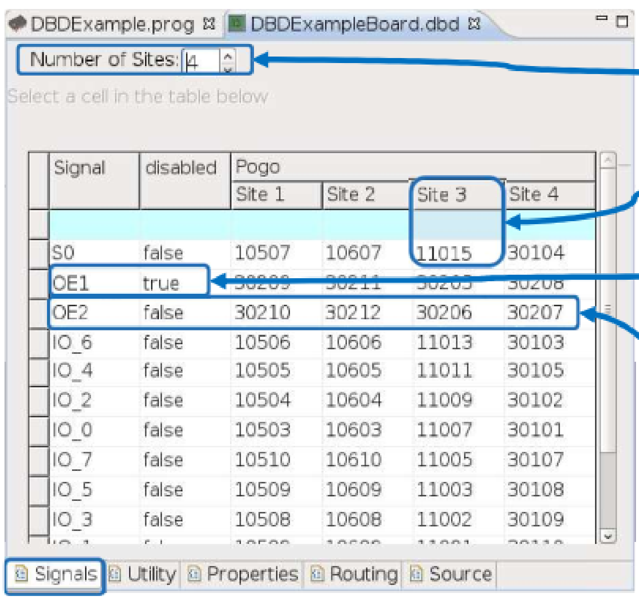
Signal name

The syntax for assigning a pogo is the same for the different types of channels:

- *digital,*
- *analog,*
- *RF,*
- *power supplies.*

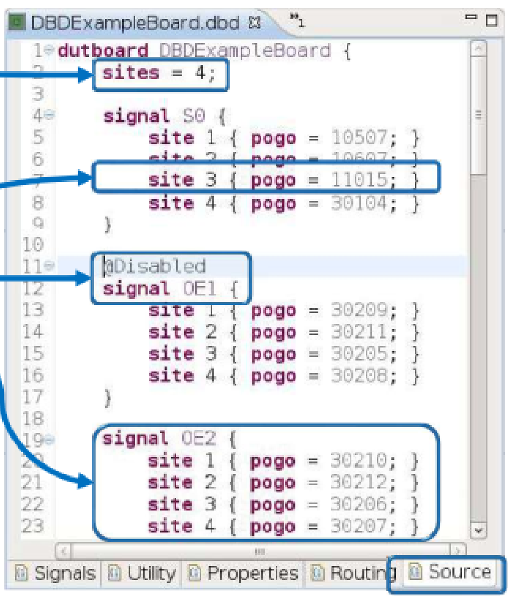
"`dutboard`", "`sites`", "`signal`", "`site`" and "`pogo`" are keywords of the syntax.

The *signal* "S0" is a logical input to the DUT or a logical output from the DUT.

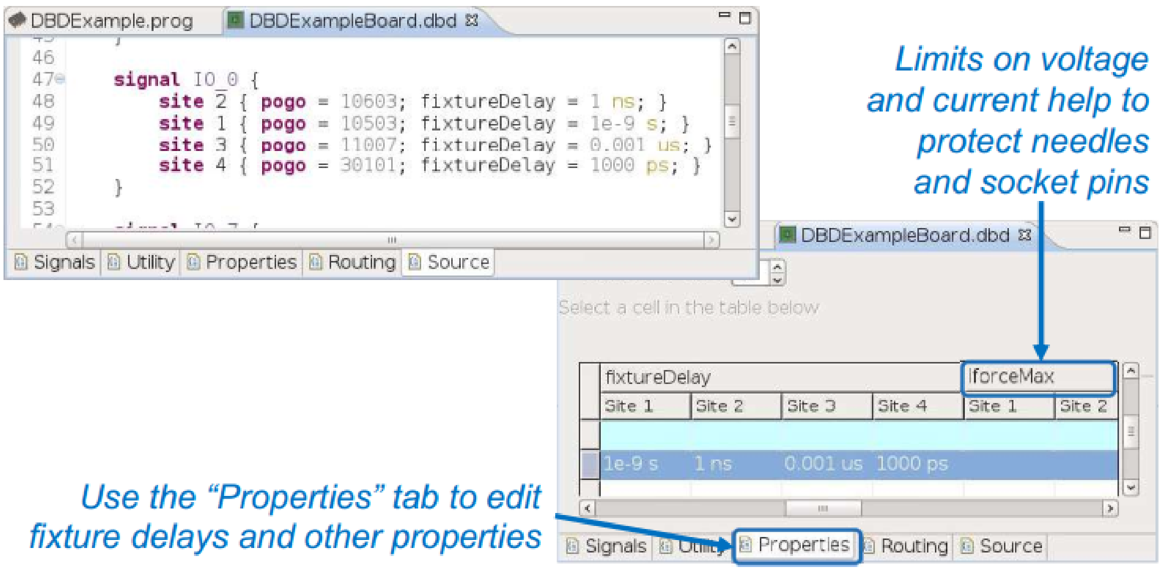


Number of Sites: 4

Signal	disabled	Pogo			
		Site 1	Site 2	Site 3	Site 4
S0	false	10507	10607	11015	30104
OE1	true	30209	30211	30205	30208
OE2	false	30210	30212	30206	30207
IO_6	false	10506	10606	11013	30103
IO_4	false	10505	10605	11011	30105
IO_2	false	10504	10604	11009	30102
IO_0	false	10503	10603	11007	30101
IO_7	false	10510	10610	11005	30107
IO_5	false	10509	10609	11003	30108
IO_3	false	10508	10608	11002	30109



```
dutboard DBDExampleBoard {
    sites = 4;

    signal S0 {
        site 1 { pogo = 10507; }
        site 2 { pogo = 10607; }
        site 3 { pogo = 11015; }
        site 4 { pogo = 30104; }
    }

    @Disabled
    signal OE1 {
        site 1 { pogo = 30209; }
        site 2 { pogo = 30211; }
        site 3 { pogo = 30205; }
        site 4 { pogo = 30208; }
    }

    signal OE2 {
        site 1 { pogo = 30210; }
        site 2 { pogo = 30212; }
        site 3 { pogo = 30206; }
        site 4 { pogo = 30207; }
```

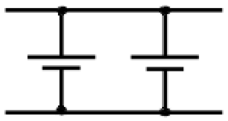

DBDExample.prog
DBDExampleBoard.dbd
Signals
Utility
Properties
Routing
Source
Limits on voltage and current help to protect needles and socket pins
Select a cell in the table below
Use the “Properties” tab to edit fixture delays and other properties

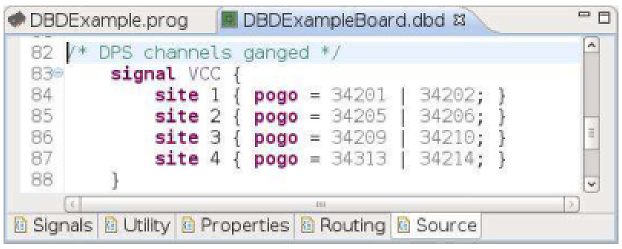
```
    signal IO_0 {
        site 2 { pogo = 10603; fixtureDelay = 1 ns; }
        site 1 { pogo = 10503; fixtureDelay = 1e-9 s; }
        site 3 { pogo = 11007; fixtureDelay = 0.001 us; }
        site 4 { pogo = 30101; fixtureDelay = 1000 ps; }
    }
```

fixtureDelay				forceMax	
Site 1	Site 2	Site 3	Site 4	Site 1	Site 2
1e-9 s	1 ns	0.001 us	1000 ps		

- Ganging DPS Channels

Syntax:

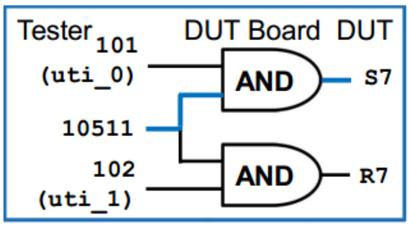


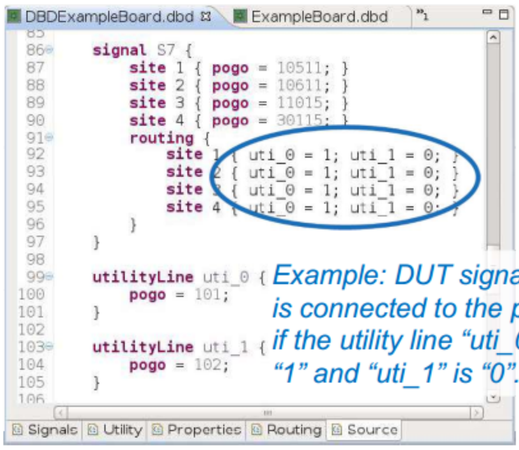


```
/* DPS channels ganged */
    signal VCC {
        site 1 { pogo = 34201 | 34202; }
        site 2 { pogo = 34205 | 34206; }
        site 3 { pogo = 34209 | 34210; }
        site 4 { pogo = 34313 | 34214; }
    }
```

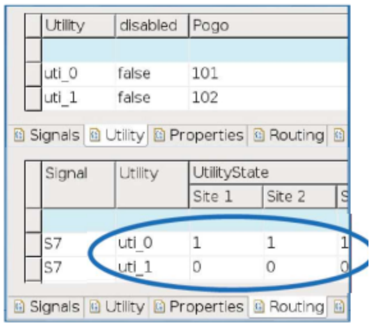
- DUT Boards with Switches

Often DUT boards make use of switches, for example to configure the routing of a single pogo to multiple DUT pins.




DBDExampleBoard.dbd
ExampleBoard.dbd
Signals
Utility
Properties
Routing
Source
Example: DUT signal “S7” is connected to the pogos if the utility line “uti_0” is “1” and “uti_1” is “0”.

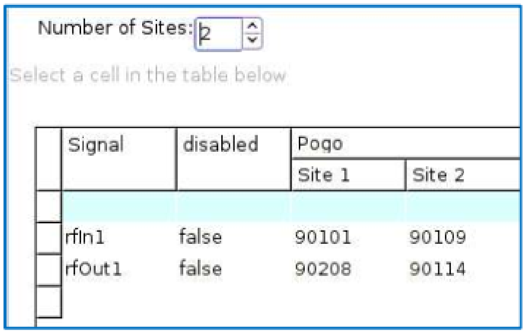
```
    signal S7 {
        site 1 { pogo = 10511; }
        site 2 { pogo = 10611; }
        site 3 { pogo = 11015; }
        site 4 { pogo = 30115; }
        routing {
            site 1 { uti_0 = 1; uti_1 = 0; }
            site 2 { uti_0 = 1; uti_1 = 0; }
            site 3 { uti_0 = 1; uti_1 = 0; }
            site 4 { uti_0 = 1; uti_1 = 0; }
        }
    }

    utilityLine uti_0 {
        pogo = 101;
    }

    utilityLine uti_1 {
        pogo = 102;
    }
```



Utility	disabled	Pogo
uti_0	false	101
uti_1	false	102

Signal	Utility	UtilityState		
		Site 1	Site 2	S
S7	uti_0	1	1	1
S7	uti_1	0	0	0

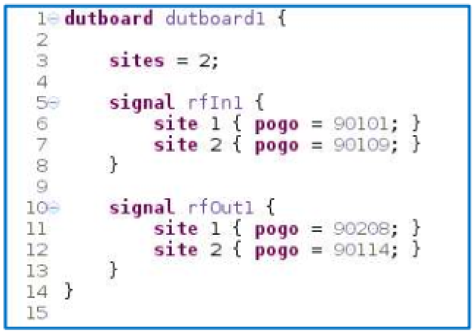
- RF Pins Setup



Number of Sites: 2

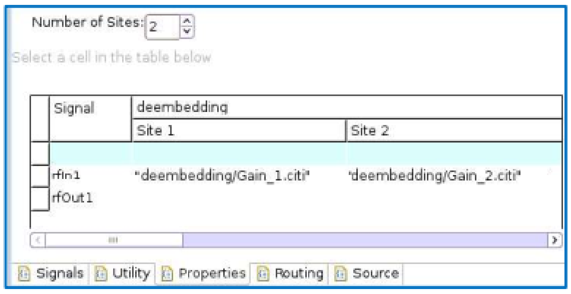
Select a cell in the table below

Signal	disabled	Pogo	
		Site 1	Site 2
rfIn1	false	90101	90109
rfOut1	false	90208	90114



```
dutboard dutboard1 {

    sites = 2;

    signal rfIn1 {
        site 1 { pogo = 90101; }
        site 2 { pogo = 90109; }
    }

    signal rfOut1 {
        site 1 { pogo = 90208; }
        site 2 { pogo = 90114; }
    }
}
```

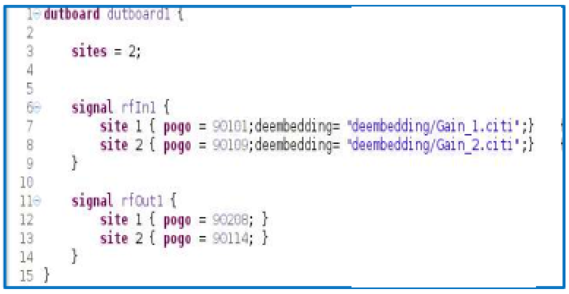
Deembedding is used to compensate for RF DUT board traces.
The optional data is stored in dedicated “*.s2p” files (Touchstone format) or “*.citi” files (Common Instruction Transfer and Interchange files).



Number of Sites: 2

Select a cell in the table below

Signal	deembedding	
	Site 1	Site 2
rfIn1	"deembedding/Gain_1.citi"	"deembedding/Gain_2.citi"
rfOut1		



```
dutboard dutboard1 {

    sites = 2;


    signal rfIn1 {
        site 1 { pogo = 90101;deembedding= "deembedding/Gain_1.citi";}
        site 2 { pogo = 90109;deembedding= "deembedding/Gain_2.citi";}
    }

    signal rfOut1 {
        site 1 { pogo = 90208; }
        site 2 { pogo = 90114; }
    }
}
```

The *property element* allows to define multiple deembedding settings, which can be distinguished by *tags*.
Such a *tag* can be referenced by a *specification file*, for example, when setting up an *action* “modulated” of the “rfSim” *instrument*.

Lab 02: DUT Board Description

Lecture 05: Building Blocks of Test Programs

Lab 03: Test Program

Lecture 06: Testflow and Test Suites

Lab 04: Testflow

Lecture 07: Operating Sequence

Lab 05: Operating Sequence

Lecture 08: Hardware Overview*

Lecture 09: Instruments

Lecture 10: Basics Level and Timing*

Lecture 11: Specification Files - Digital Setups

Lab 06: Timing Sets and Level Sets

Lecture 12: Specification Files - Multi Domains

Lab 07: Timing Debug

Lecture 13: Patterns

Lab 08+09: X-Mode + Pattern Debug

Lecture 14: Actions, Patterns and Transactions

Lab 10: DC Measurement

Lecture 15: Technical Documentation Center

Lecture 16: Testflow File

Lab 11: Test Method Library*

Lecture 17: Test Methods – Introduction

Lab 12: Test Method Creation

Lecture 18: Test Methods – Basics

Lecture 19: Test Methods – Multi Site Types

Lab 13: Test Setup Modifications

Lecture 20: Test Methods – Result Access

Lab 14: Retrieving Test Results

Lecture 21: Datalogging

Lab 15: Datalogging

Lecture 22: Binning

Lecture 23: Usage of Test Tables

Lecture 24: Content of Test Tables

Lecture 25: Test Program Execution and Debug

Lecture 26: Internal Steps of Test Program Execution

Lecture 27: Protocol Aware – Introduction

Lab 16: Protocol Aware Setup Files*

Lecture 28: Test Methods – Programmatic Setup Generation*

Lab 17: Setup Generation in Test Methods*

Lecture 29: Test Methods – Parameters*

Lecture 30: Debug Tools

Lab 18: Debugging Measurement Setups

Lecture 31: Characterization Tools

Lab 19: Characterization/Shmoo*

Lecture 32: Recommended Setup Structure

Lecture 33: Utility Lines

Lecture 34: Licensing

Lecture 35: TCCT

Lecture 36: Test Program Migration Framework*

Lecture 37: Conversion of STIL files*

Open Questions

Feedback

进阶+++

SmarTest 8.2.0-8.3.0 Delta Trainings