ADVANTEST “T2000 GS” IC Tester

VLSI Design & Test Seminar

Victor P. Nelson

2/3/2016

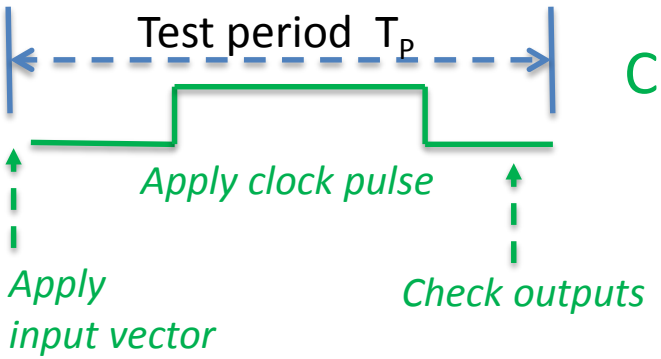
Presentation outline

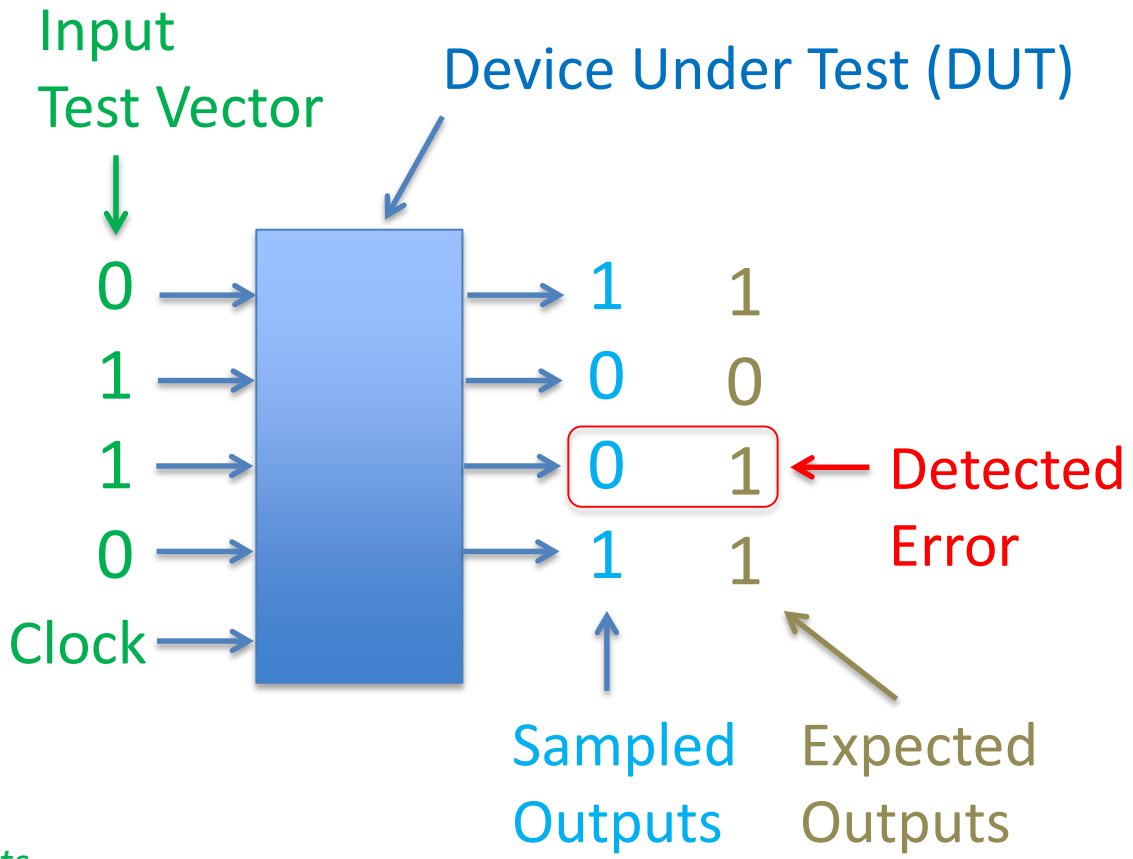
- IC testing process
- Tester architecture
- Device test fixture
- Test plan elements
- Creation of test vectors
- Running tests

IC testing process

For each test vector:

1. Apply test vector to DUT input pins
2. Activate clock
3. Sample DUT outputs
4. Compare sampled to expected outputs





Other IC tests

- Contact test - test chips' package pin opens/shorts
- Input pin DC parametric tests (signal pins)
  - Input leakage current
  - Input threshold voltage
- Output pin DC parametric tests (signal pins)
  - Driving voltage test
- IDD test (power supply pins)
  - Test for VDD current
  - Gross, Static, Dynamic and IDDQ
- AC parametric tests
  - Test quality of output signal and signal timing parameters

ADVANTEST T2000 GS Test System

Operator Station

GS Mainframe:
- system controller
- site controller

Performance board:
- DUT socket pins connect to module channels via HIFIX (High Fidelity Tester Access Fixture)

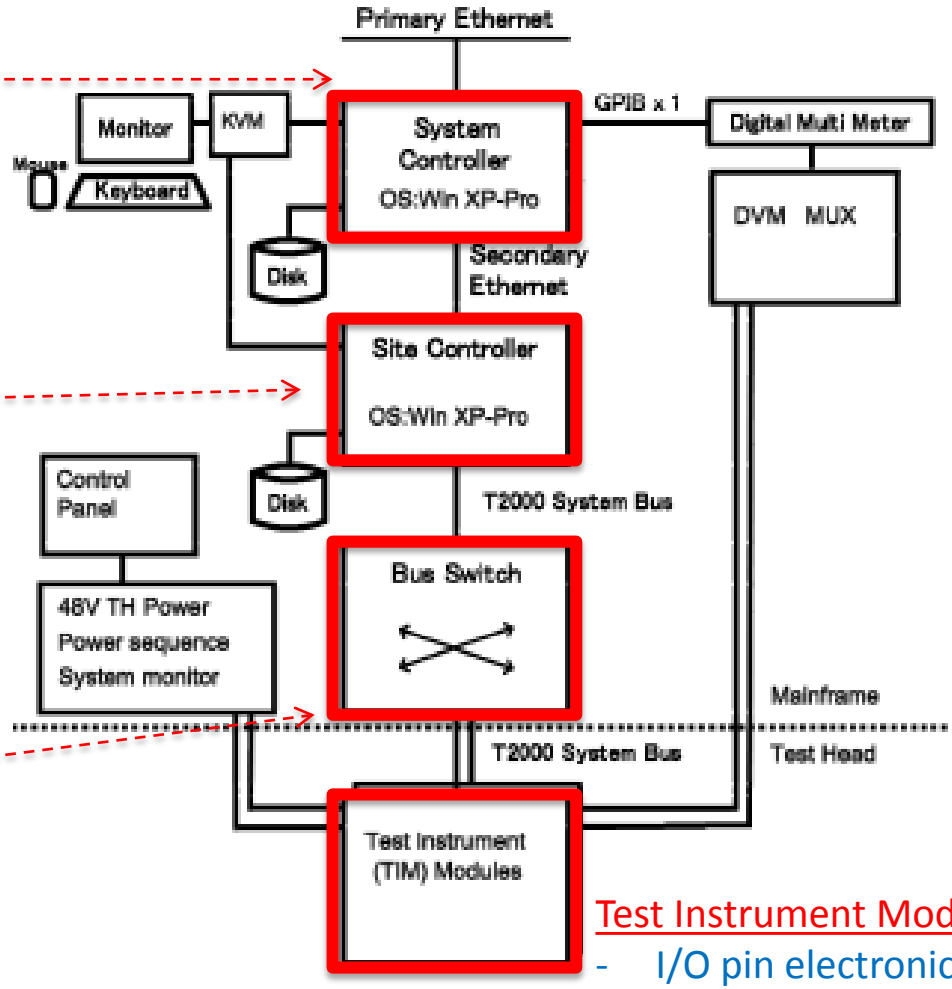
Test Head:
- 13 module slots
  - I/O pin electronics
  - power supplies
- HIFIX
- Optional:
  - handler (volume test)
  - wafer prober
  - manipulator

T2000 GS computing architecture

System controller
- User GUI to the test system to develop and store test plans and patterns
- Sends commands to site controller

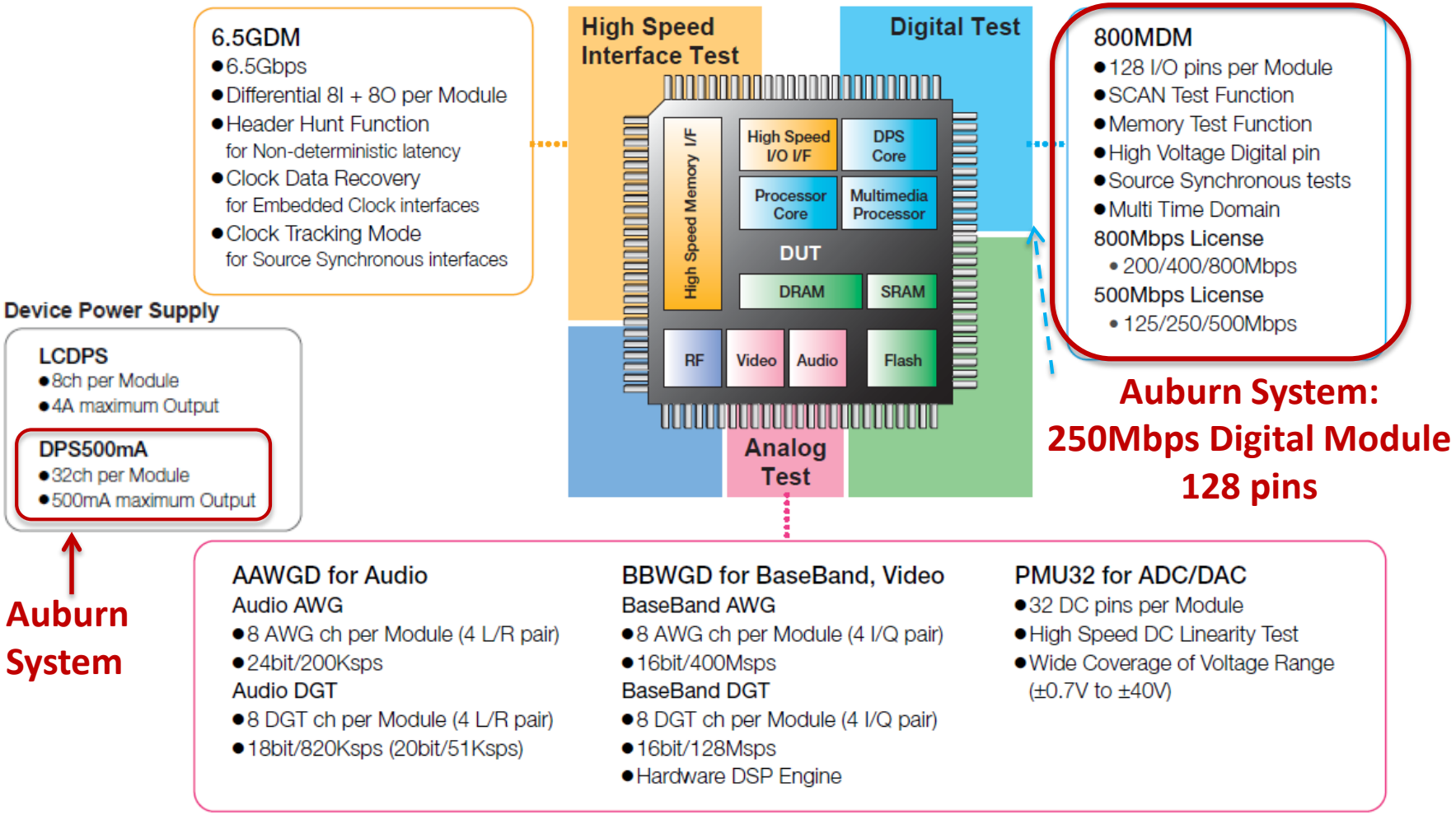
Site Controller
- One per DUT (we have only one)
- Executes test plans on the DUT
  - Controls test instrument modules
  - Returns results to user

Bus Switch
- Configured by a socket file
- Connects site controller to test modules



Test Instrument Modules
- I/O pin electronics
- Power supplies

Test instrument modules
(up to 12 in T2000 GS test head)



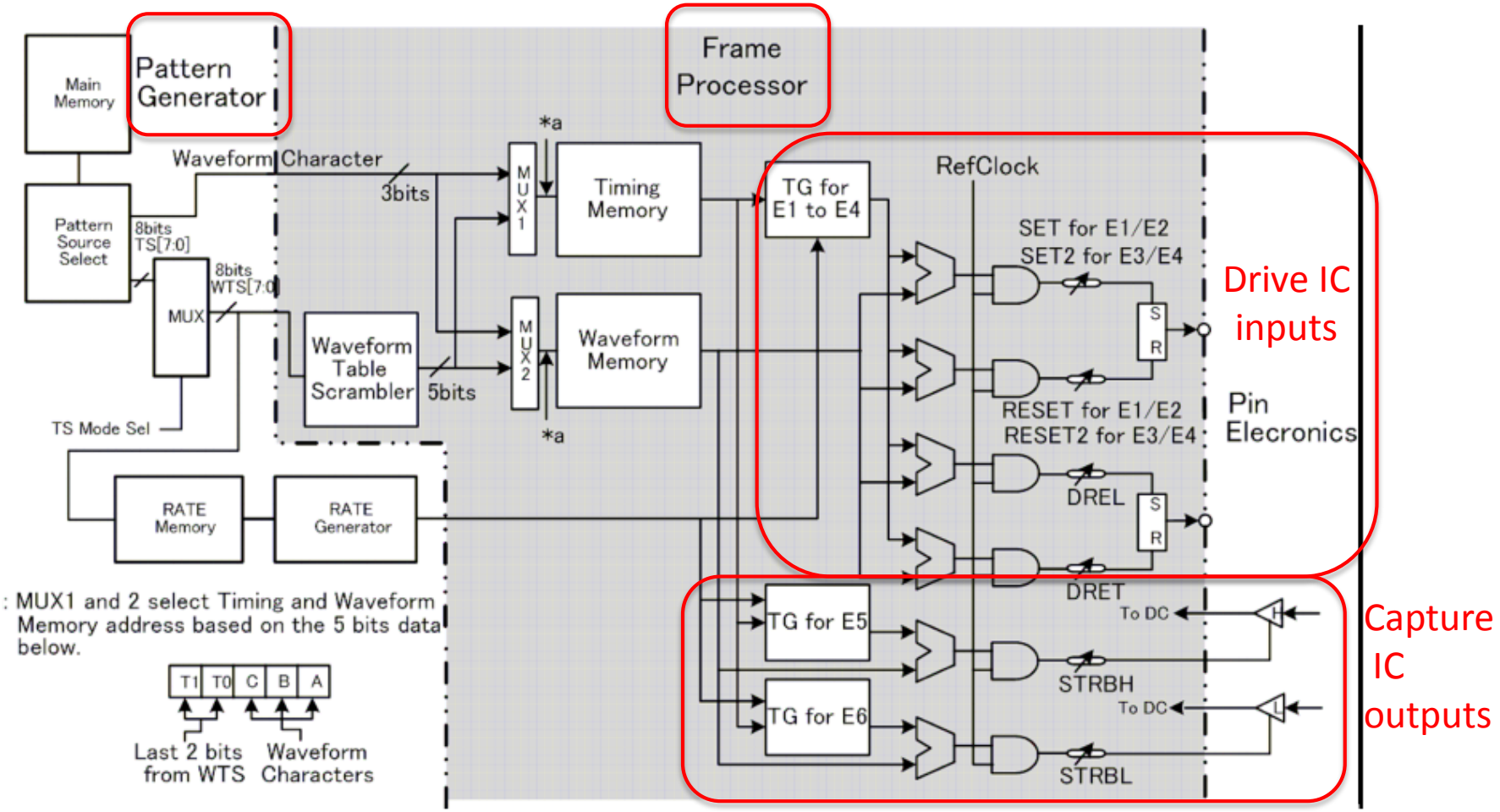
250MDMA pin electronics - specifications

Pin Drivers			250MDMA Specification Value
Voltage range	IC input	VIH	-1.15 ~ +7.0V
		VIL	-1.25 ~ +5.9V
Voltage amplitude		(VIH-VIL)	0.1V ~ 8.0V
Voltage resolution			2mV
Transition time		20% ~ 80%	1.2nS @ 3V
		20% ~ 80%	1nS @ 1V

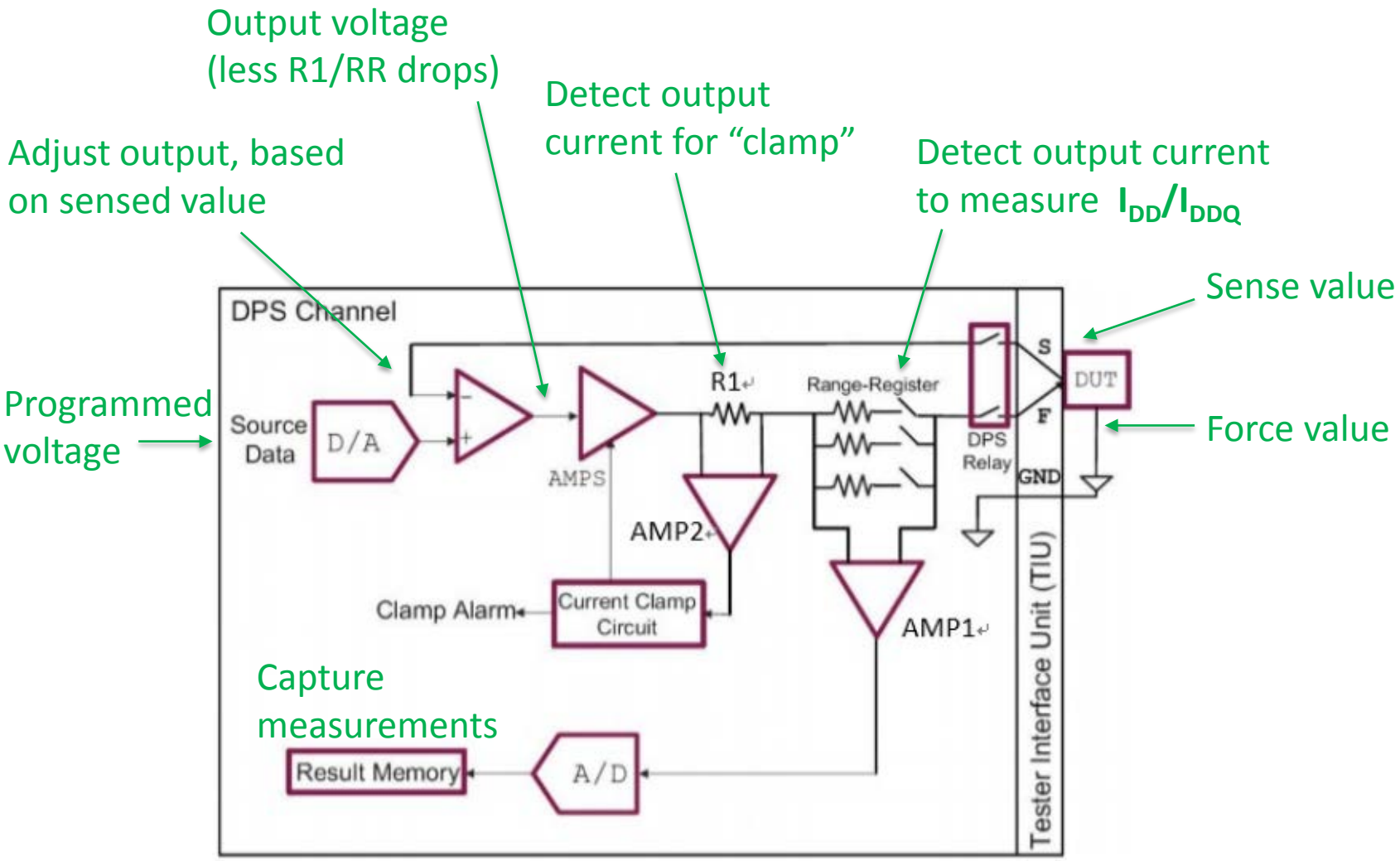
Timing edge resolution: 7.8125ps
Number of timing edges:
6 per pin (4 drive/2 compare)

Pin Comparators			250MDMA Specification Value
Voltage range	IC output	VOH	-1.25 ~ +6.75V
		VOL	-1.25V ~ +6.75V
Minimum voltage (VOH-VOL)			0.0V
Voltage resolution			2mV

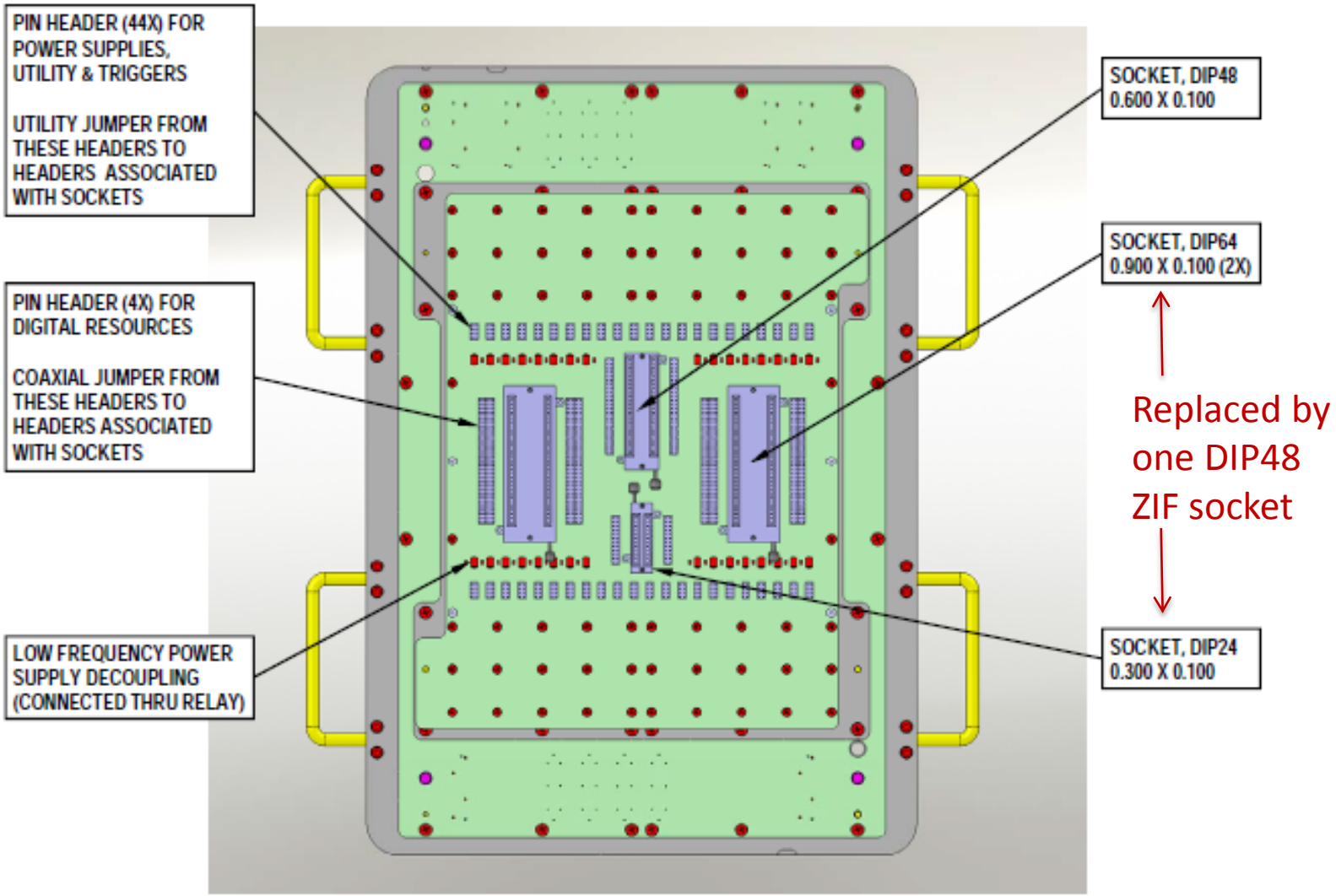
250MDMA pattern generator and frame processor



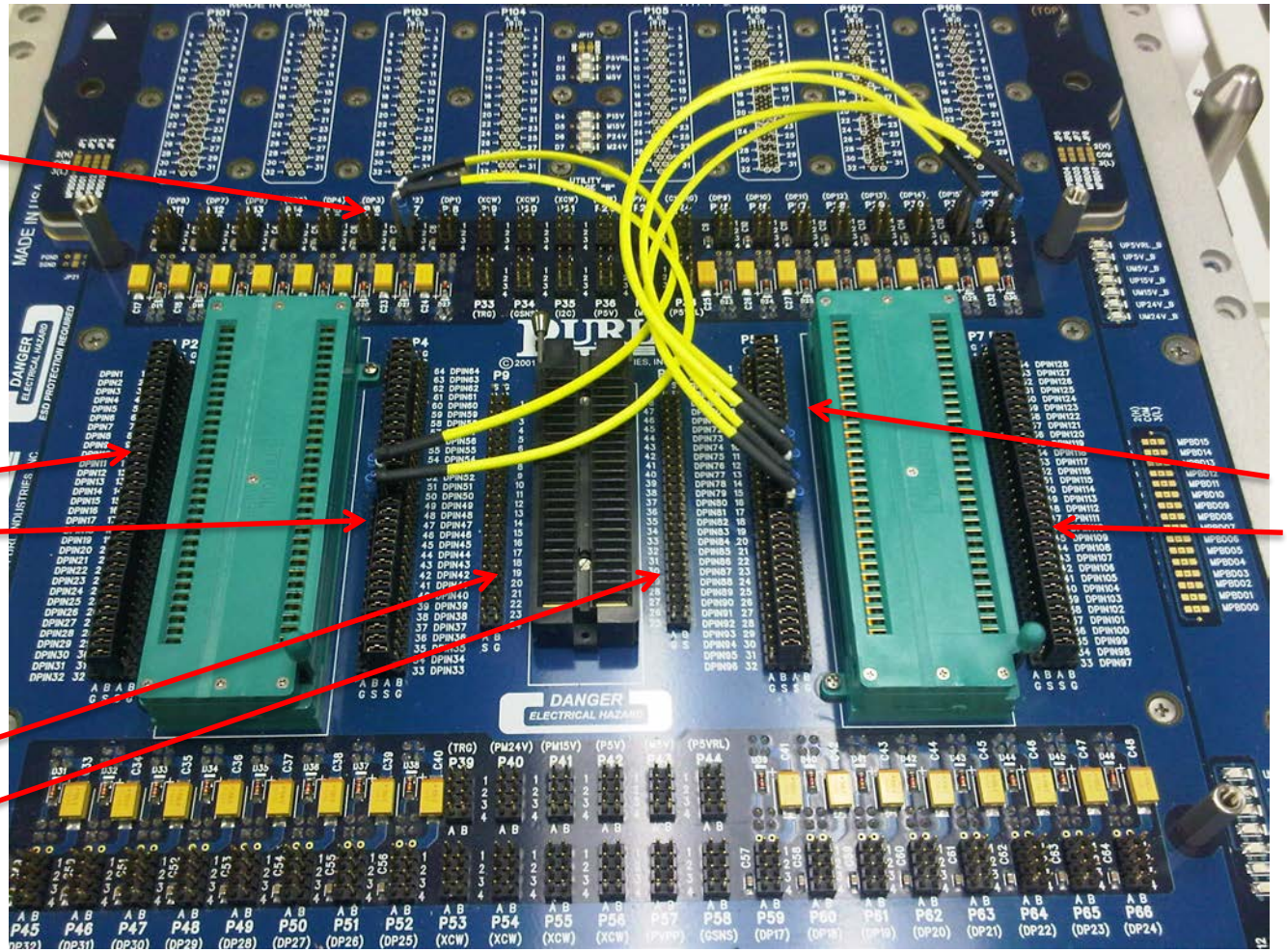
DPS500mA (power supply) pin electronics



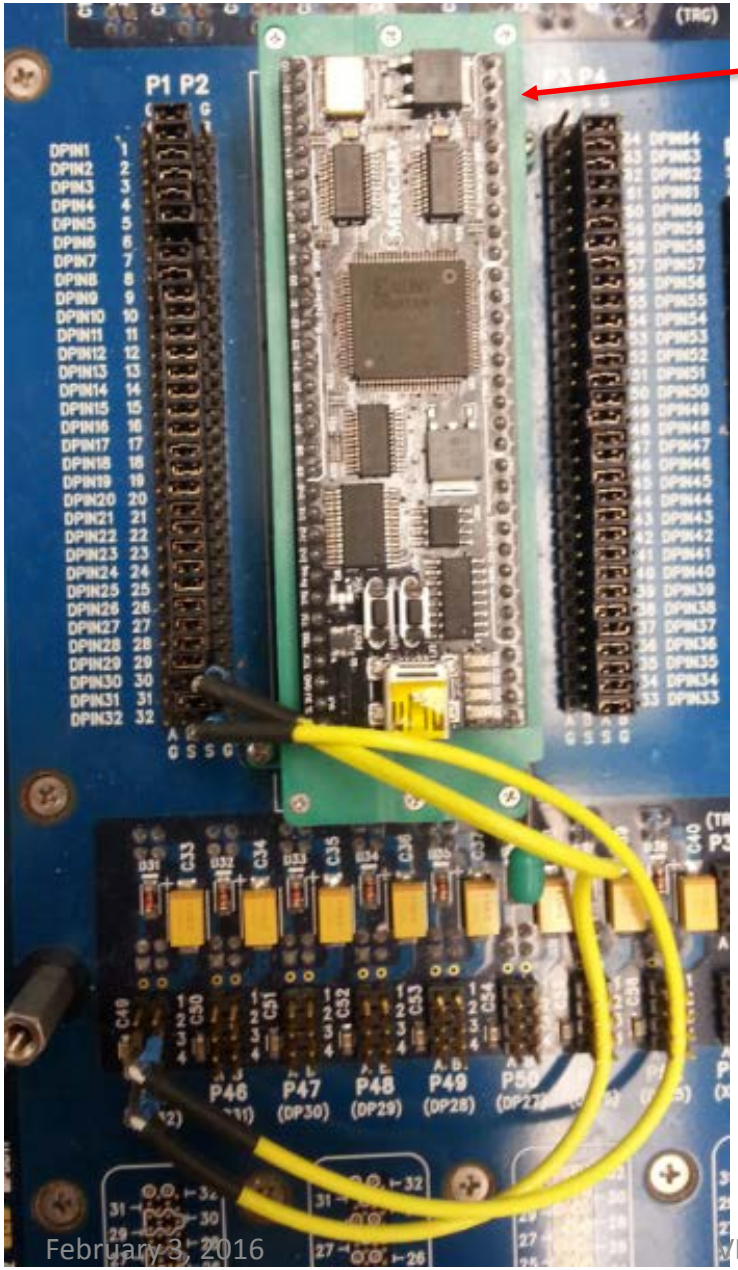
Auburn T2000 performance board
(test fixture for devices to be tested)



Performance board IC sockets

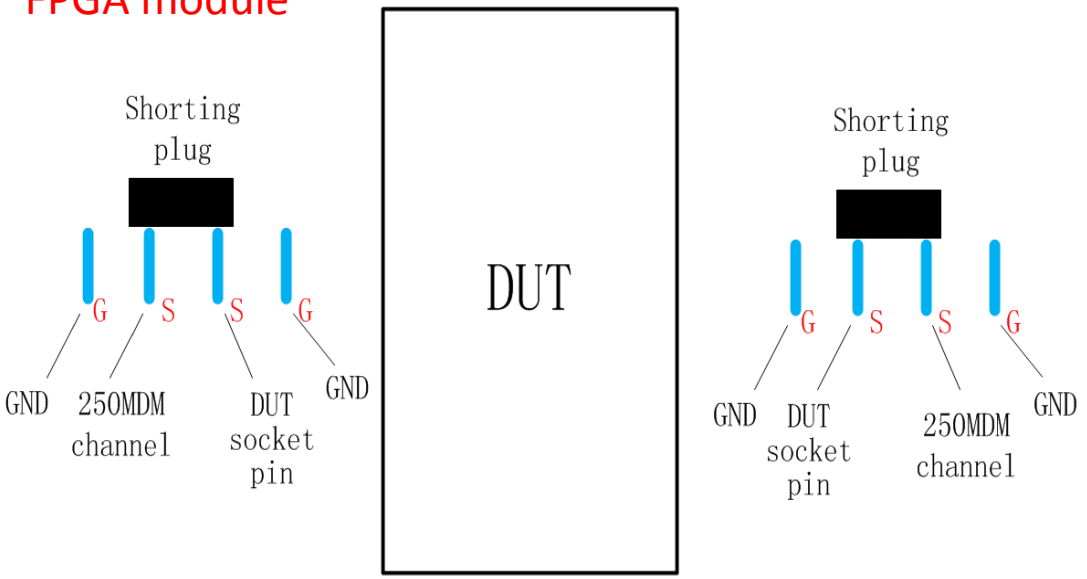


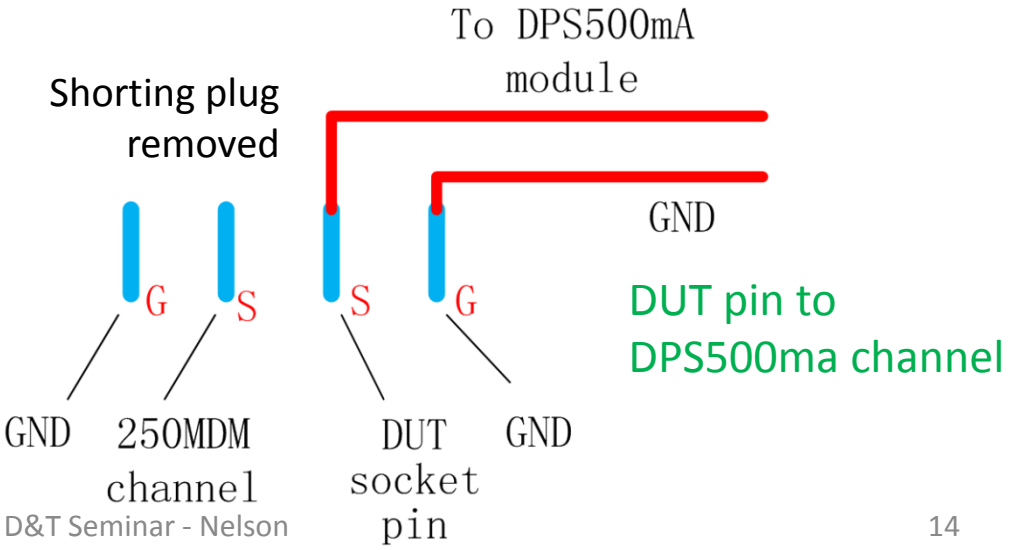
Connecting DUT pins to 250MDMA/DPS500ma



DUT:
FPGA module

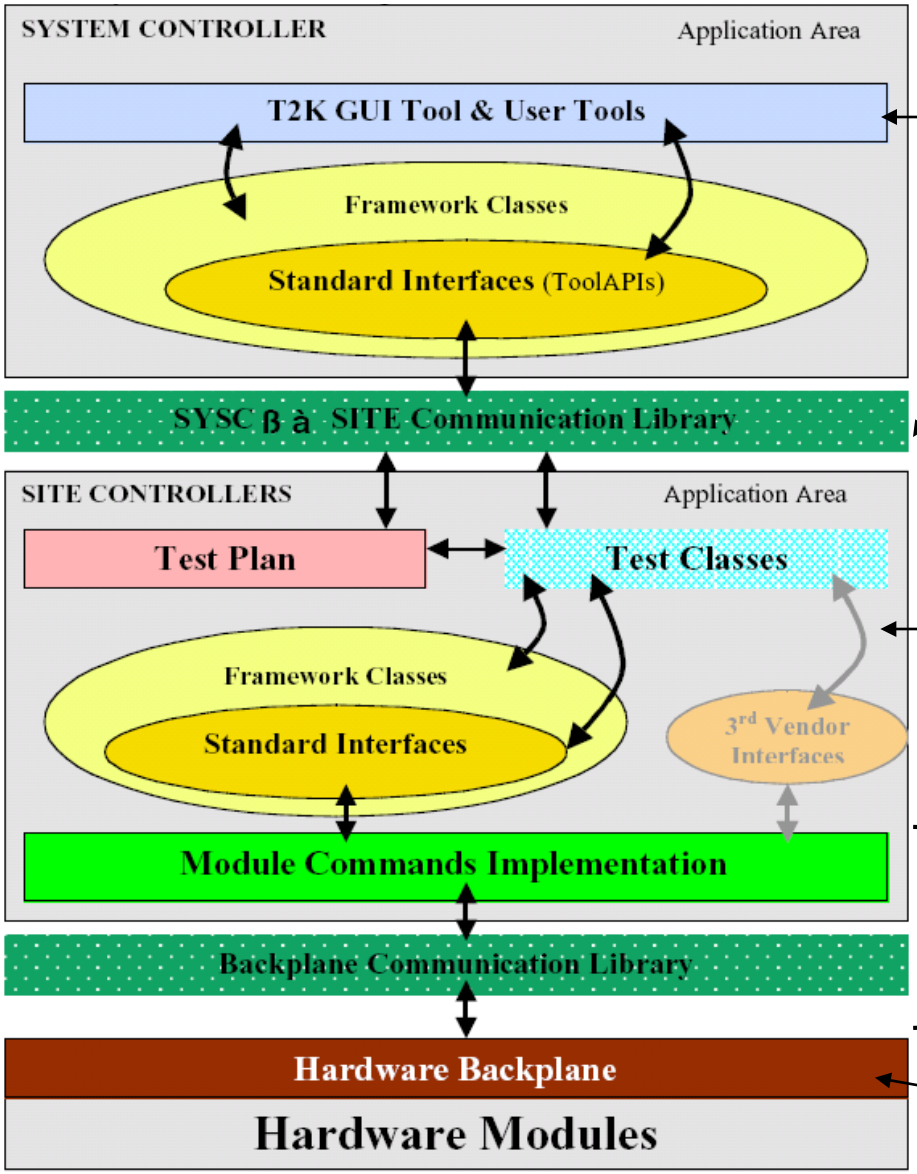
DUT pins to 250MDMA channels





DUT pin to DPS500ma channel

TSS (T2000 System Software) Structure



The primary **User Interface** with the Tester.

Allows communication between GUI and User Tools, Test Plan and Test Classes on the Site Controller(s).

Test Plan resides here along with the Test Classes needed for device test. Interfaces to specific Framework Classes ultimately with Std. Interfaces that translate to module-specific commands.

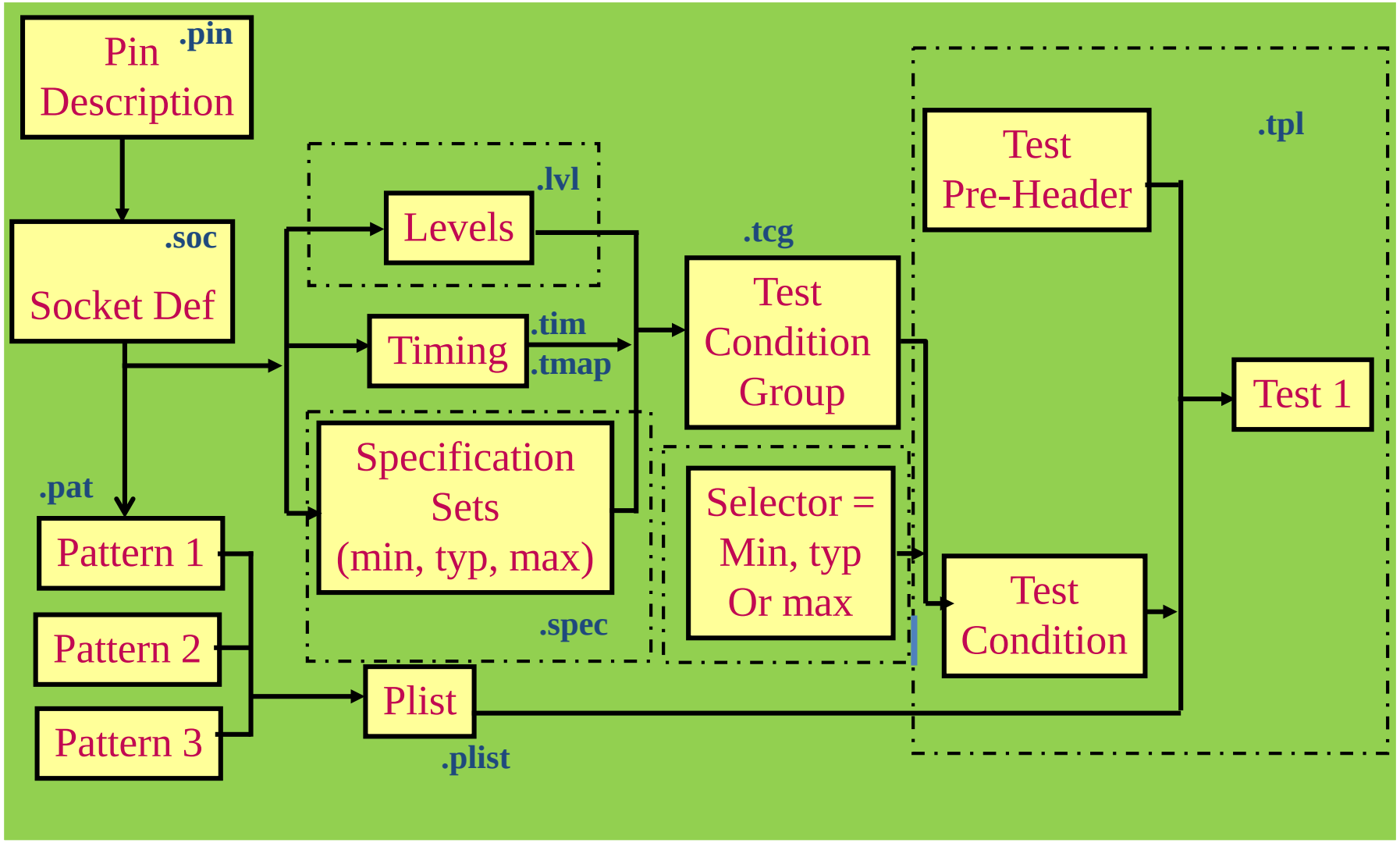
Software layers that control the H/W modules from API's and Functions implemented by the Test Class and Test Plan.

Module Backplane provides **optical/electrical I/F to individual test modules.**

Test Plan

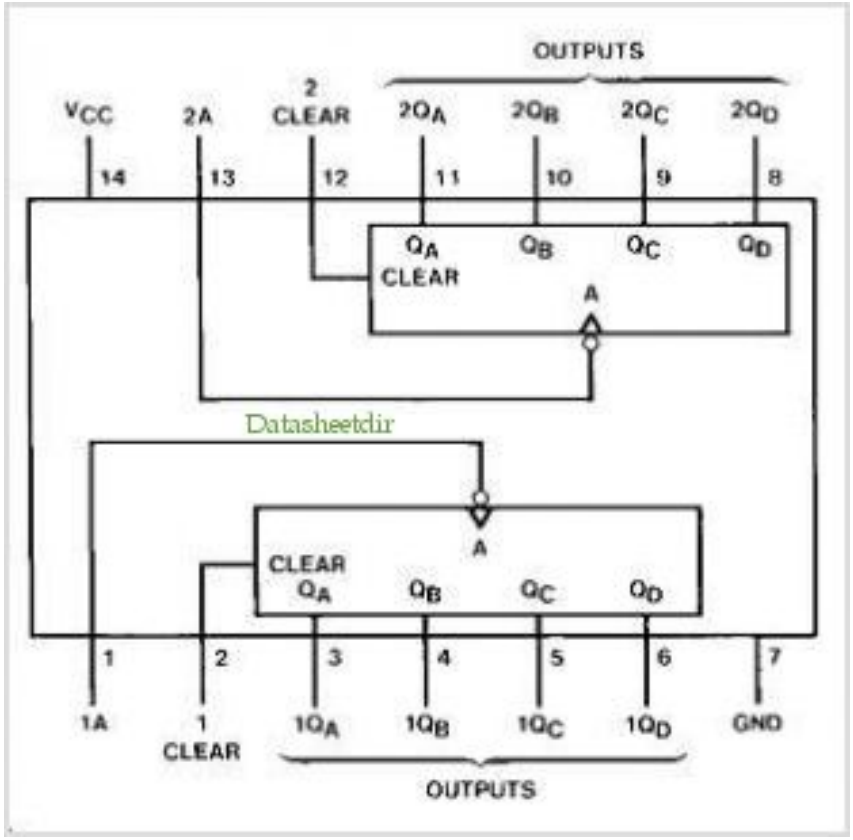
- A *test plan* (*program*) is written by a test engineer.
  - Defines the test flow
  - Executes on the Site Controller
    - Controls the modules to test the device
  - Written in OTPL
    - Open Architecture Test Programming Language
  - Uses framework classes
    - Test, Level, Timing, DCParametrics, user-supplied
  - Configures hardware using standard interfaces
    - provides a way for test plans to interact with common test system hardware components and other test-related objects.

OPTL Test Plan Structure (multiple files)



Example – Test plan for a 74LS393 dual 4-bit binary counter

(14-pin DIP package)



74LS393 “pin description file” (.pin)

DUT pin names and pin groups for timing domains & patterns

```
Version 1.0.0;
PinDescription
{
   Resource AT.Digital.dpin
   {
        A1;
        CLR1;
        QA1;
        QB1;
        QC1;
        QD1;
        A2;
        CLR2;
        QA2;
        QB2;
        QC2;
        QD2;
```

All individual pins

Pins controlled/observed as groups in the test plan

```
Group inpins1
  {
        A1, A2
  }
Group inpins2
  {
        CLR1, CLR2
  }
Group outpins1
  {
        QA1, QB1, QC1, QD1
  }
Group outpins2
  {
        QA2, QB2, QC2, QD2
  }
Domain default
  {
        allpins
  }
```

```
DomainGroup DefaultDG
  {
        default
  }
}

Resource dps500mA
  {
     VDD;
  }

 Resource moduletrigger
  {
        PMDTR0;
        PMDTR1;
        PMDTR2;
        PMDTR3;
  }
}
```

Power supply

(OTPL requires strict formatting)

74LS393 “socket file” (.soc)

Specify DUT pin connections to module channels

```
Version 1.0.0;
SocketDef
{
   DUTType DiagPB
   {
      PinDescription pindesc.pin;
      DUT 1
      {
         SiteController 1;
         Resource AT.Digital.dpin
         {
            A1      1003.1;
            CLR1    1003.2;
            QA1     1003.3;
            QB1     1003.4;
            QC1     1003.5;
            QD1     1003.6;
            QD2     1003.58;
            QC2     1003.59;
            QB2     1003.60;
            QA2     1003.61;
            CLR2    1003.62;
            A2      1003.63;
         }
```

250MDMA connectors:
1003.1 .. 64
2003.1 .. 64

connector.channel

```
         Resource dps500mA
         {
            VDD   1010.2;
         }
```

DPS500ma connector:
1010.1 .. 32

```
         Resource moduletrigger
         {
            PMDTR0 1003.129;
            PMDTR1 1003.130;
            PMDTR2 2003.131;
            PMDTR3 2003.132;
         }
      }
   }
}
```

Connector 1003 -> left 64-pin ZIF socket & 48-pin ZIF socket
Connector 2003 -> right 64-pin ZIF socket

74LS393 device “specification file” (.spec)

Device voltage/current specifications (from its data sheet)

```
Version 1.0;
Import uservar.usrv;
SpecificationSet functional_Specs(min, typ, max)
{
        Voltage vforce =    4.75V,          5V,      5.25V;
        Current ich    =    20mA,     100mA,     200mA;
        Current icl    = -400mA, -1600mA, -2400mA;
        VoltageSlew slewrate = 78.125;
        Voltage vih = 5V;
        Voltage vil  = 0V;
        Voltage voh =   2.5V,   3.4V, 3.4V;
        Voltage vol  = 0.35V, 0.35V, 0.5V;
}
```

A “test condition” will select min, typ, or max values

Tester output levels applied to DUT inputs

Thresholds on tester inputs from DUT outputs

Levels file (.lvl)

Voltages/currents for DUT signal pins, Force voltages for DUT power supply pins.

```
Version 1.0;
Import pindesc.pin;
# pindesc.pin declares names:
#       VDD, inpins, outpins
# resource.rsc declares names:
#       VSRange, VForce, Relay, VIH, etc.
Levels Lvl1
{
        VDD
        {
                VSRange = 7V;
                VForce = vforce;
                DpsRelay = CLOSE;
                PowerSequence = ON;
        }
        Delay 3mS;

        inpins
        {
                VIH = vih;
                VIL = vil;
                PinOutRelay = CLOSE;
                PowerSequence = ON;
        }

        outpins
        {
                VOH = voh;
                VOL = vol;
                PinOutRelay = CLOSE;
                PowerSequence = ON;
        }
}
```

Power on sequence

Variable defined in spec file

Delay before using I/O pins

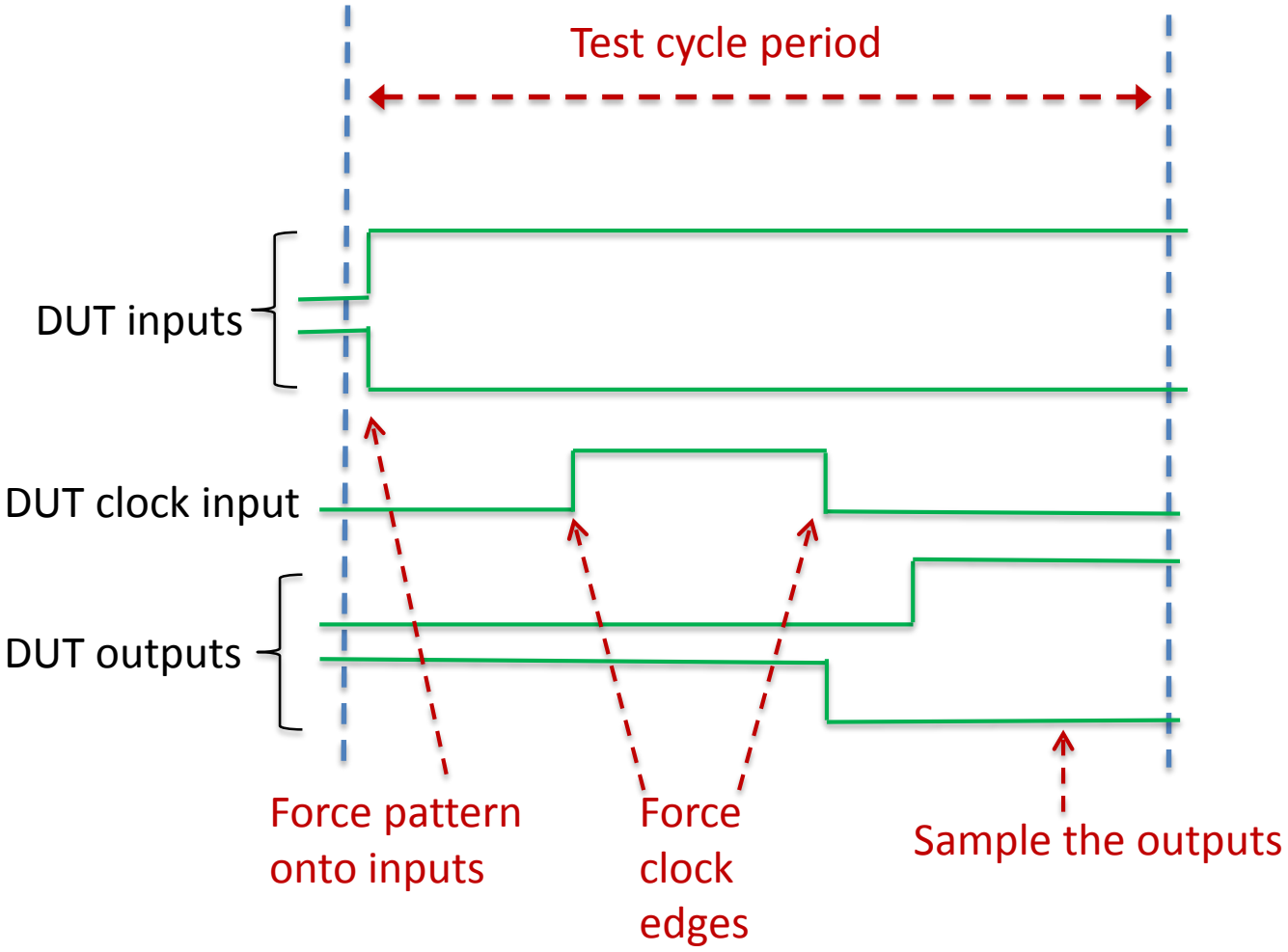
System "resource"

Variable defined in spec file

Set up DUT pins

Test pattern timing – for each test vector

May define different timing patterns for different pins and/or test steps.
(4 force edges + 2 compare edges per pin)



Timing file (.tim)

Define timing of input transitions and sample times

```
PeriodTable
      {
       #Cycle time "rate0" for test freq = 5MHz
       Period rate0 { 200nS; }
      }
#Force times for device inputs
Pin INPCONTROL_PINS
      {
       WaveformTable inpctrl
            {
              { 1 { U@0nS; } }
              { 0 { D@0nS; } }
            }
      }
```

Test period

Test pattern Symbols

Up/Down

Transition TIme

```
#Sample times for device outputs
Pin OUTPINS
      {
       WaveformTable out
            {
              { H { H@85nS,E5; } }
              { L { L@85nS,E6; }
              { X { Z@0nS; } }
            }
}
```

Timing Edge

Test pattern Symbols

Sample High/Low

Sample TIme

Timing file example

This test engineer wanted to repeat tests for different periods to see when chips begin to fail

```
Version 1.0;
Import pindesc.pin;
# Perform the test with one set of parameters
Timing Tim_300_to_290
{
    CommonSection
    {
        Domain default
        {
            PeriodTable
            {
               Period per0 { 300nS; }
               Period per1 { 297.5nS; }
               Period per2 { 295nS; }
               Period per3 { 292.5nS; }
            }
            Pin inpins
            {
                WaveformTable seq1
                {
                    { 1 { U@0nS,E1; } }
                    { 0 { D@0nS,E1; } }
                }
            }
            Pin outpins
              {
                WaveformTable seq1
                    {
                     { H { H@299.5nS,E5; } }
                     { L { L@299.5nS,E6; } }
                  }
                 WaveformTable seq2
                  {
                      { H { H@297nS,E5; } }
                       { L { L@297nS,E6; } }
                  }
                 WaveformTable seq3
                  {
                       { H { H@294.5nS,E5; } }
                       { L { L@294.5nS,E6; } }
                  }
                  WaveformTable seq4
                  {
                        { H { H@292nS,E5; } }
                        { L { L@292nS,E6; } }
                  }
                }
            }
        }
    }
}
```

Define 4 periods

Apply all inputs at start of period

4 different output sample times - one for each period

Timing map file (.tmap)

Combine individual pin & rate timings into DUT "timing sets"

```
Version 1.0;
Import pindesc.pin;
TimingMap TMap1
{
    Domain default
    {
        WaveformMap
        {
            PinFormat { inpins, outpins }
            wfs1, per0, { seq1, seq1 }
            wfs2, per1, { seq1, seq2 }
            wfs3, per2, { seq1, seq3 }
            wfs4, per3, { seq1, seq4 }
        }
    }
}
```

From .pin description file

Test to be performed 4 times, using different timing, i.e. 4 different "waveform sets"

Waveform Set

Period value

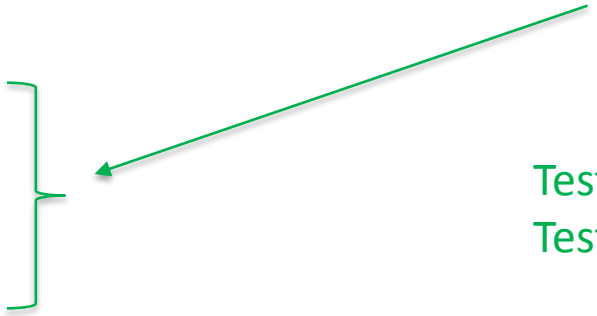
Input waveform table

Output waveform table

From .tim file

Test condition group file (.tcg)

TCG associates: spec, level, timing & timing map files

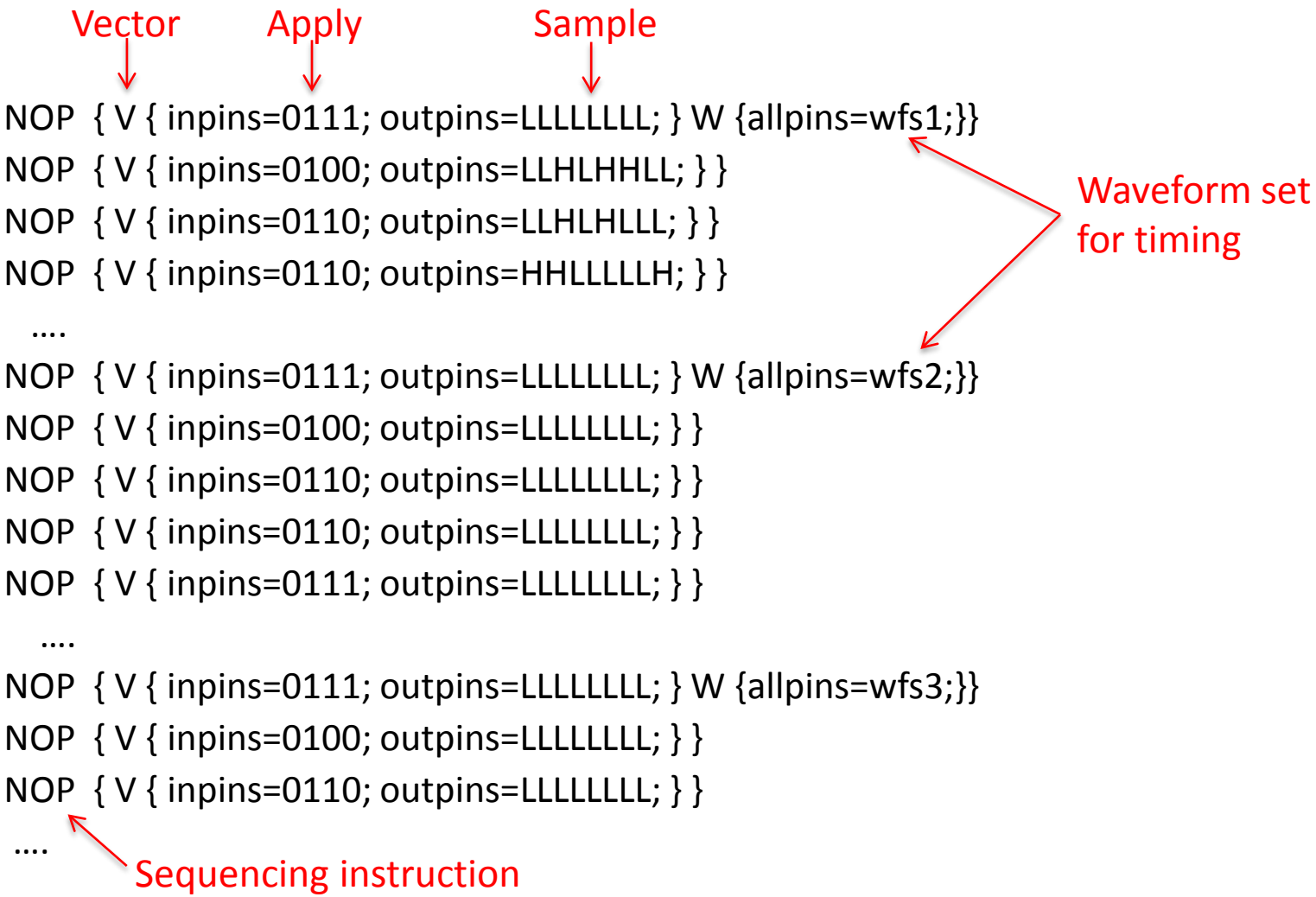


```
Version 1.0;
Import timing.tim;
Import timingmap.tmap;
Import level.lvl;
Import DiagPBSpec.spec;

# A Levels-Only Test Condition Group.
TestConditionGroup DiagPBTCG_300_to_290
{
        SpecificationSet DiagPBSpec;      #from .spec file
        Levels Lvl1;                      #from .lvl file
        Calibration CalBlock1;            #from .tim file
        Timings
        {
                Timing = Tim_300_to_290;  #from .tim file
                TimingMap = TMap1;        #from .tmap file
        }
}
```

Test conditions for one
Test or set of tests

Pattern (vector) files

Pin order is defined in .pin file



Functional test vectors may be created from simulation results

Time(ns)	^A	^B	^Function	^S	^Clk1	^Clk2	^F	^Fb	^Other1	^Other2
0.0	0	0	2	B	0	0	Xr	Xr	Xr	Xr
1.0	0	0	2	B	1	0	X	X	X	X
2.0	0	0	2	B	0	0	X	X	X	X
18.0	0	0	2	B	0	1	X	X	X	X
19.0	0	0	2	B	0	0	X	X	X	X
21.0	0	0	2	B	0	0	0	0	1	1
23.0	0	1	2	B	0	0	0	0	1	1
24.0	0	1	2	B	1	0	0	0	1	1
25.0	0	1	2	B	0	0	0	0	1	1
41.0	0	1	2	B	0	1	0	0	1	1
42.0	0	1	2	B	0	0	0	0	1	1
46.0	0	2	2	B	0	0	0	0	1	1
47.0	0	2	2	B	1	0	0	0	1	1
48.0	0	2	2	B	0	0	0	0	1	1
64.0	0	2	2	B	0	1	0	0	1	1
65.0	0	2	2	B	0	0	0	0	1	1
69.0	0	3	2	B	0	0	0	0	1	1
70.0	0	3	2	B	1	0	0	0	1	1
71.0	0	3	2	B	0	0	0	0	1	1
87.0	0	3	2	B	0	1	0	0	1	1
88.0	0	3	2	B	0	0	0	0	1	1
92.0	0	4	2	B	0	0	0	0	1	1
93.0	0	4	2	B	1	0	0	0	1	1
94.0	0	4	2	B	0	0	0	0	1	1
110.0	0	4	2	B	0	1	0	0	1	1

One “test cycle”:

Inputs applied at 23

Clk1 applied from 24-25

Clk2 applied from 41-42

Outputs stable after 42

Partition into “test periods”.

Within each period define:

- Times inputs applied
- Times of clock edges
- Times outputs sampled

Vectors extracted from functional simulation

(to be translated to T2000 pattern format)

Each vector: →

Inputs (A,B,Fct) to be applied at start of cycle

Clocks (Ck1,Ck2) to be pulsed during cycle

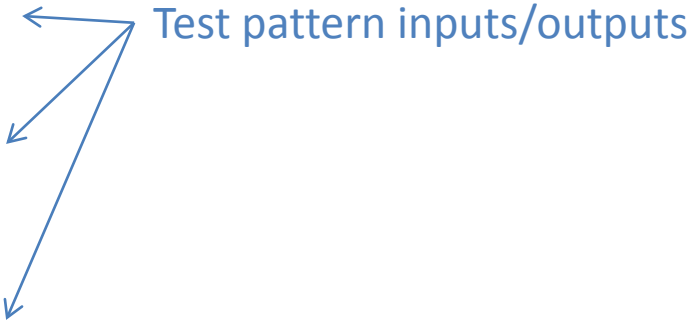
Outputs (S,F,Fb) to be sampled at end of cycle

A	B	Fct	S	Ck1	Ck2	F	Fb	O1	O2
0	0	2	B	1	1	0	0	1	1
0	1	2	B	1	1	0	0	1	1
0	2	2	B	1	1	0	0	1	1
0	3	2	B	1	1	0	0	1	1
0	4	2	B	1	1	0	0	1	1
0	5	2	B	1	1	0	0	1	1
0	6	2	B	1	1	0	0	1	1
0	7	2	B	1	1	0	0	1	1
0	8	2	B	1	1	0	0	1	1
0	9	2	B	1	1	0	0	1	1
0	A	2	B	1	1	0	0	1	1
0	B	2	B	1	1	0	0	1	1
0	C	2	B	1	1	0	0	1	1
0	D	2	B	1	1	0	0	1	1
0	E	2	B	1	1	0	0	1	1
0	F	2	B	1	1	0	0	1	1
1	0	2	B	1	1	0	0	1	1
1	1	2	B	1	1	1	1	3	3
1	2	2	B	1	1	0	0	1	1

Fastscan ATPG tool - ASCII test file

(convert to T2000 test patterns)

```
SETUP =
    TEST_CYCLE_WIDTH = 1;

    DECLARE INPUT BUS "ibus" = "/A0", "/A1", "/A2", "/A3",
                       "/B0", "/B1", "/B2", "/B3",
                       "/M", "/S3", "/S2", "/S1",
                       "/S0", "/C'n";
    DECLARE OUTPUT BUS "obus_1" = "/A=B", "/C'n+4", "/F0", "/F1",
                       "/F2", "/F3", "/X", "/Y";
END;

CYCLE_TEST =

  PATTERN = 0;
    CYCLE = 0;
    FORCE "ibus" "100011000011101" 0;
    MEASURE "obus_1" "01101100" 1;

  PATTERN = 1;
    CYCLE = 0;
    FORCE "ibus" "000110000011010" 0;
    MEASURE "obus_1" "00010011" 1;

  PATTERN = 2;
    CYCLE = 0;
    FORCE "ibus" "001100000110100" 0;
    MEASURE "obus_1" "00000001" 1;
```



Ex: FPGA boundary scan register test patterns

```
NOP{V{TCK=1;TMS=0;TDI=1;TDO=X;}}  #first bit shifted in at TDI
NOP{V{TCK=1;TMS=0;TDI=0;TDO=X;}}
NOP{V{TCK=1;TMS=0;TDI=1;TDO=X;}}
NOP{V{TCK=1;TMS=0;TDI=0;TDO=X;}}
NOP{V{TCK=1;TMS=0;TDI=1;TDO=X;}}
NOP{V{TCK=1;TMS=0;TDI=1;TDO=X;}}
NOP{V{TCK=1;TMS=0;TDI=0;TDO=X;}}
NOP{V{TCK=1;TMS=0;TDI=0;TDO=X;}}
```

Shift pattern 10101100
into BSR via the TDI pin

```
IDXI 363 {V{TCK=1;TMS=0;TDI=X;TDO=X;}}
```

Repeat 363 times to shift pattern
through all cells of the FPGA's BSR

```
NOP{V{TCK=1;TMS=0;TDI=X;TDO=H;}} #first bit shifted out at TDO
NOP{V{TCK=1;TMS=0;TDI=X;TDO=L;}}
NOP{V{TCK=1;TMS=0;TDI=X;TDO=H;}}
NOP{V{TCK=1;TMS=0;TDI=X;TDO=L;}}
NOP{V{TCK=1;TMS=0;TDI=X;TDO=H;}}
NOP{V{TCK=1;TMS=0;TDI=X;TDO=H;}}
NOP{V{TCK=1;TMS=0;TDI=X;TDO=L;}}
NOP{V{TCK=1;TMS=0;TDI=X;TDO=L;}}
```

Verify pattern 10101100
shifted out of BSR via the TDO pin

Test plan (.tpl)

Specify test conditions and test flow

(Standard “header section” of the test plan file omitted)

```
# Declarations of TestConditions TC1Min, TC1Typ, TC1Max, TC2Min, TC2Typ, etc
TestCondition TC_300_to_290
{
        TestConditionGroup = DiagPBTCG_300_to_290;
        Selector = typ;
}
# Other TestConditions

# Declare a "FunctionalTest“, which refers to a C++ test class that runs the test
# and returns a 0, 1 or 2 as a result.
Test FunctionalTest DiagPBFunctionalTest_300_to_290
{
        PListParam = DiagPBPat;
        TestConditionParam = TC_300_to_290;
}
# Other functional tests
```

continued

Test plan (continued)

```
# FlowMain is the main flow.
DUTFlow FlowMain
{ # First flow to be executed:
        DUTFlowItem DatalogSetupFlow DatalogSetup
        {
                Result 0  {
                        Property PassFail = "Pass";
                        GoTo FlowMain_300_to_290;
                }
        }
        DUTFlowItem FlowMain_300_to_290 DiagPBFunctionalTest_300_to_290
        {
                Result 0  {
                        Property PassFail = "Pass";
                        IncrementCounters PassCount;
                        GoTo FlowMain_290_to_280;
                }
                Result 1        {
                        Property PassFail = "Fail";
                        IncrementCounters FailCount;
                        SetBin SoftBins.FailCache3GHz;
                        Return 1;
                }
        }
```

StepName & FunctionalTest

Result 0 = “Pass”
Result 1 = “Fail”

Continue with this flow item if test passes

Quit if test fails

continued

Test plan example – FPGA

(1) power up, (2) configure FPGA, (3) test the circuit

```
Test FunctionalTest Functional_power_typ
        {   ## Test Description = "Functional Test for typ values";
            PListParam = powerup;
            TestConditionParam = TC_typpower;
            DebugMode = 0;
      }
Test FunctionalTest Functional_dpins_typ
     {    ## Test Description = "Functional Test for DPINS typ for FPGA configuration";
         PListParam = fpgaconfigpat;
         TestConditionParam = TC_typdpins;
         DebugMode = 0;
         }
Test FunctionalTest Funct_test
         { ## Test Description = "Functional Test post configuration";
          PListParam = testpat;
          TestConditionParam = TC_typtest;
          DebugMode = 0;
          }
```

Power up the FPGA

Download bit file to the FPGA

Test the configured FPGA circuit

FPGA Test Plan (continued)

```
DUTFlowItem FlowMain_Func_power_typ Functional_power_typ
{
        Result 0  {
            Property PassFail = "Pass";
            GoTo FlowMain_Func_dpins_typ;
        }
        Result 1 {
              Return 1;
        }
    }
DUTFlowItem FlowMain_Func_dpins_typ Functional_dpins_typ
    {
        Result 0 {
            Property PassFail = "Pass";
            GoTo Flowmain_functional_test;
        }
        Result 1 {
            Return 1;
        }
    }
DUTFlowItem Flowmain_functional_test Funct_test
{       Result 0  {
                Return 0;
        }
        Result 1        {
                Return 1;
         }
}
```

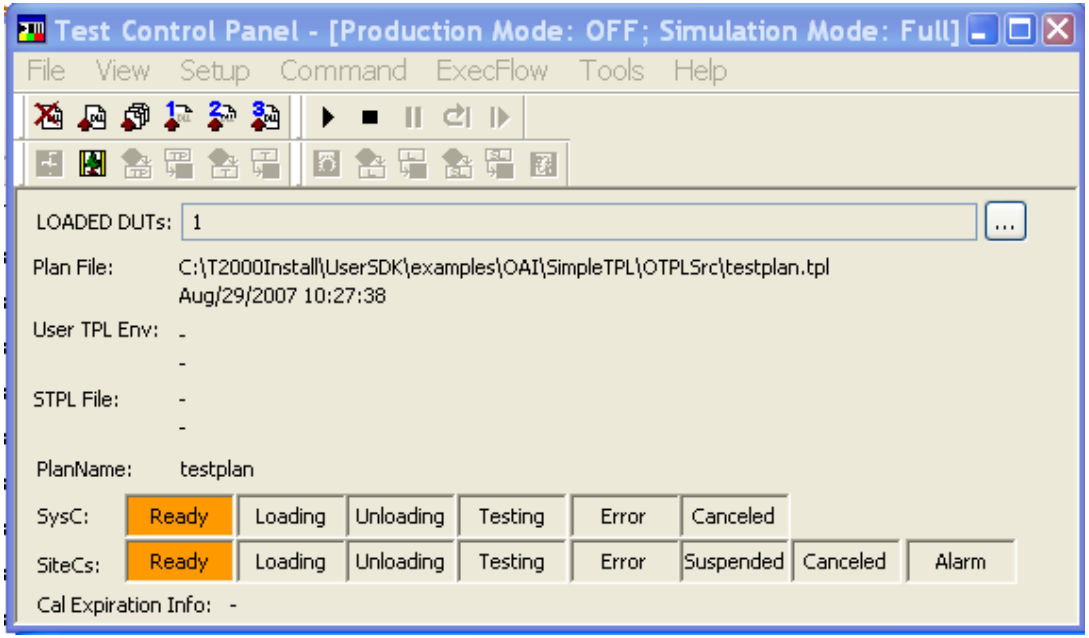
Power up the FPGA

Download bit file to the FPGA

Test the configured circuit in the FPGA

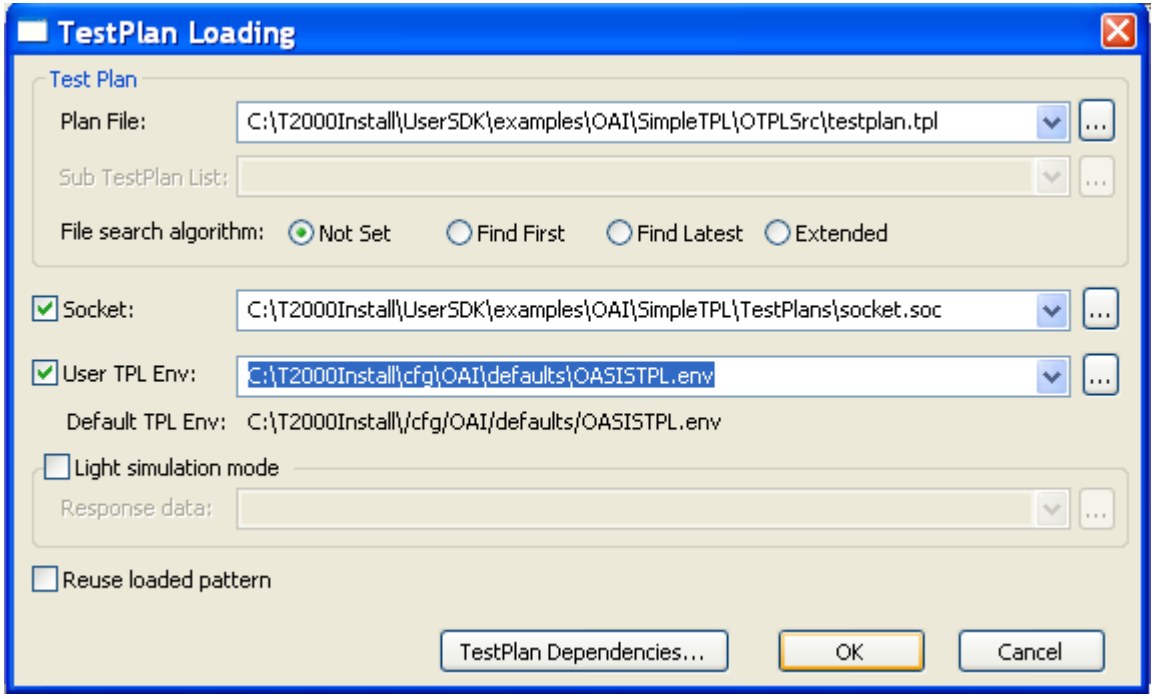
Running T2000 ATE

- Open DOS command prompt
- Go to your working directory
- Go to Pattern directory
- Type t2kctrl start



Loading test plan

From Test Control Panel, select File->Load TestPlan

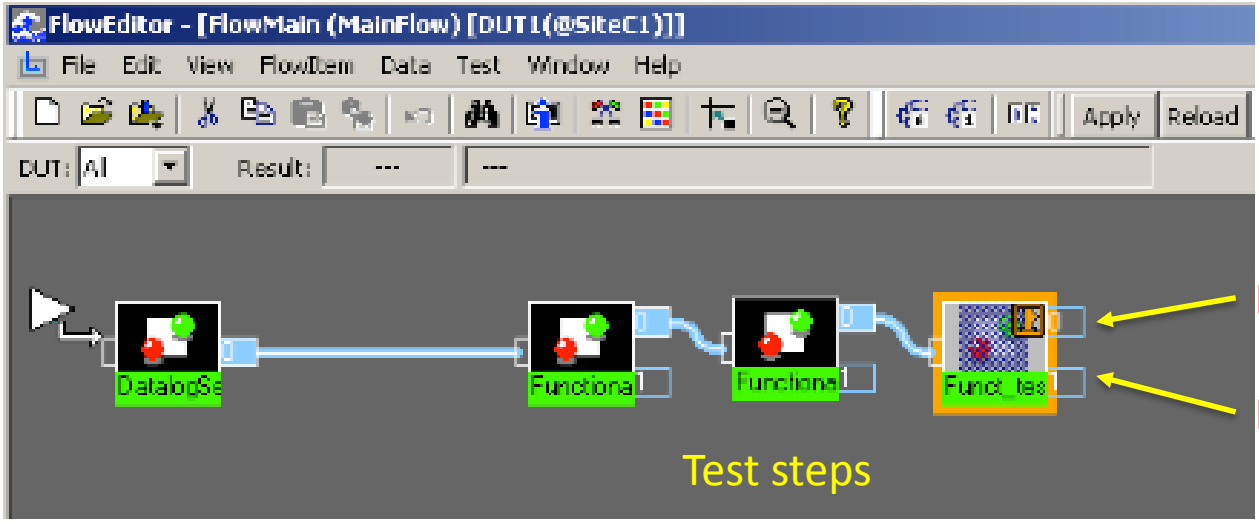


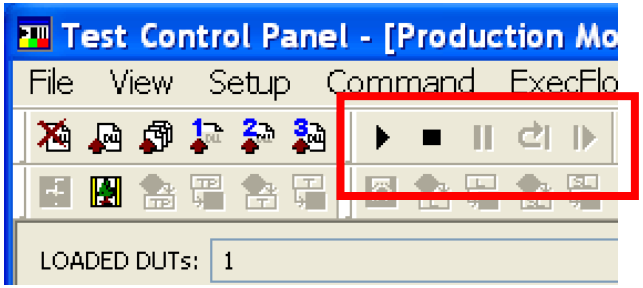
In the "Plan file" load the .tpl file from the OTPLSrc folder
Select "User TPL Env" and choose your .env file from your work directory

Flow Editor

Control and/or edit the main test flow

From Test Control Panel, select Tools -> flow editor





Control the test:

Start, Stop, Suspend, Reset, Continue

Other test options

- DC Parametric Tests
  - Per-pin parametric measurement unit
  - IDD tests
- Pattern editor – modify the test patterns
- Oscilloscope tool – study signal waveforms
- SHMOO plots
  - Modify variables over a range and plot #pass/fail vec's
- Test of scan-based designs
- Complex timing (ex. double data rate)
- Binning (hard and soft)
  - Control handler to move failed parts to bins