# Functional IC test with the ADVANTEST T2000 GS system

VLSI Design & Test Seminar

Victor P. Nelson

1/15/2014

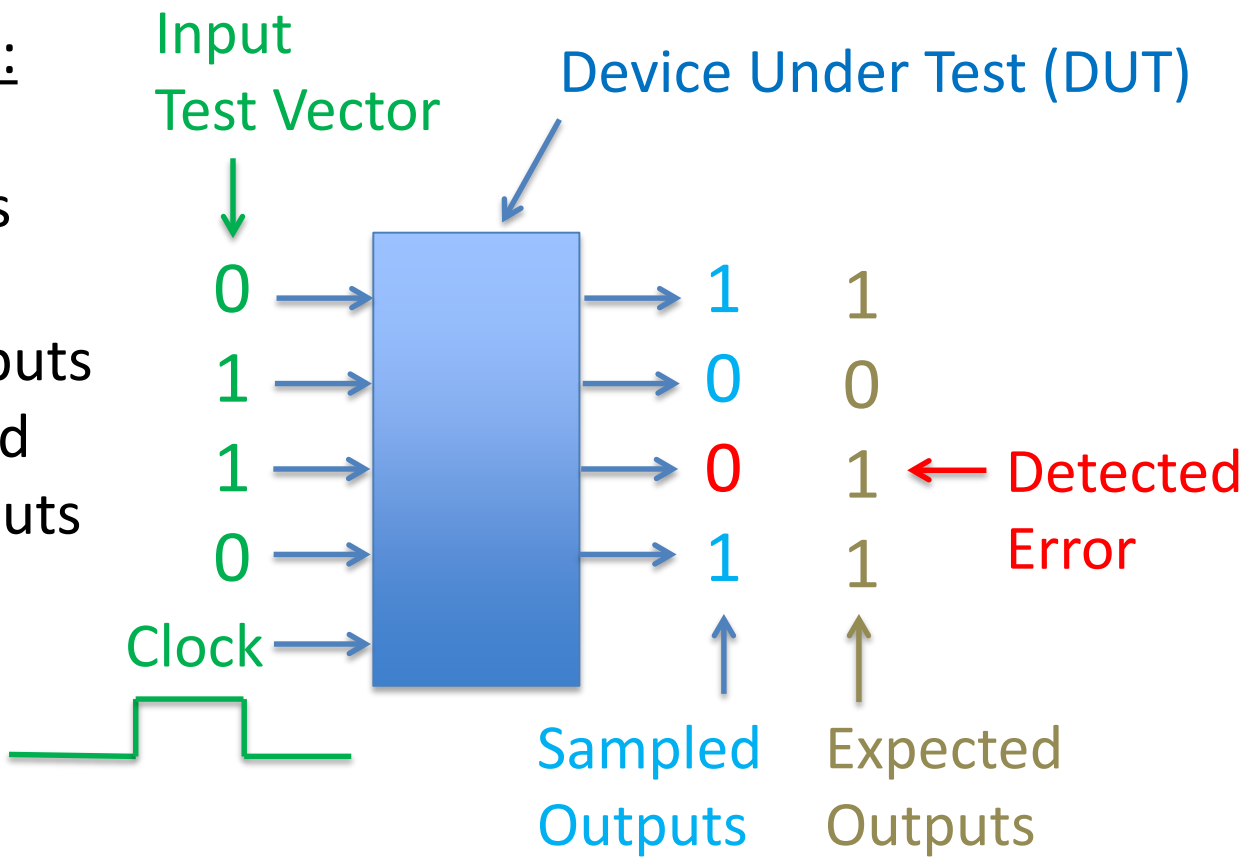
# Presentation outline

- IC testing process
- Tester architecture
- Device test fixture
- Test plan design
- 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



# ADVANTEST T2000 GS Test System

GS Mainframe:
- system controller
- site controller

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

Operator Station


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

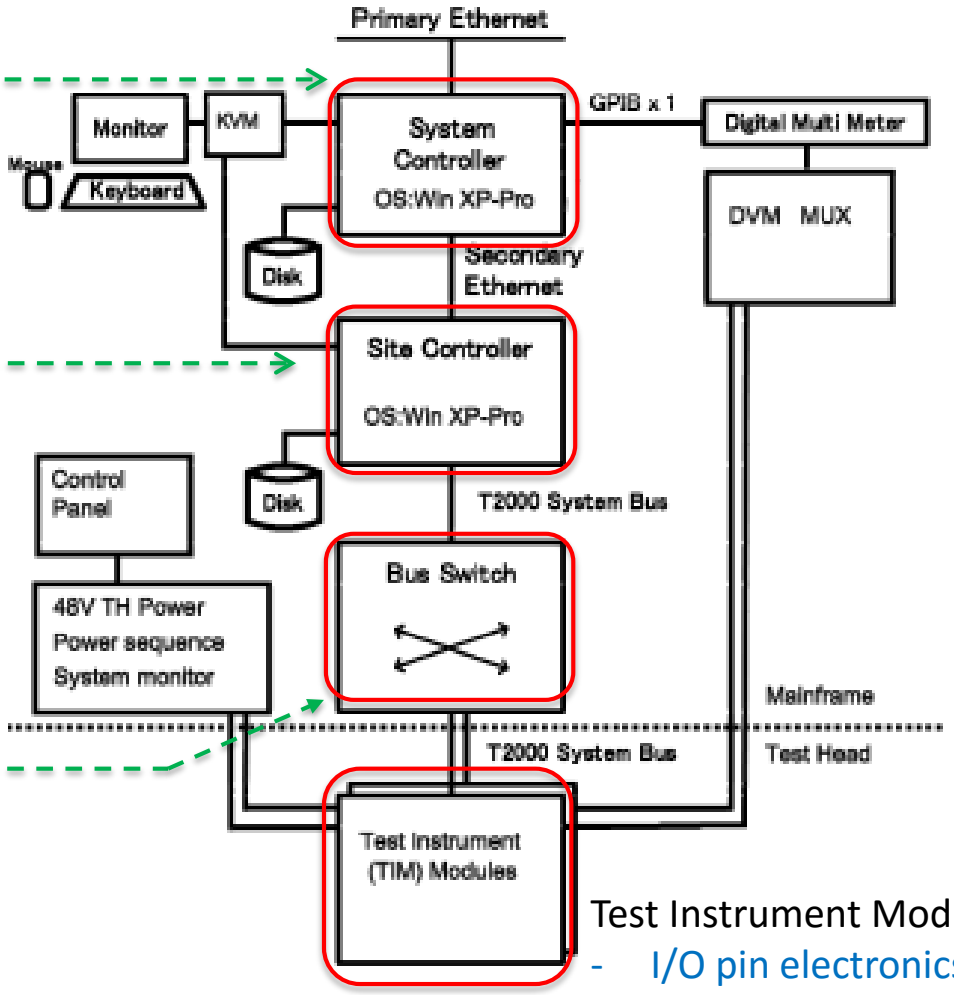
VLSI Test Lab
Broun Hall, Room 318

# T2000 GS computing architecture

System controller:
- User GUI 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

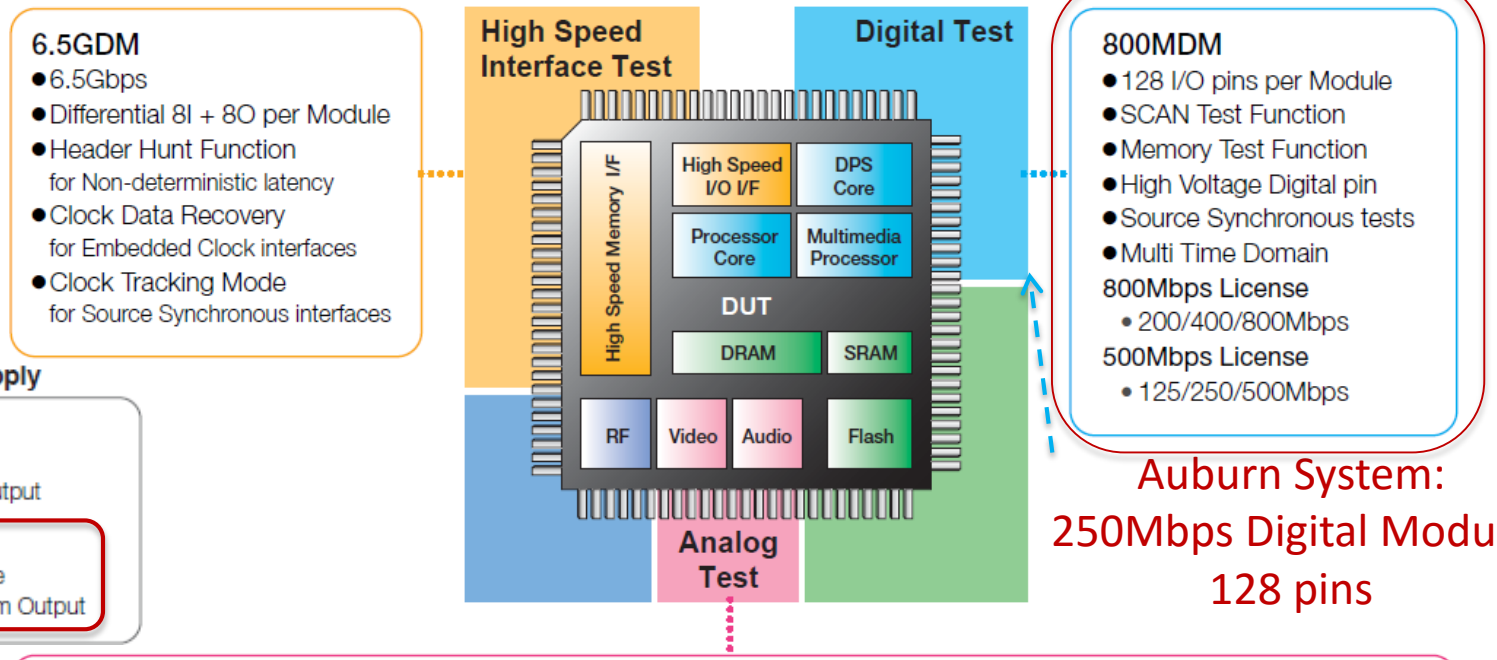
Bus Switch:
- Connects site controller to test modules
- Configured by a socket file



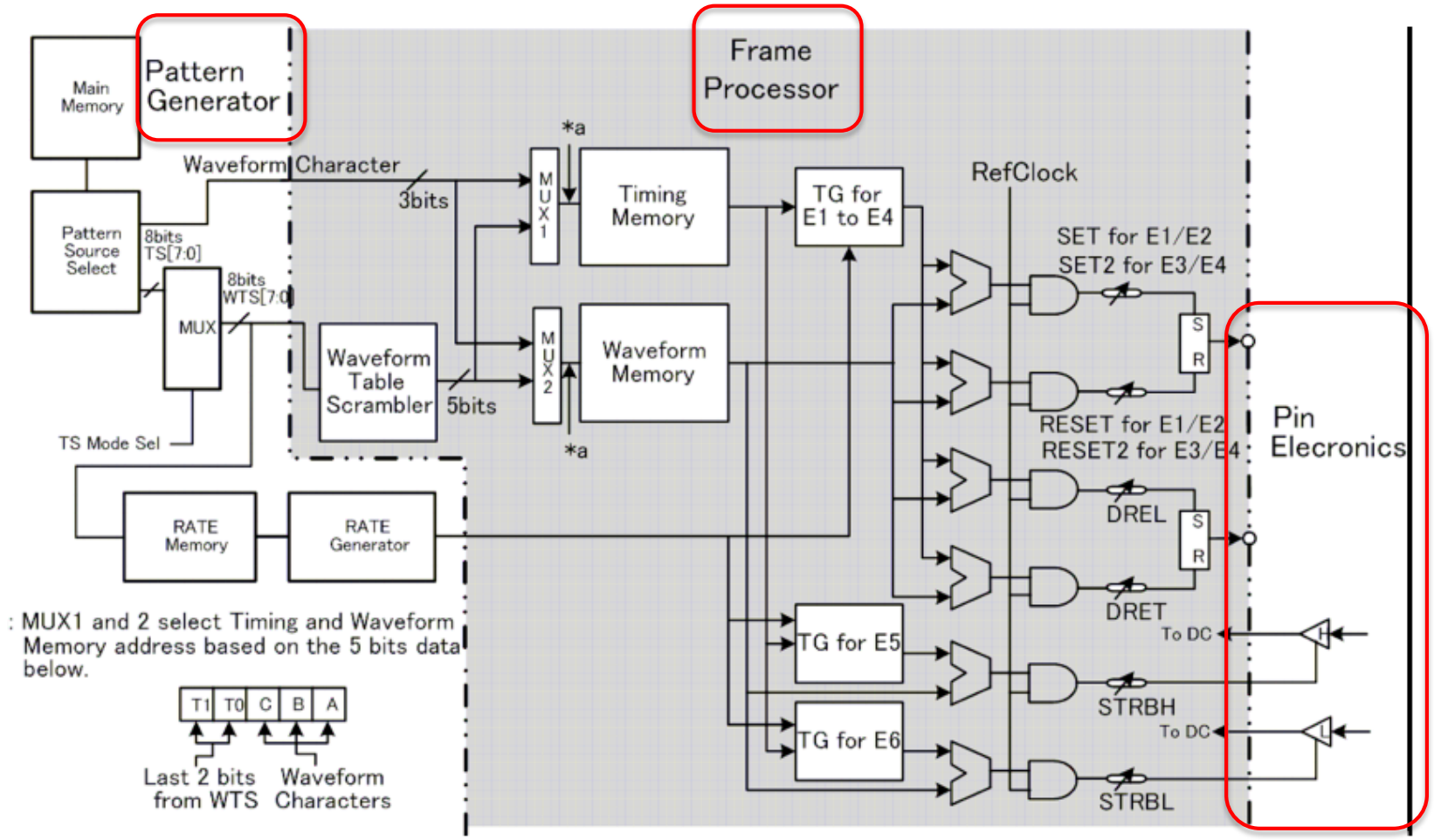

Test Instrument Modules
- I/O pin electronics
- Power supplies

# Test instrument modules

## (up to 12 in T2000 GS test head)



# 250MDMA pattern generator and frame processor



# Driver pin electronics

| Item | | 250MDMA Specification Value |
|---|---|---|
| **Voltage range** | **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 |
| **Minimum pulse width** | **50%** | 4.0nS @ 3V |
| | **50%** | 2.5nS @ 1V |
| **Driver minimum on time** | **Hi-Z** | 10.0nS |
| | **VTT** | 5.0 |
| **Driver minimum off time** | **Hi-Z** | 10.0nS |
| | **VTT** | 5.0nS |

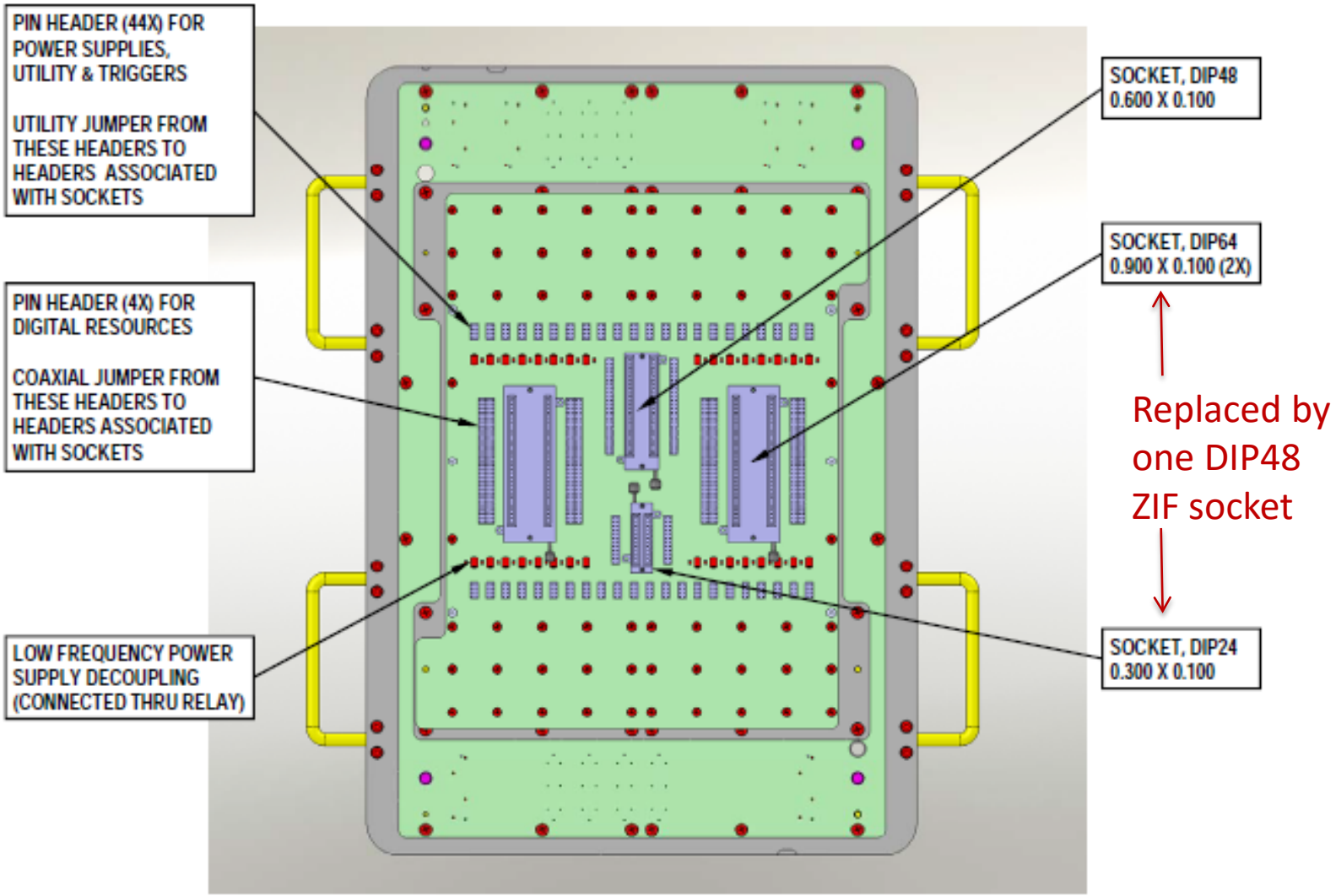
# Comparator specifications

| Item | | 250MDMA Specification Value |
|---|---|---|
| Voltage range | VOH | -1.25 ~ +6.75V |
| | VOL | -1.25V ~ +6.75V |
| Minimum voltage (VOH-VOL) | | 0.0V |
| Voltage resolution | | 2mV |
| Equivalent transition time | Hi-Z: 20% ~ 80% | 2.4ns @ 3V |
| | VTT: 20% ~ 80% | 2.0ns @ 1V |
| Window strobe minimum on time | | 4.0ns |
| Window strobe minimum off time | | 4.0ns |
| Window strobe minimum glitch detection | | 4.0ns |

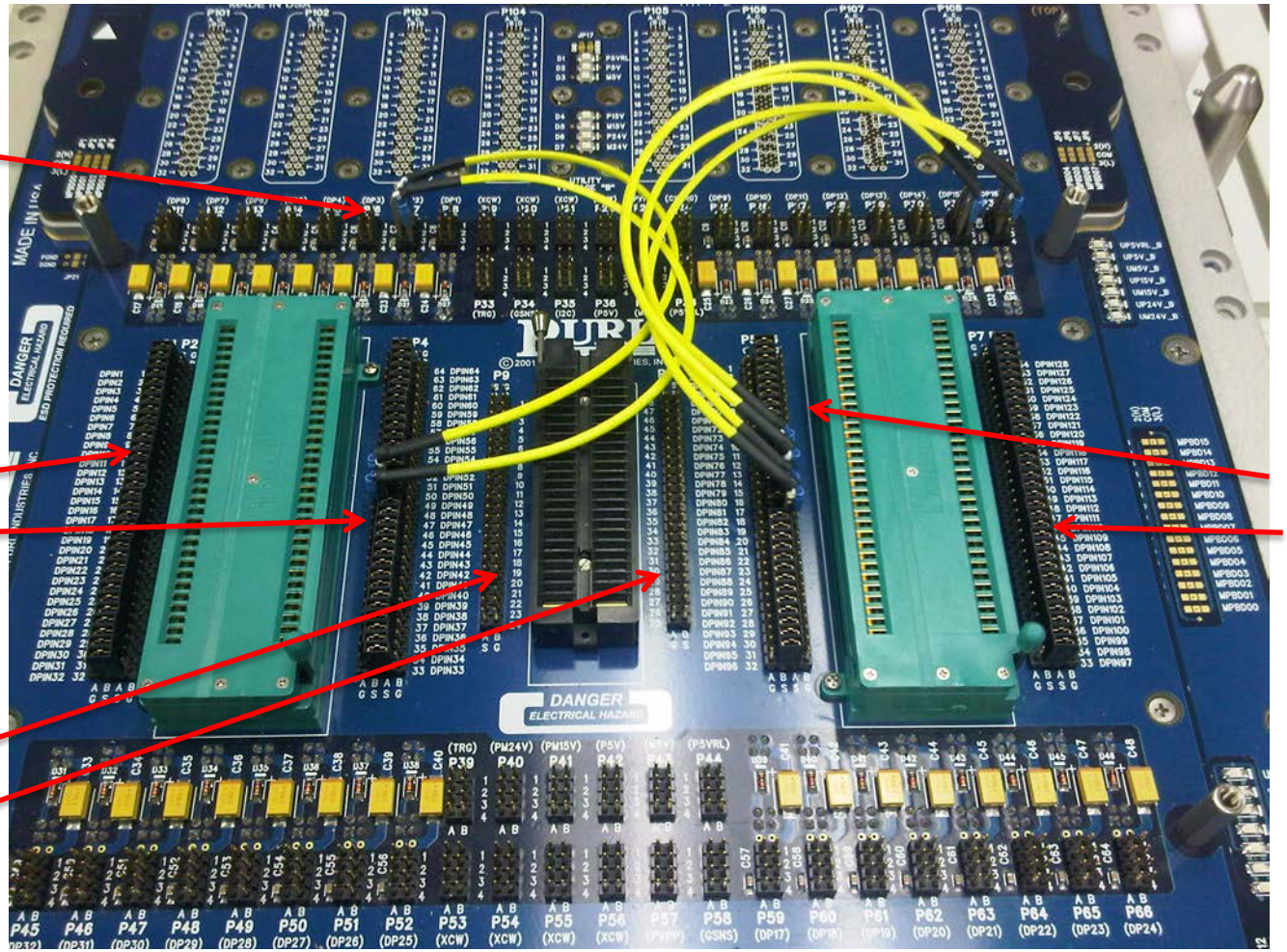
# Timing generator

| Item | 250MDMA Specification value |
|---|---|
| **Number of timing edges** | 6 timing edges per pin (4drive / 2compare) |
| **Number of waveforms** | 32 |
| **TS scrambler** | 256 |
| **Timing edge setting range** | 0s ~ 1ms<br>0ns ~ (4cycles - 4ns) |
| **Timing edge resolution** | 7.8125ps |
| **Edge shift** | Not Available |
| **Long-range timing** | Exists |

# Auburn T2000 performance board

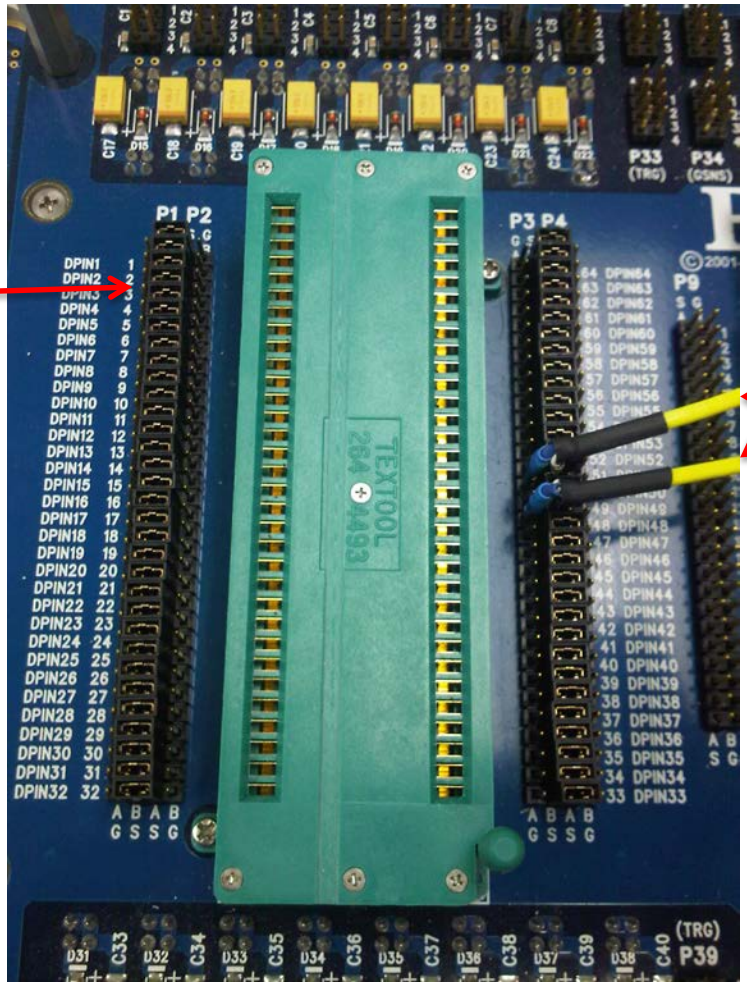


# Performance board IC sockets



# Configuring DUT signal/power pins

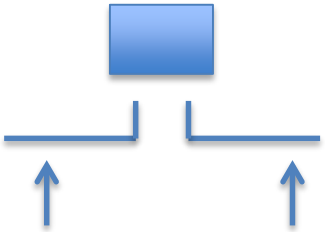
Leave shorting plugs to connect DUT pins to 250MDMA




Remove shorting plugs on DUT power/ground pins.

Connect DUT pwr/gnd pins to power supply

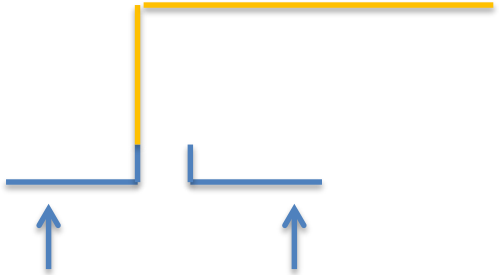
Shorting plug



250MDMA channel

DUT socket pin

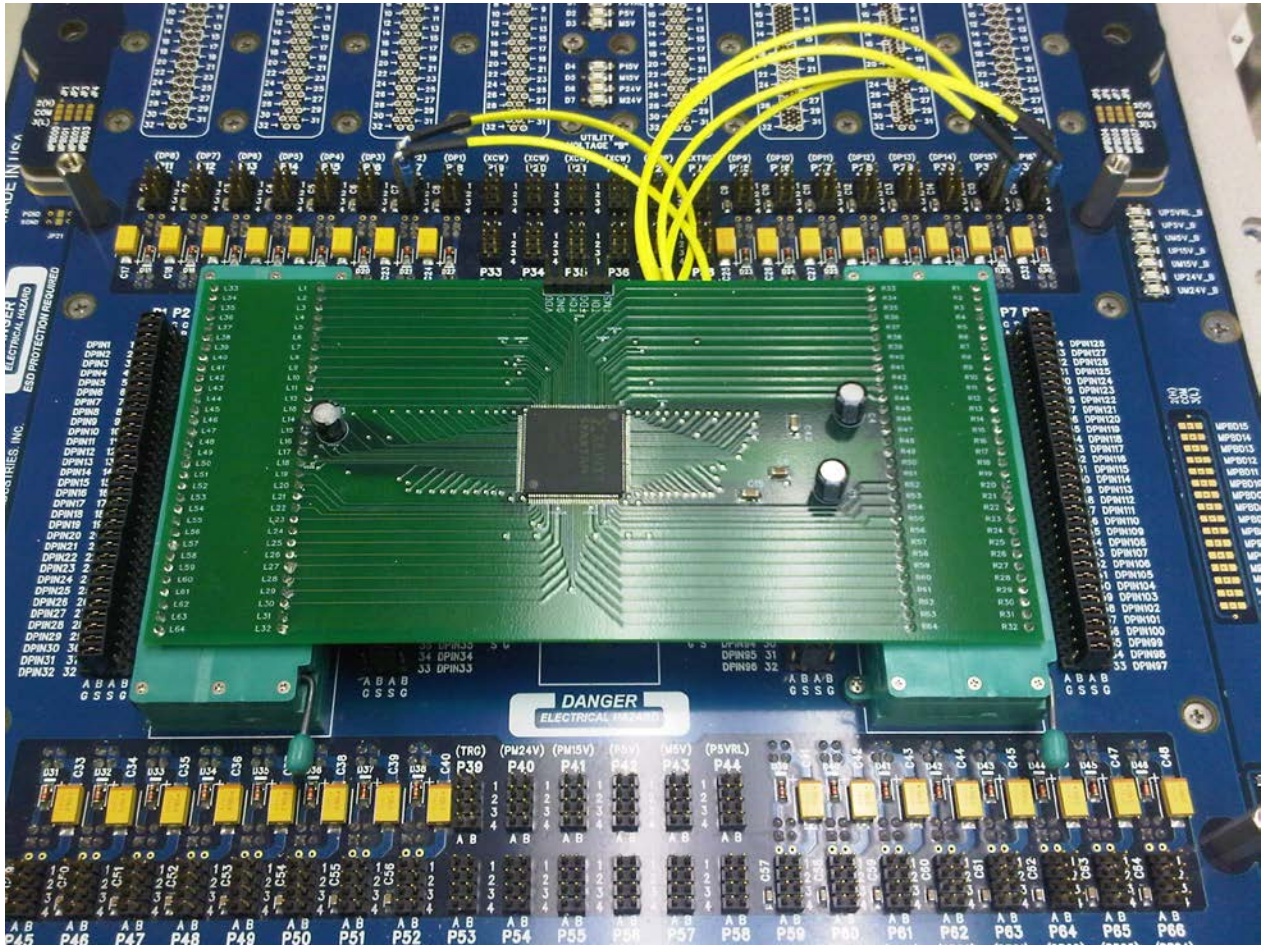
To power supply



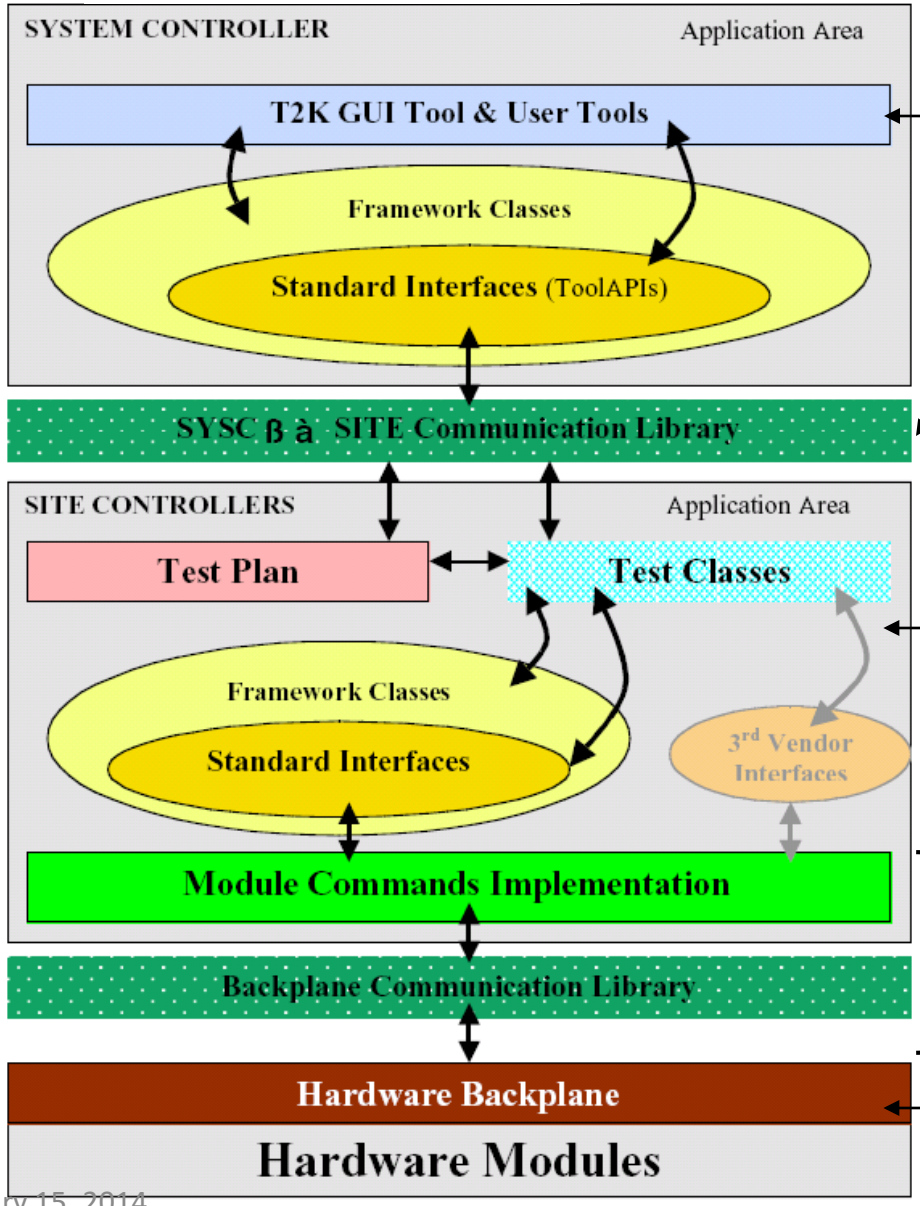
DUT socket pin

250MDMA channel

# Xilinx Spartan 3 FPGA daughter board mounted on the PB



# 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.

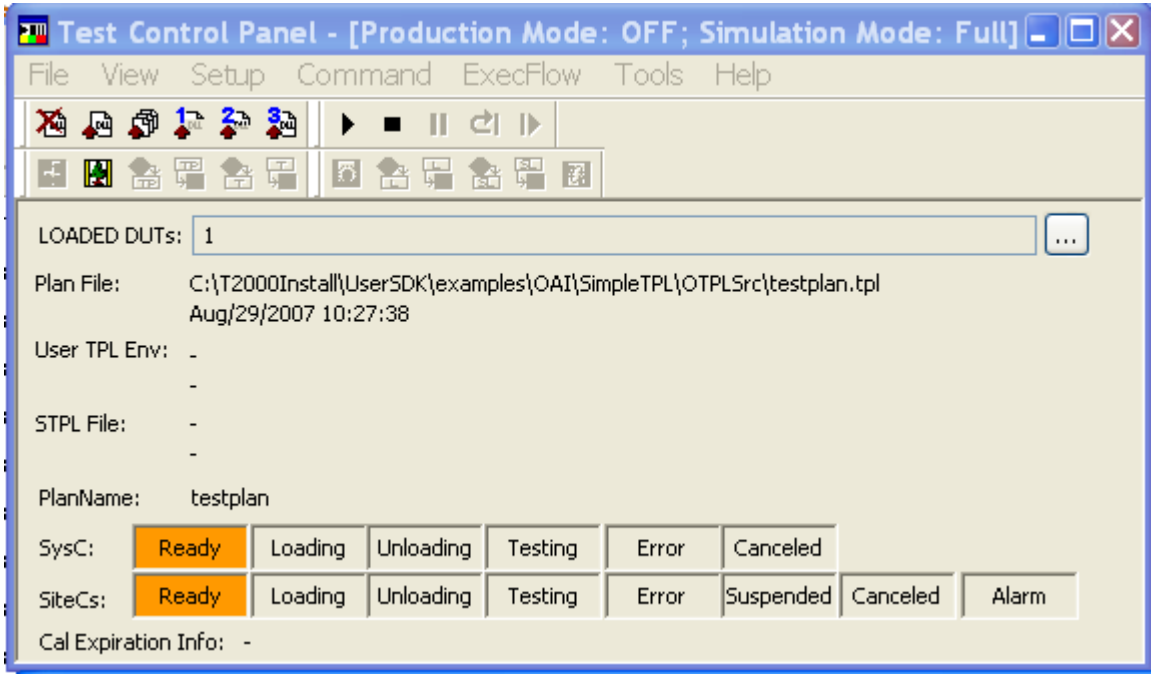
Module Backplane provides **optical/electrical I/F to individual test modules.**

# Test Plan

- *Test plan* = test program written by test engineer.
  - Defines the test flow (sequence of test steps)
  - Executes on the Site Controller
    - SC 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 via standard interfaces
    - test plans interact with common test system hardware components and other test-related objects.

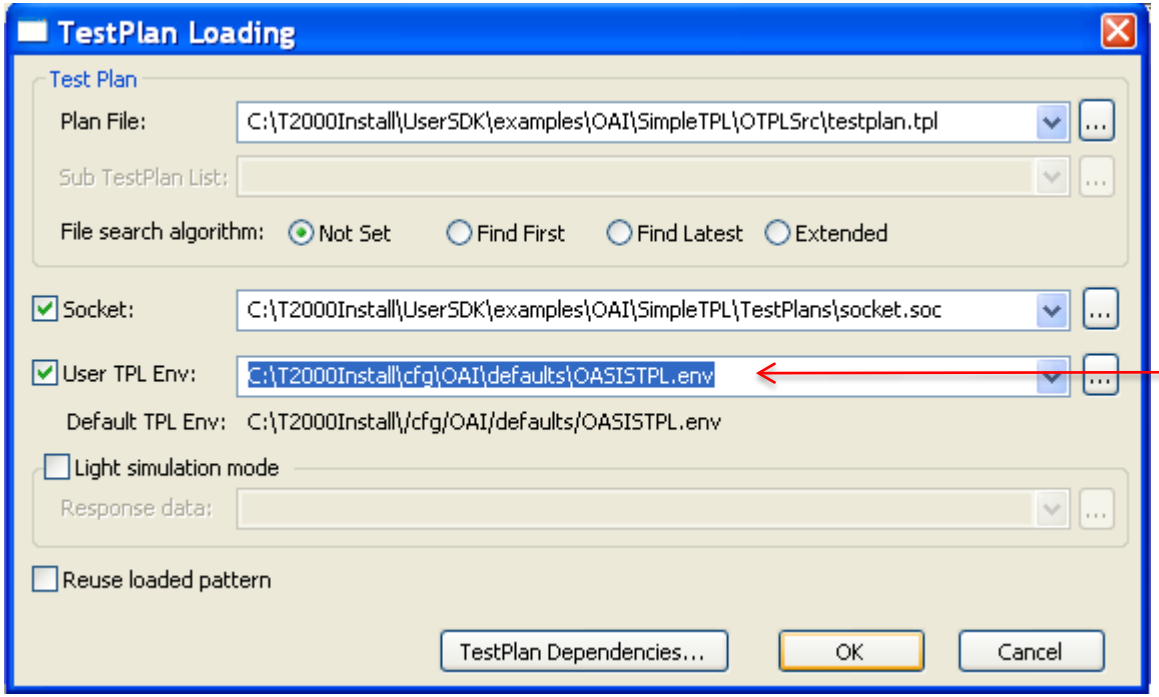
# T2000 control panel
## (*t2kctrl start – from a DOS window*)

- GUI to load/unload test plans
- Open other tools:



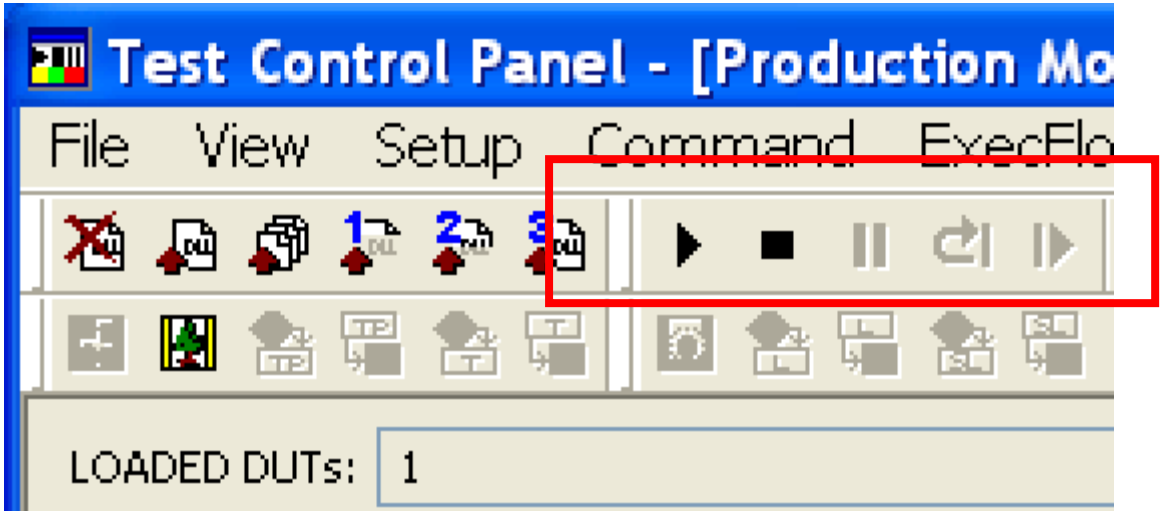
# Loading the test plan

From Control Panel, select: *File > Load Test Plan*




Environment file

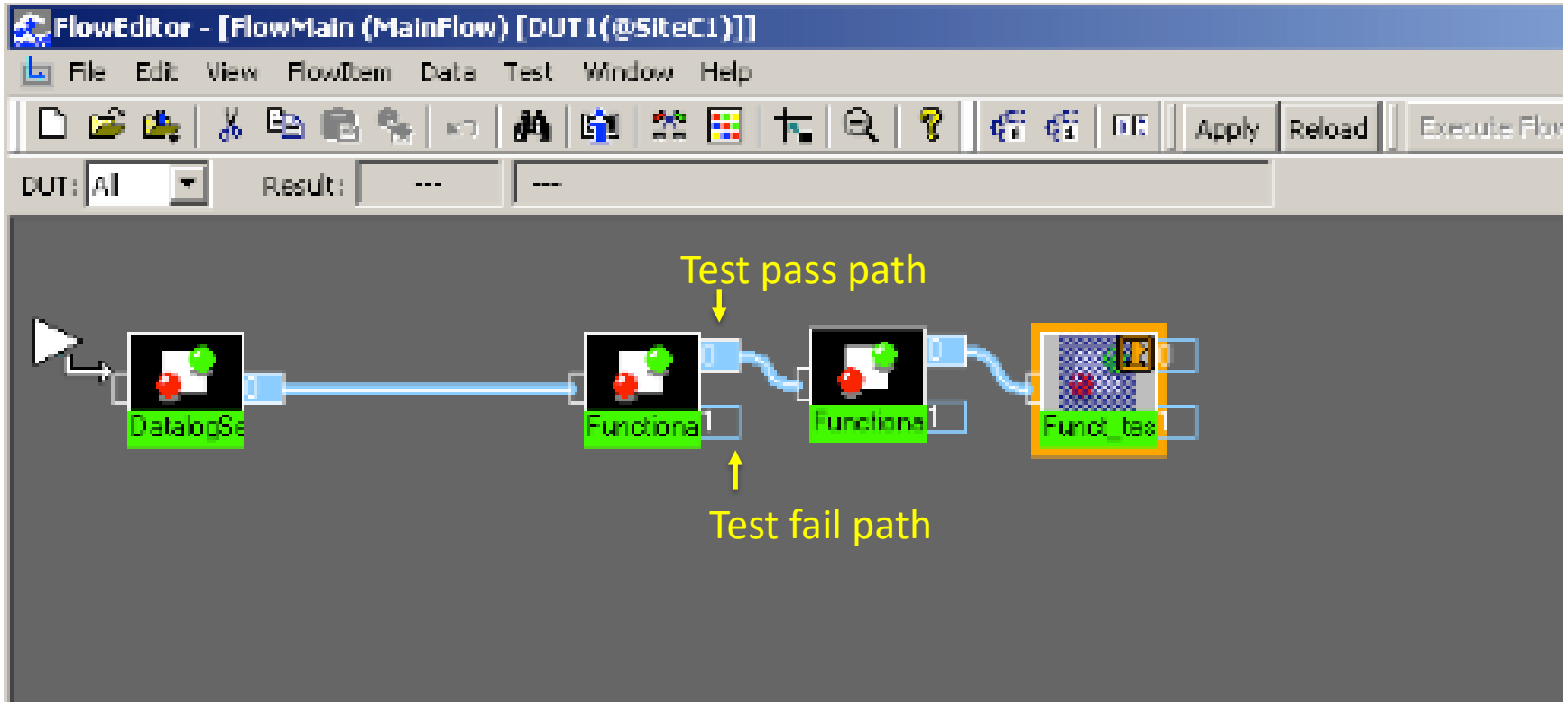
# Test Control Panel



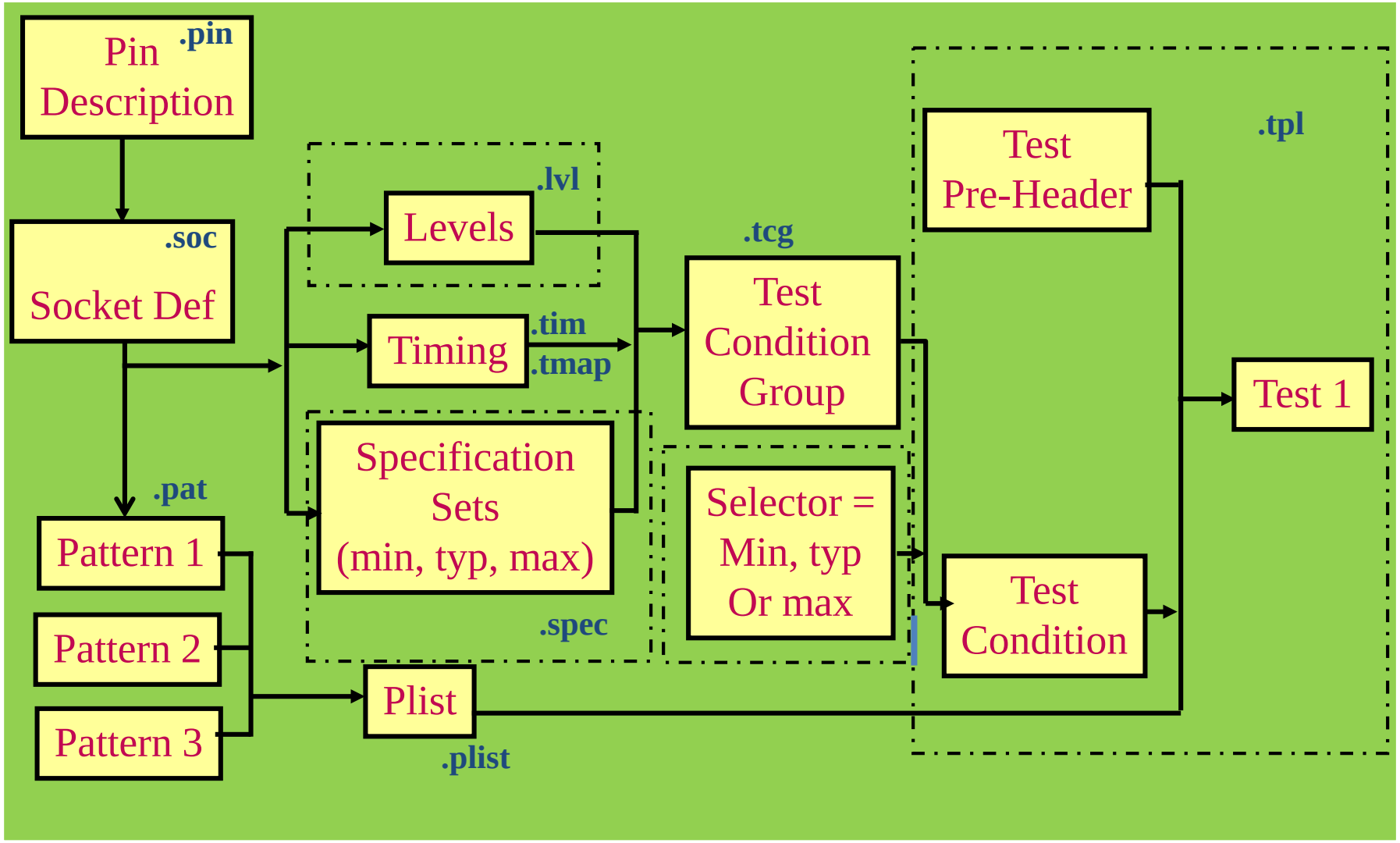

| Icon | Description |
|---|---|
| ▶ | Shortcut to Command -> Start |
| ■ | Shortcut to Command -> Stop |
| ⏸ | Shortcut to Command -> Suspend |
| ↻ | Shortcut to Command -> Reset |
| ⏭ | Shortcut to Command -> Continue |

# Flow editor

Control and/or edit the main test flow



# OPTL Test Plan Structure

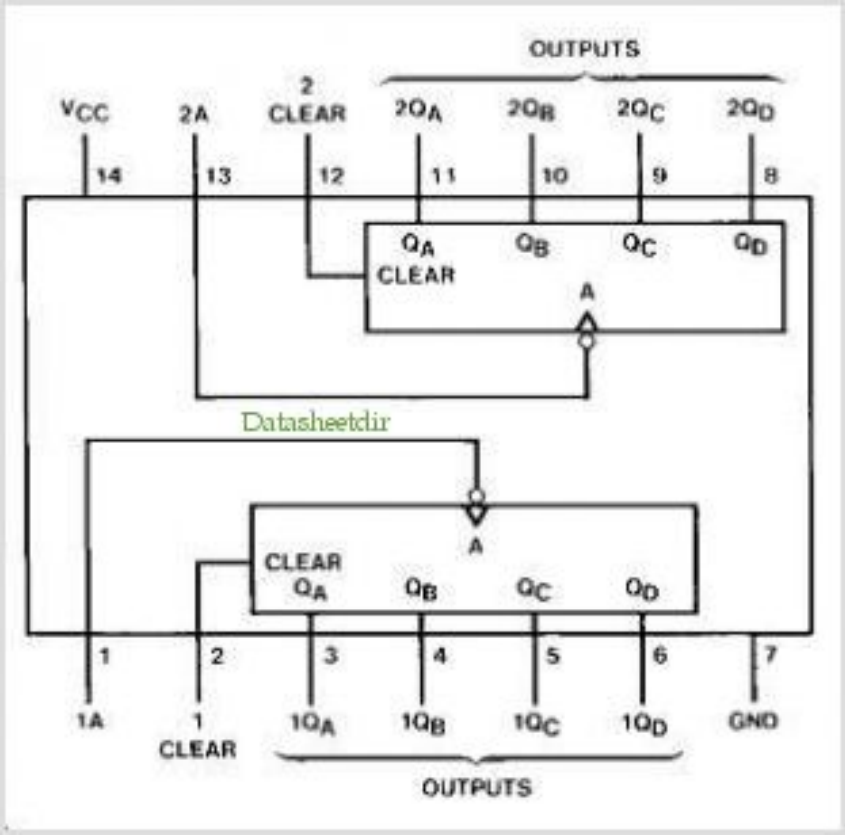


# OTPL test plan directory structure

/MyTestPlanFiles – create for each “project”

- /OTPLOutput – compiler output
- /OTLPSrc – test plan source code
- /Patterns – test pattern source files
- /Plist – pattern list files
- /TestClasses – class DLL files
- /TestPlans – compiled test plan and pin/socket files

# Example – 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

(OTPL requires strict formatting)

Pins controlled/observed as groups in the test plan

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

All individual pins

```
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

# 74LS393 “socket file” (.soc)

## Tell test plan which DUT pins connected to which 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;
         }
         Resource dps500mA
         {
            VDD   1010.2;
         }
         Resource moduletrigger
         {
            PMDTR0 1003.129;
            PMDTR1 1003.130;
            PMDTR2 2003.131;
            PMDTR3 2003.132;
         }
      }
   }
}
```

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

connector.channel

DPS500ma connector:
1010.1 .. 32

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

# 74LS393 device “specification file” (.spec)

## Voltage/current specifications (from device data sheet)

## Value chosen from multiple options by a selector

```
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;
}
```

- Select min/typ/max for test condition

From DUT perspective:

* Drive DUT inputs to vih/vil
* Threshold for DUT outputs = voh/vol

# Levels file (.lvl)

## Voltages/currents for DUT signal pin groups, Force voltages for DUT power supply pin groups.

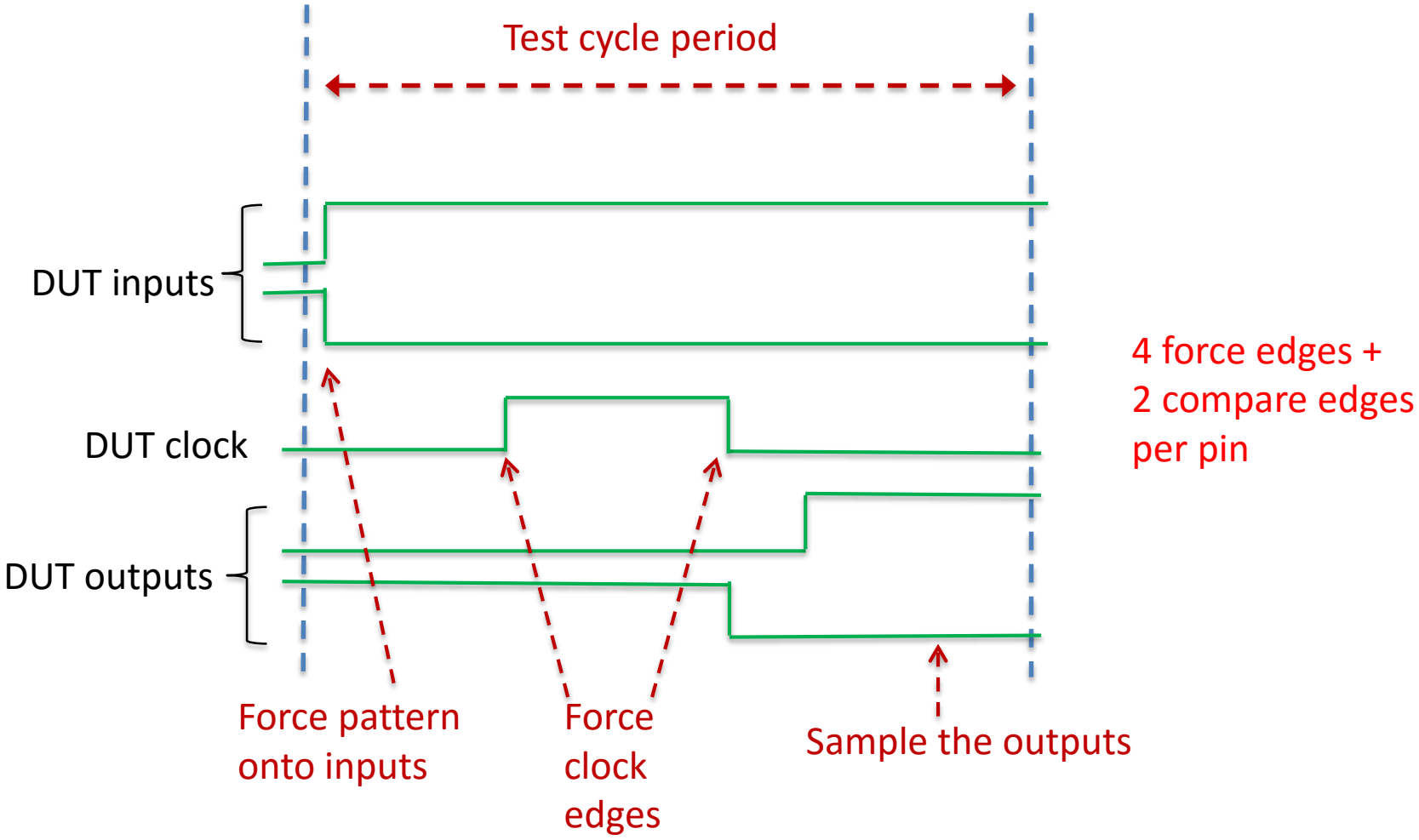
```
Version 1.0;
Import pindesc.pin;
# pindesc.pin declares names:
#       VDD, inpins, outpins
# resource.rsc declares names:
#       VSRange, VForce, Relay, VIH, etc.
Levels Lvl1
{
        VDD
        {
                VSRange = 7V;
                VForce = vforce;
                DpsRelay = CLOSE;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

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

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

Driver voltages defined in spec file

Reference voltages defined in spec file

# Test pattern timing – for each test vector

May define different timing patterns for different pins and/or test steps.



# 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 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; } }
            }
}
```

Edge

Test pattern Symbols

Sample High/Low

Sample TIme

# Timing file example

## Test engineer wanted to repeat tests for different periods.

```
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 inputs at start of period

Different output sample times 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

Waveform Set

Period table

Input waveform table

Output waveform table

From .tim file

# Test condition group file (.tcg)

## Combine: specification set + levels + timing

## (one set of test conditions)

```
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
        }
}
```

# Pattern files

## (pin order taken from .pin file)


```
NOP  { V { inpins=0111; outpins=LLLLLLLL; } W {allpins=wfs1;}}
NOP  { V { inpins=0100; outpins=LLHLHHLL; } }
NOP  { V { inpins=0110; outpins=LLHLHLLL; } }
NOP  { V { inpins=0110; outpins=HHLLLLLH; } }
  ....
NOP  { V { inpins=0111; outpins=LLLLLLLL; } W {allpins=wfs2;}}
NOP  { V { inpins=0100; outpins=LLLLLLLL; } }
NOP  { V { inpins=0110; outpins=LLLLLLLL; } }
NOP  { V { inpins=0110; outpins=LLLLLLLL; } }
NOP  { V { inpins=0111; outpins=LLLLLLLL; } }
  ....
NOP  { V { inpins=0111; outpins=LLLLLLLL; } W {allpins=wfs3;}}
NOP  { V { inpins=0100; outpins=LLLLLLLL; } }
NOP  { V { inpins=0110; outpins=LLLLLLLL; } }
....
```

Sequencing instruction

# 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

# 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) 1 => pulse 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 - ASCII test file

## (to be converted 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;
```

Test patterns

# Test plan (.tpl)

## Specify test conditions and test flow

```
Version 1.0;

# Import OTPL sources & pre-headers
Import testcondition.tcg;
Import asicbins.bdefs;
Import DatalogSetupTest.ph;
Import FunctionalTest.ph;

# Import Runtime files
Import pindesc.pin;

#-------------------------------------------------------
# Start of the test plan
#-------------------------------------------------------

# Name of the TestPlan
TestPlan testplan;

# The type of DUT
DUTType "DiagPB";
```

```
PListDefs
{
        # Pattern lists for this test plan  (file:object)
        pattern.plist:DiagPBPat
}


# SocketDef, UserVars declaration as before ...
SocketDef = socket.soc;
```

# Test plan (continued)

```
# Declare conditions for tests: 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
```

Use “typical” values from this group

Pattern list for this test

Conditions for this test

# Test plan (continued)

## (define the test flow)

```
# 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;
                }
        }
```

# Test plan example – FPGA

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

```
# Define the three functional “tests”
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

# FPGA Test Plan (continued)

## (Define the test flow)

```
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 FPGA

# Other test options

## (Students might want to try these)

- Scan-based testing
- DC Parametric Test
  - Per-pin parametric measurement unit
- IDD tests
- SHMOO plots
  - Modify variables over a range and plot #pass/fail vec's
- Complex timing (ex. double data rate)
- Binning (hard and soft)
  - Control handler to move failed parts to bins