TCE User Group Meeting
Working With FLUSH

Presented by: Stephen Fu

31-Mar-2014

Overview

- What's FLUSH
- FLUSH Syntax
- Experiment on FLUSH API
  - Changes to Setup Data Without Flush
  - Changes to Hardware Without Flush
  - Changes to Setup Data With Flush
  - Changes to Hardware With Flush
  - Changes to Hardware With Flush APRM
  - Changes to Hardware With Functional Test
- Summary of FLUSH Effect
- APIs contained operation of FLUSH
- Things to Remember

What's FLUSH

FLUSH is a semi-automatic mean to minimize HW access and to optimize HW setups to achieve better test time.

- This API reflects the changes specified by the following APIs to the setup data or hardware:
  - ANALOG
  - ANALOG_STATUS
  - ROUTING_STATUS
- And also, this reflects the changes specified by the following APIs to the MCD (firmware level):
  - PRIMARY_STATUS
  - TIMING_SPEC / LEVEL_SPEC / SPECIFICATION

FLUSH Syntax

- Syntax

  void FLUSH();

  void FLUSH(const TM::FLUSH_POST_ACTION action);

- Parameters

action	This parameter controls how the setup data will be downloaded to the hardware. You can leave it empty or use TM::APRM. When using the TM:APRM option, the APRM CNL firmware command will be issued following the normal FLUSH() actions.

APRM, APRM? - (Activate PRiMary sets)
The APRM command activates primary sets in the hardware

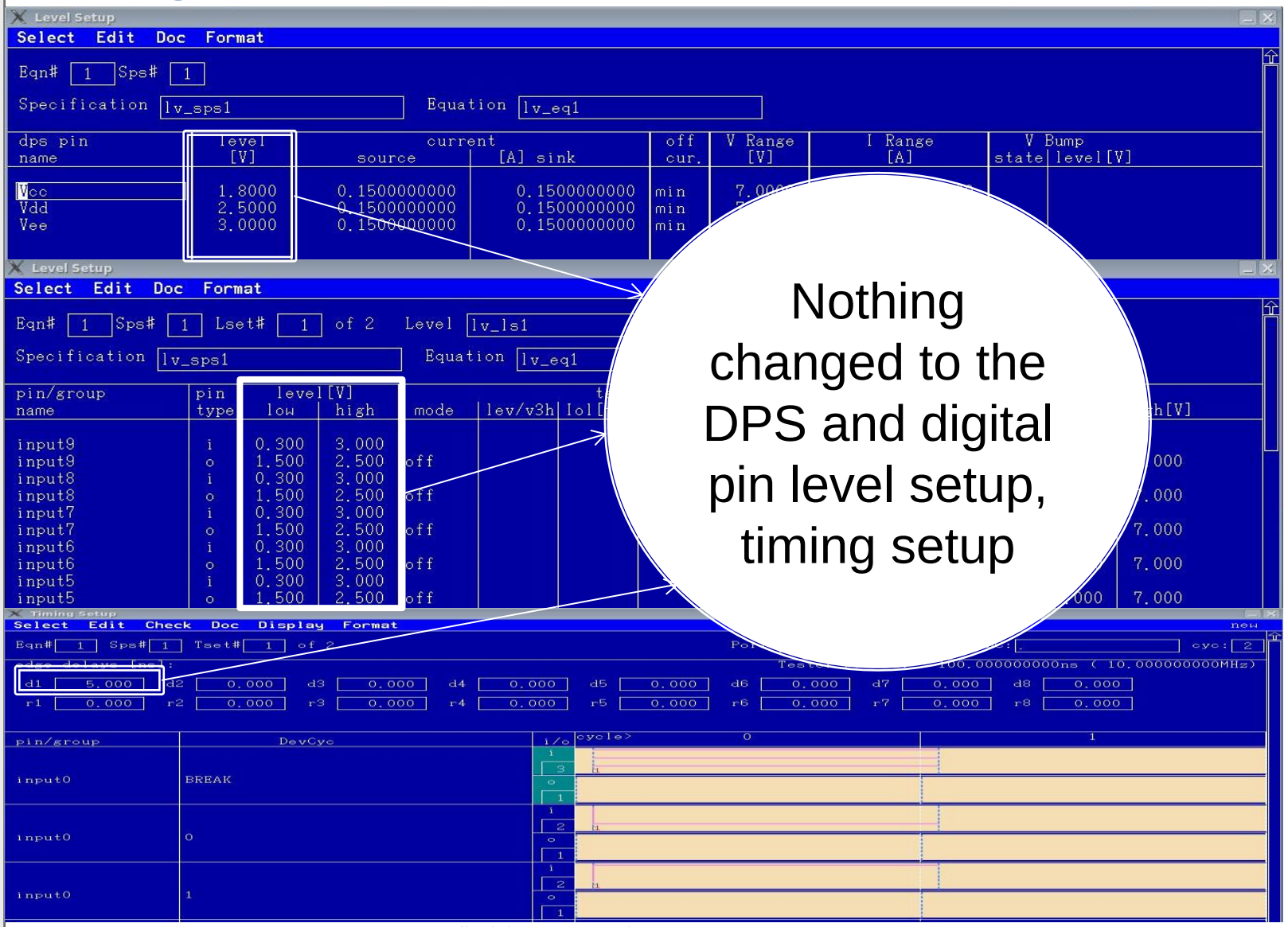
Experiment on FLUSH API

```
48  virtual void run()
49  {
50    //Add your test code here.
51
52      CONNECT();
53
54      //Primary_status ,timing_Spec,level_Spec,specification
55      Primary.getLevelSpec().change("vdd1",1);
56      Primary.getLevelSpec().change("vdd2",2);
57      Primary.getLevelSpec().change("vdd3",3);
58
59      Primary.getLevelSpec().change("vil_lel",0.5);
60      Primary.getLevelSpec().change("vih_lel",3.3);
61      Primary.getLevelSpec().change("voh_lel",3);
62      Primary.getLevelSpec().change("vol_lel",2);
63
64      Primary.getTimingSpec().change("t1",10);
65      Primary.getTimingSpec().change("t2",30);
66
67      //Routing_status
68      Routing.util("ut101").on();
69      Routing.pin("Ain_p").connect();
70
71      //Analog_status
72      Analog.AWG("Ain_p").enable();
73      ANALOG_SET AnalogSet_01("AnalogSet_01");
74      ANALOG_WAVEFORM Wave_Adc_Sine("ADC_Sine");
75      Wave_Adc_Sine.definition(TM::SINE).periods(7).phase(180).min(-1.25).max(1.25).samples(2048);
76      ANALOG_SEQUENCER Seq_ADC_Sine_seq("ADC_Sine_seq");
77      Seq_ADC_Sine_seq.add(TM::RPT,Wave_Adc_Sine,5);
78      Seq_ADC_Sine_seq.add(TM::RPT,Wave_Adc_Sine,5);
79      AnalogSet_01.CLOCK_DOMAIN(2,TM::ANALOG).clock(TM::MCLK_AUTO);
80      AnalogSet_01.AWG("Ain_p").clockDomain(2).rule(TM::FIXFS).frequency(100E6);
81      AnalogSet_01.AWG("Ain_p").coreFunction(TM::HF).attn(10).vOffset(0.1 V).filter("15M").sequencerProgram(Seq_ADC_Sine_seq);
82
```

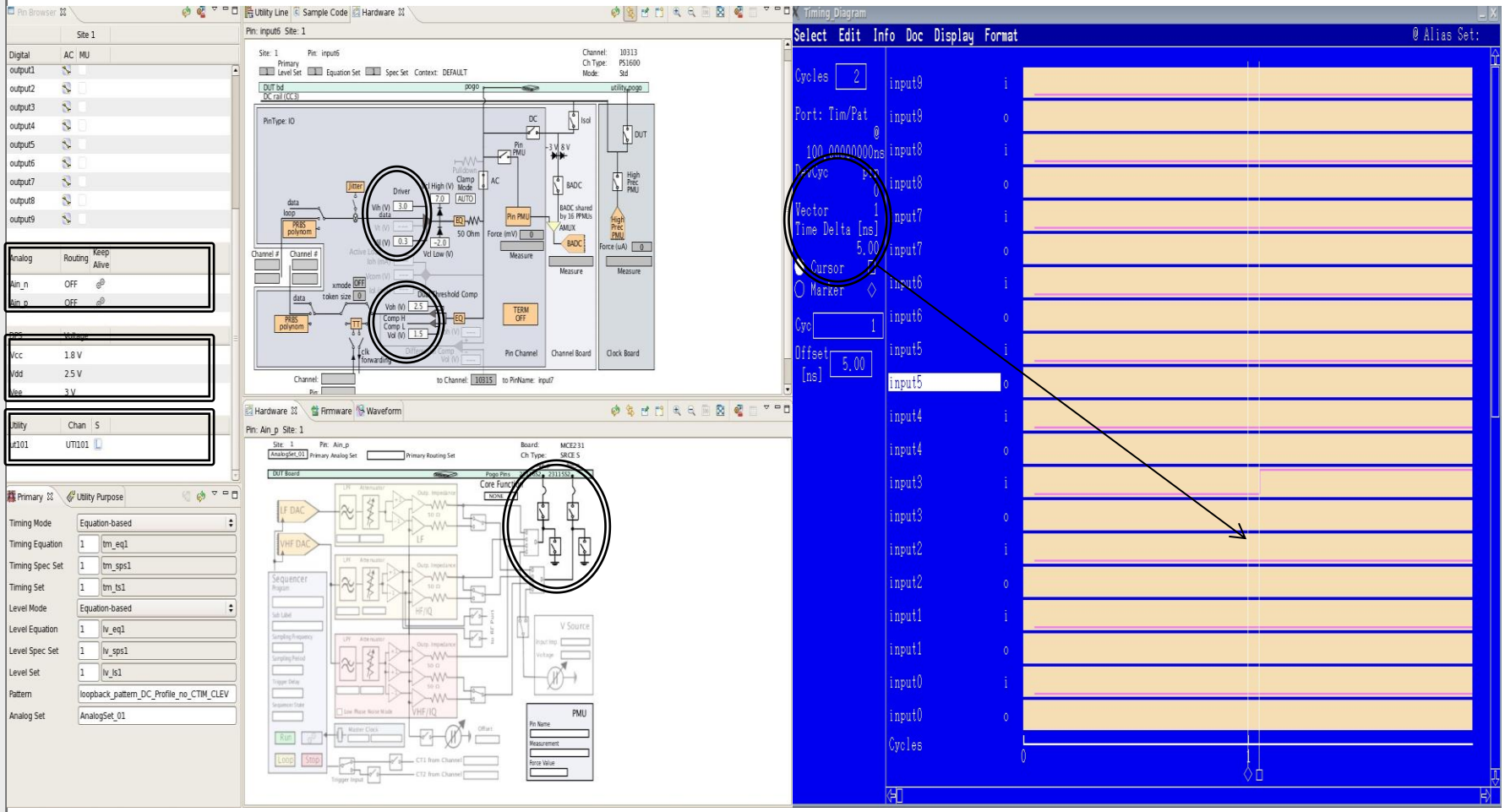
In the testmethod, Change the SpecVariable values by timing_spec and level_spec API, specify the routing setting data by Routing API, Create Analog set using analog API.

Check the different effect to setup data and hardware with or without FLUSH and FLUSH APRM, Functional Test API.

Changes to Setup Data Without Flush

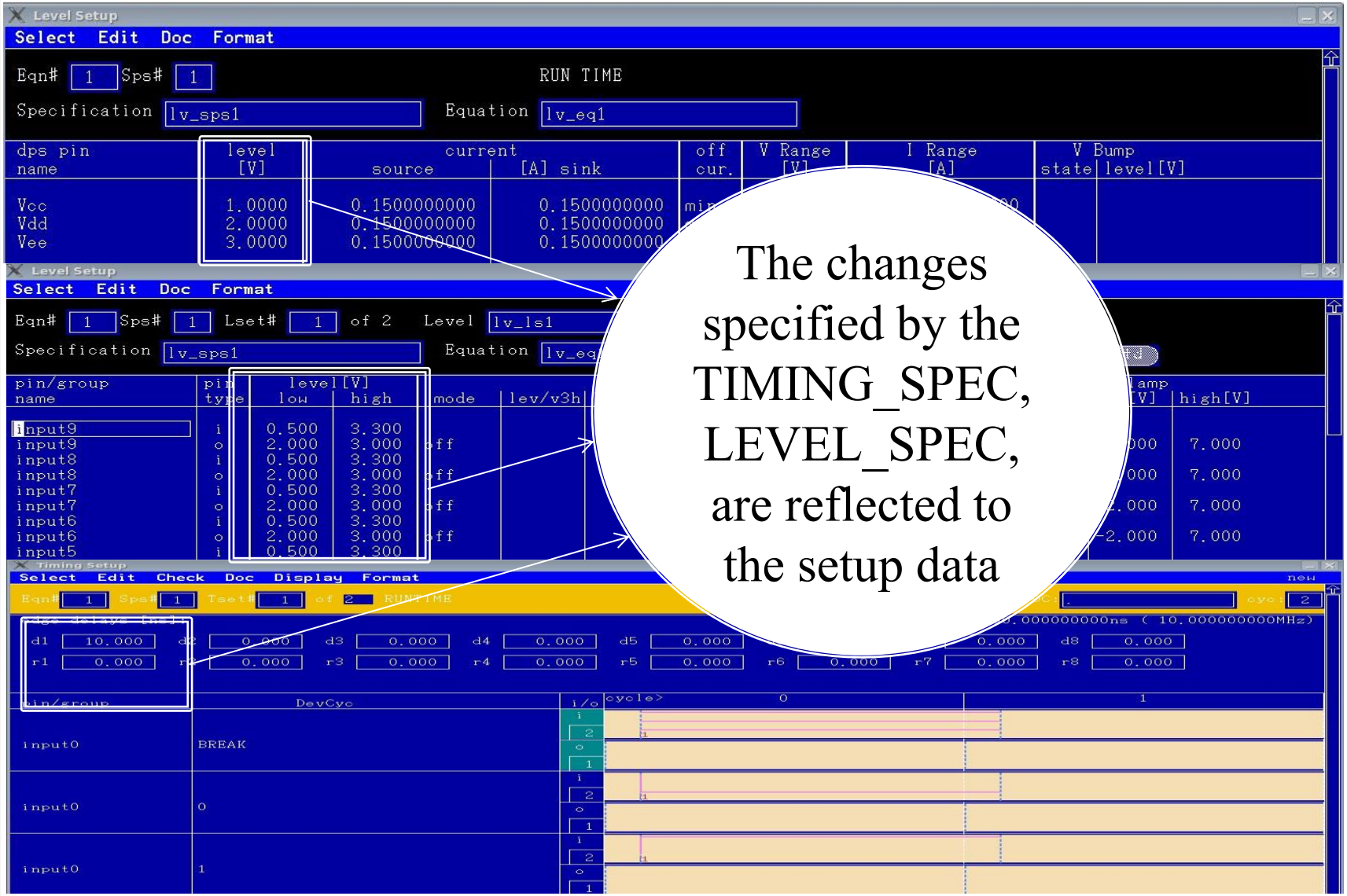


Changes to Hardware Without Flush



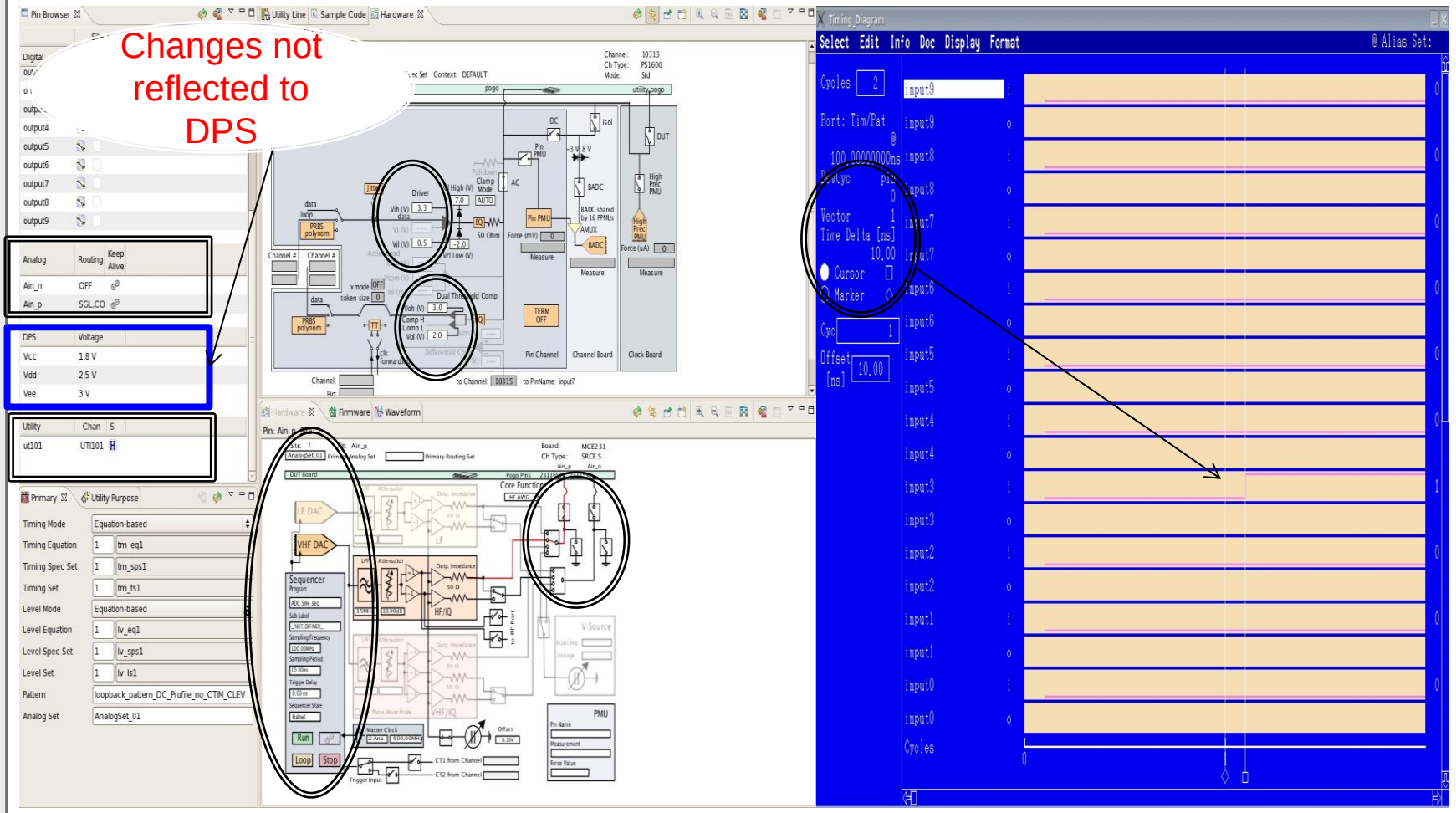
The changes specified by the TIMING_SPEC, LEVEL_SPEC, and Routing, Analog APIs are not reflected to the hardware.

Changes to Setup Data With Flush



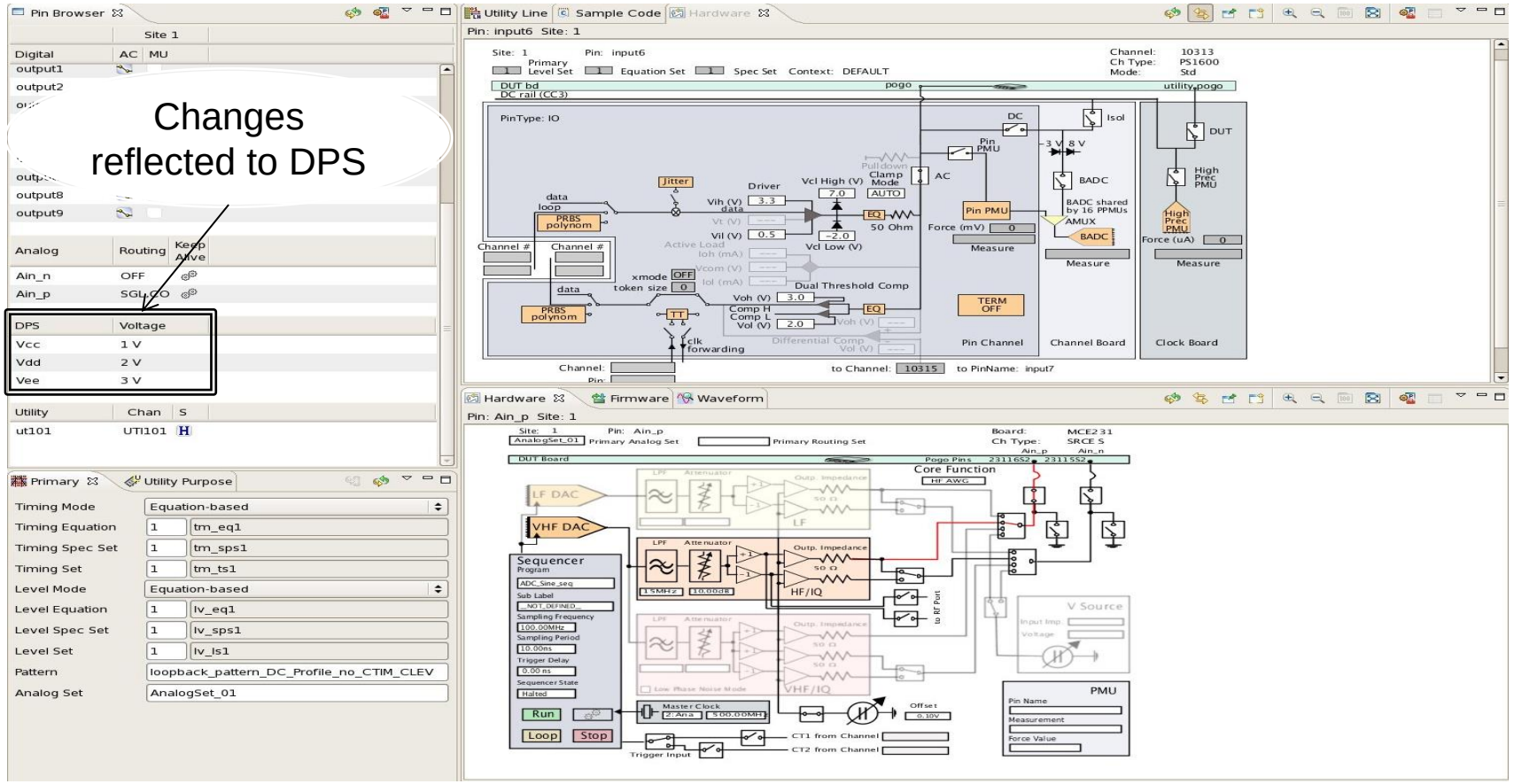
The changes specified by the TIMING_SPEC, LEVEL_SPEC, are reflected to the setup data

Changes to Hardware With Flush



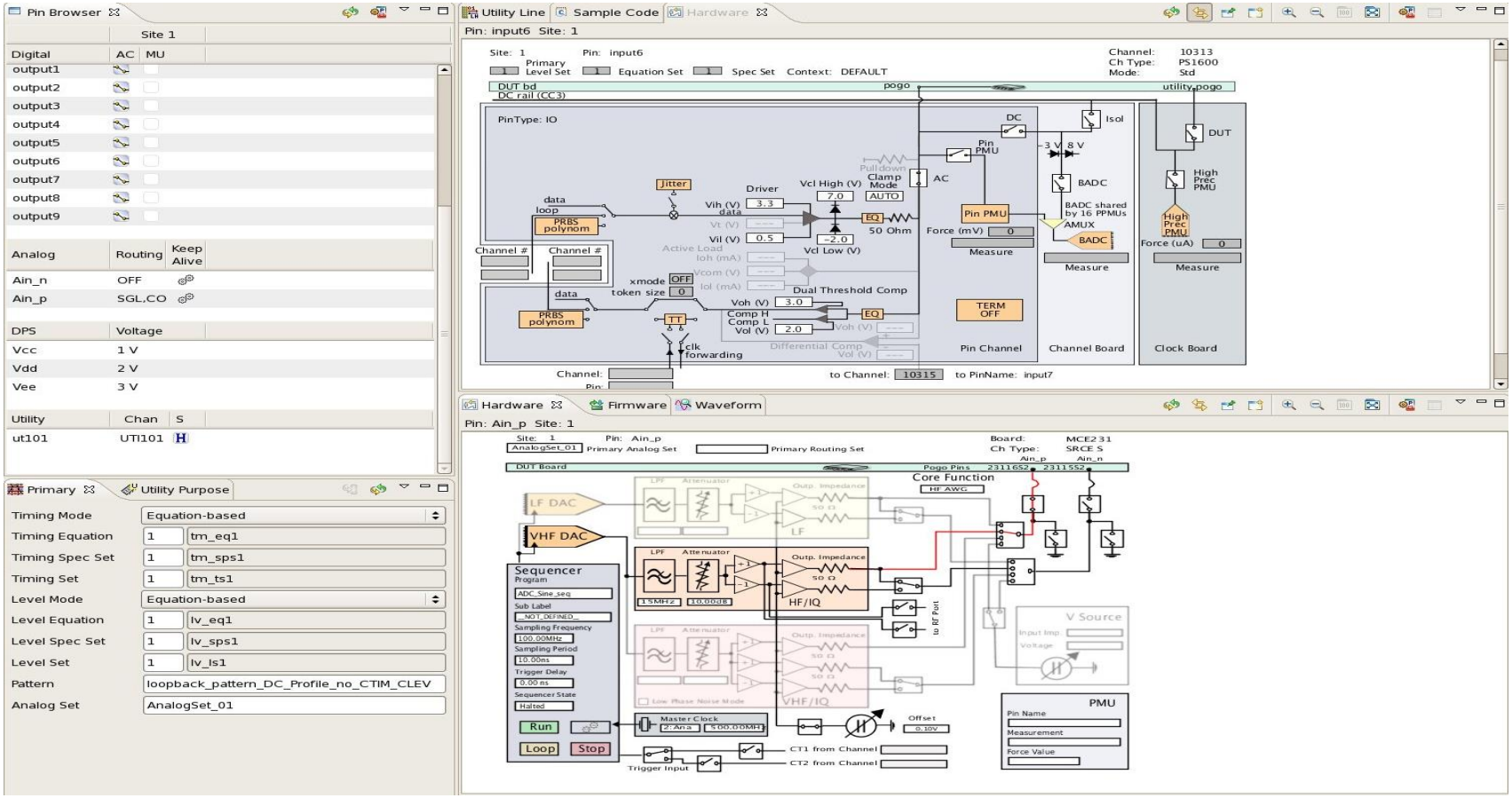
The changes specified by the TIMING_SPEC, LEVEL_SPEC, and Routing, Analog APIs are reflected to the hardware except for the changes to DPS.

Changes to Hardware With Flush APRM



The changes specified by the TIMING_SPEC, LEVEL_SPEC, and Routing, Analog APIs are reflected to the setup data and hardware including the changes to DPS.

Changes to Hardware With Functional Test



The similar effect with FLUSH APRM API, The changes are reflected to the setup data and hardware including the changes to DPS.

Summary of FLUSH Effect

Change Spec with APIs	Execute without Flush		Execute with FLUSH()		Execute with FLUSH(TM:: APRM)		Execute with FUNCTIONAL_TEST()	
	Changes reflect to Setup Data	Changes reflect to Hardware	Changes reflect to Setup Data	Changes reflect to Hardware	Changes reflect to Setup Data	Changes reflect to Hardware	Changes reflect to Setup Data	Changes reflect to Hardware
Primary.getLevelSpec (for DPS)	No	No	Yes	No	Yes	Yes	Yes	Yes
Primary.getLevelSpec (for digital channel)	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Primary.getTimingSpec	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Routing.util	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Analog set	No	No	Yes	Yes	Yes	Yes	Yes	Yes

APIs contained operation of FLUSH

The changes specified by the PRIMARY_STATUS, TIMING_SPEC, LEVEL_SPEC, and SPECIFICATION are actually reflected to the hardware when any of the following APIs is executed.

- EXECUTE_TEST
- START_TEST
- DIGITAL_CAPTURE_TEST
- NB_DIGITAL_CAPTURE_START
- FUNCTIONAL_TEST
- FLUSH then SPEC_SEARCH (execute() member function)
- FLUSH then SEQUENCER_STATUS (run() member function)
- FLUSH then PMU_IFVM (execute() member function)
- FLUSH then PMU_VFIM (execute() member function)
- FLUSH then TASK_LIST (execute() member function)
- FLUSH then DPS_VFIM (execute() member function)

The FLSH also reads the buffered setting data in the current analog set and routing setting data in the primary routing set and defined by the ACMD "CNCT" command from the MCD, and downloads the contents to the hardware.

Things to Remember…

When using the FLUSH API, the following points should be considered:

- Each HW access takes time. Analog setups may get cached and activated (flushed) in a single shot when performing a FLUSH(). In addition the internal wait time can be optimized, which would be otherwise required for each single HW access.
- HW setup optimization means that FLUSH may recognize situations where it can avoid redundant setups (e.g. a disconnect, if the next test suite is performing a connection again).
- FLUSH() execution time is in a range of 1 to 10 and more ms dependant on the complexity of the HW changes.
- It scales with number of cores, actions, sequencer program length and tester period. MSE is in the range of 60%.
- Although FLUSH() tries to optimize execution time, unnecessary calls of FLUSH() will add test time.
- Debugging may require to put additional FLUSH()'s in the code to force immediate activation to allow visibility in debug tools, which needs to be removed later for production.

Example

Test program with 500 testsuites with one unnecessary FLUSH() each may end up with an overhead of 500 x 3ms = 1.5s

Q/A