Low Power Debug

1 Introduction

VCS Native Low Power (NLP) has the Native Verdi integrated, which can be used for debugging low power simulations. Verdi supports IEEE 1801 UPF. It allows you to visualize, trace, and analyze the source of low power events. It simplifies low power debug by allowing you to visualize the power intent (UPF) and supports features to determine whether an unexpected design behavior is caused by the functional logic or a power event.

This document describes the low power debug using Verdi in the following sections:

- Use Model
- Low Power Debug Features
- Signal Shading
- List of Default Static Debug Reports Dumped by VCS NLP
- Verdi Database

2 Use Model

For VCS and Verdi databases to be consistent for accurate debugging, VCS NLP instruments the low power objects in HDL design to mimic the power intent specified in the UPF. Verdi allows you to debug low power objects instrumented by VCS NLP.

The following is the syntax to compile the low power design:

```
% vcs -sverilog <design_file> -upf <upf_file> -power_top <design_top> -
kdb -debug_access+all <compile_options>
```

The following is the syntax to invoke Verdi GUI:

```
% verdi -sverilog <design_file> -upf2.0|-upf1.0 <upf_file> -ssf file.fsdb
```

Example: `% verdi -sverilog *.v -upf2.0 *.upf -ssf demo.fsdb`

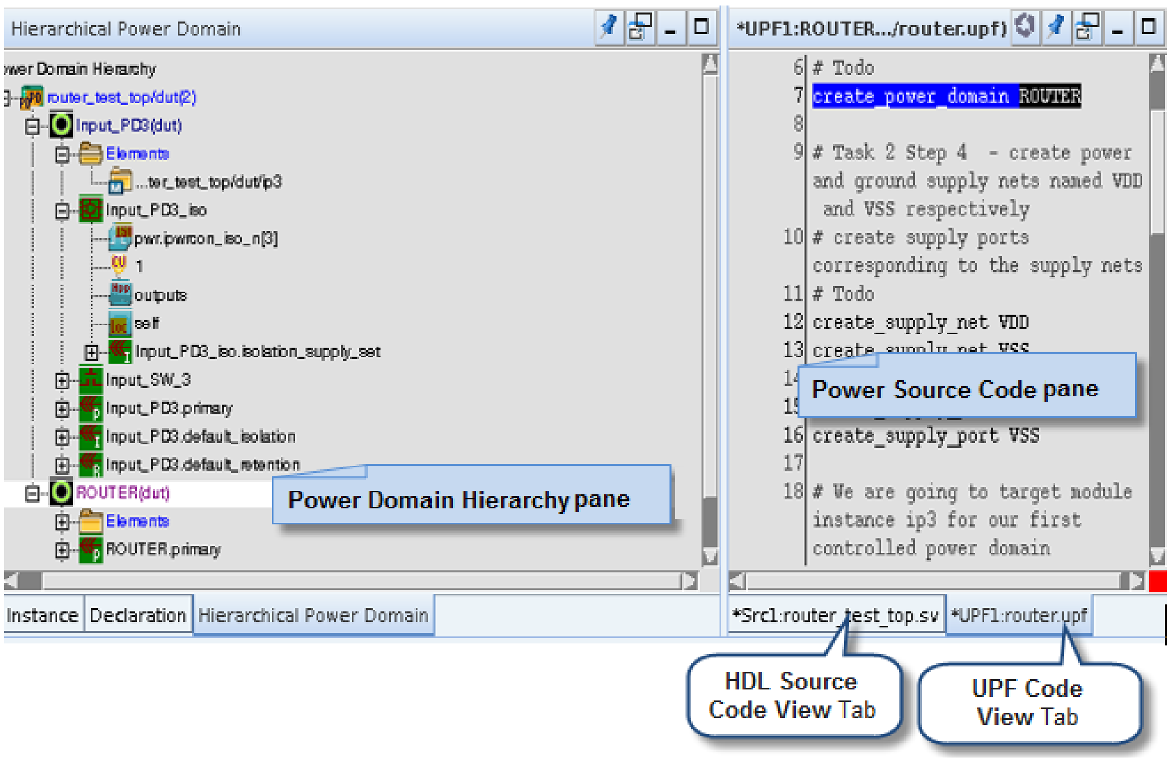
To dump the power-related signals that include the power supply nets and power-domain states (`upf_simstate`), you must use the `+power` or `+all` in the `$fsdbDumpvars` system task as follows:

```
$fsdbDumpvars("+power");
```

3 Low Power Debug Features

The Power Aware Debug support is integrated with the main **nTrace** window. The power related panes are opened in the main window after importing the UPF power file into the Verdi platform. Also, you can select the **Power Debug Mode** work mode from the **Welcome** page or select the **Window** > **Power Debug Mode** command to open the power related panes.

The following figure illustrates an example of Power Aware Debug panes:



```
# Todo
create_power_domain ROUTER

# Task 2 Step 4  - create power
and ground supply nets named VDD
 and VSS respectively
# create supply ports
corresponding to the supply nets
# Todo
create_supply_net VDD
create_supply_net VSS


create_supply_port VSS

# We are going to target module
instance ip3 for our first
controlled power domain
```

The following sections describe the Power Aware Debug panes:

- Power Domain Hierarchy Pane
- Power Source Code Pane
- Power State Table Pane
- Power State Annotation in nSchema
- Power Map Pane
- Temporal Flow View Pane
- Viewing Power Objects in nWave

3.1.1 Power Domain Hierarchy Pane

The **Power Domain Hierarchy** pane displays all the power domains defined in the design. This pane is docked to the same pane location as the design browser as a new tab. Each domain is marked with a different color.

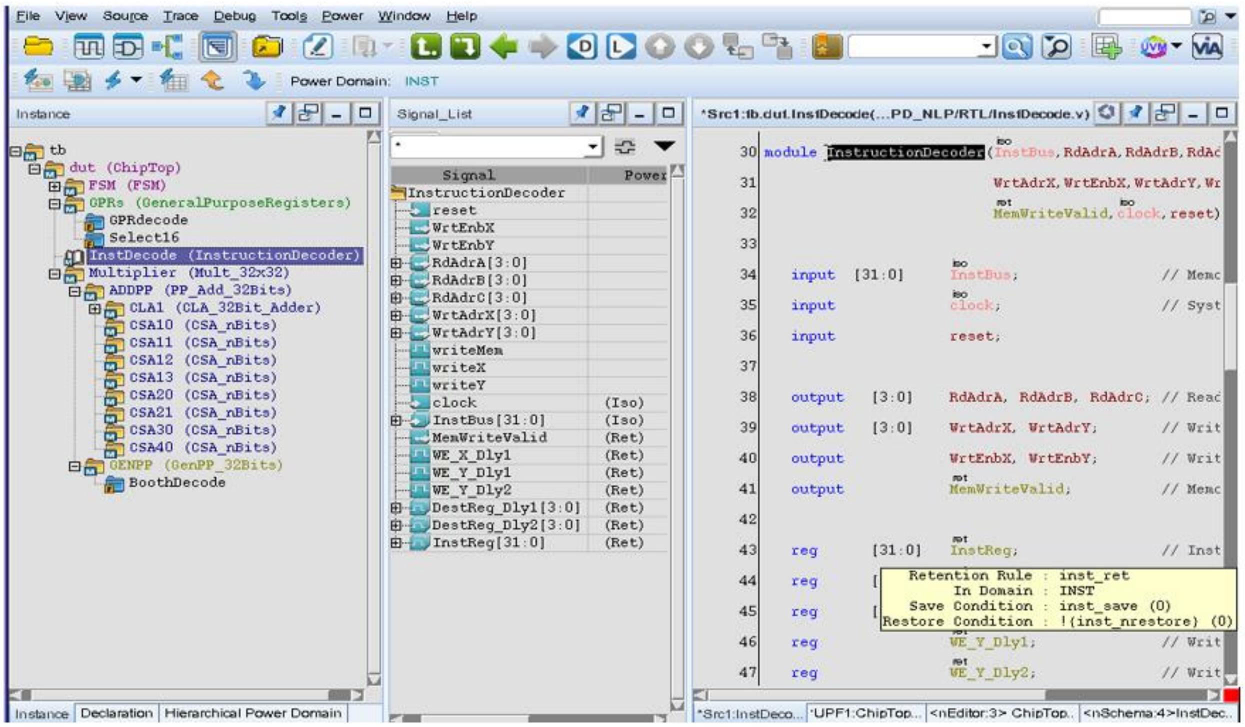
The power states for all power domains are represented by different icons. For example, the 🟩 icon represents the Normal Power Domain On and the 🟥 icon represents the Normal Power Domain Off. Use the **Help** > **Legend** menu command and select the **Power Manager** tab to check all the icons.

3.1.2 Power Source Code Pane

The **Power Source Code** pane consists of **HDL Source Code View** tab that displays the HDL source code and **UPF Code View** tab that displays the UPF code. The **UPF Code View** tab is displayed by default. Click the **HDL Source Code View** tab to view the HDL source code.

You can double-click an element in the **Power Domain Hierarchy** pane to view its definition in both UPF Code View and HDL Source Code View of the **Power Source Code** pane.

The **HDL Source Code View** annotates power information from UPF. The instance hierarchy list is color coded and each color represents a power domain. The signal list annotates the type of protection cells by which each signal is impacted. Moving the mouse on a signal opens a tooltip box with information on its policy.


File View Source Trace Debug Tools Power Window Help
Power Domain: INST
Instance
Signal_List
*Src1:tb.dut.InstDecode(...PD_NLP/RTL/InstDecode.v)
Instance | Declaration | Hierarchical Power Domain
*Src1:InstDeco... | 'UPF1:ChipTop... | <nEditor:3> ChipTop.. | <nSchema:4>InstDec..

```
tb
  dut (ChipTop)
    FSM (FSM)
    GPRs (GeneralPurposeRegisters)
      GPRdecode
      Select16
    InstDecode (InstructionDecoder)
    Multiplier (Mult_32x32)
      ADDPP (PP_Add_32Bits)
        CLA1 (CLA_32Bit_Adder)
        CSA10 (CSA_nBits)
        CSA11 (CSA_nBits)
        CSA12 (CSA_nBits)
        CSA13 (CSA_nBits)
        CSA20 (CSA_nBits)
        CSA21 (CSA_nBits)
        CSA30 (CSA_nBits)
        CSA40 (CSA_nBits)
      GENPP (GenPP_32Bits)
        BoothDecode
```

Signal	Power
InstructionDecoder	
reset	
WrtEnbX	
WrtEnbY	
RdAdrA[3:0]	
RdAdrB[3:0]	
RdAdrC[3:0]	
WrtAdrX[3:0]	
WrtAdrY[3:0]	
writeMem	
writeX	
writeY	
clock	(Iso)
InstBus[31:0]	(Iso)
MemWriteValid	(Ret)
WE_X_Dly1	(Ret)
WE_Y_Dly1	(Ret)
WE_Y_Dly2	(Ret)
DestReg_Dly1[3:0]	(Ret)
DestReg_Dly2[3:0]	(Ret)
InstReg[31:0]	(Ret)

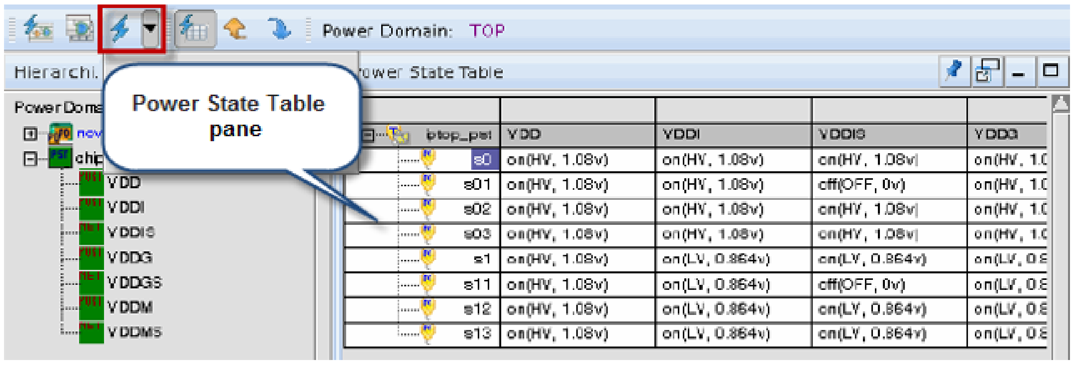
```
module InstructionDecoder(InstBus, RdAdrA, RdAdrB, RdAc
                         WrtAdrX, WrtEnbX, WrtAdrY, Wr
                         MemWriteValid, clock, reset)

   input  [31:0]     InstBus;                  // Memc
   input             clock;                    // Syst
   input             reset;

   output    [3:0]   RdAdrA, RdAdrB, RdAdrC;   // Reac
   output    [3:0]   WrtAdrX, WrtAdrY;         // Writ
   output            WrtEnbX, WrtEnbY;         // Writ
   output            MemWriteValid;            // Memc

   reg       [31:0]  InstReg;                  // Inst
   reg       [
   reg       [
   reg               WE_Y_Dly1;                // Writ
   reg               WE_Y_Dly2;                // Writ
```

```
 Retention Rule : inst_ret
      In Domain : INST
 Save Condition : inst_save (0)
Restore Condition : !(inst_nrestore) (0)
```

3.1.3 Power State Table Pane

The **Power State Table** pane shows the power state table of the design. The table is enabled by default if there is power state information in the UPF file.

If the **Power State Table** pane is not opened, invoke the following command to open it:
Power > **Show Power State Table** (if UPF file is loaded).

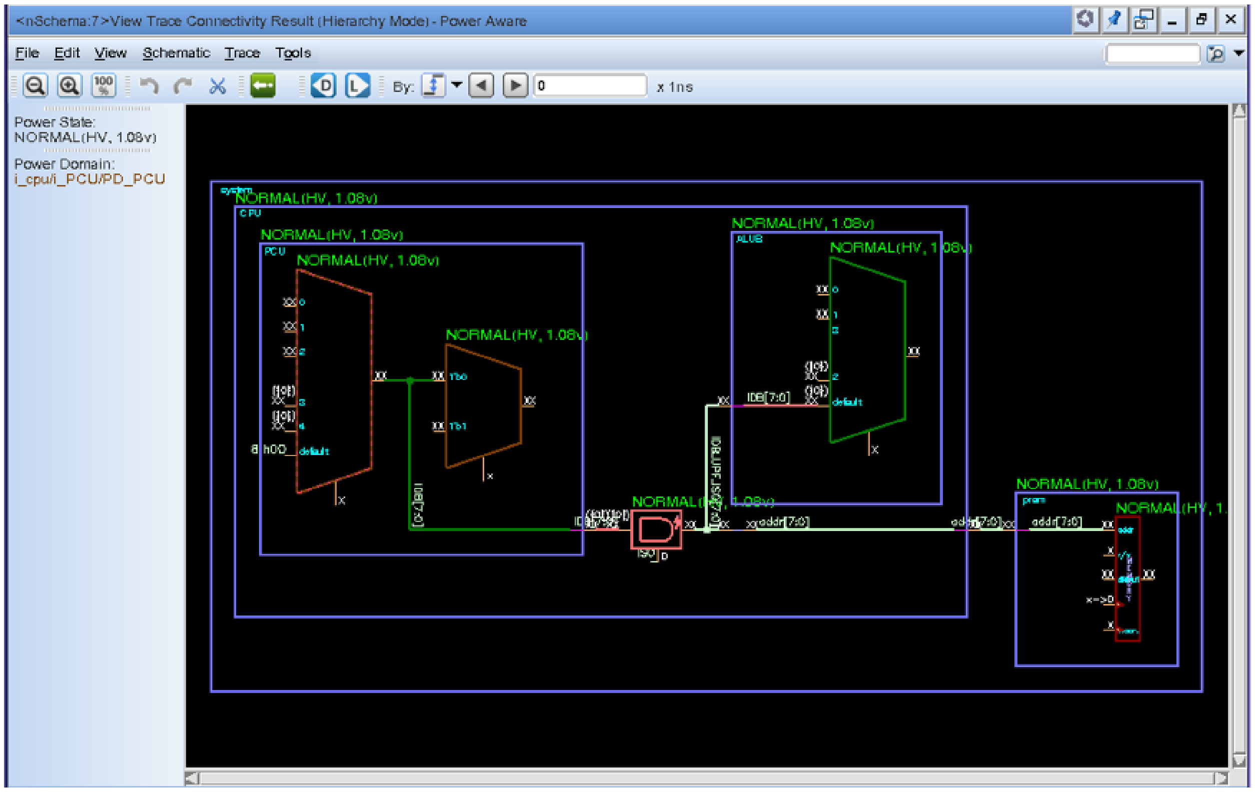


ptop_pst	VDD	VDDI	VDDIS	VDDG
s0	on(HV, 1.08v)	on(HV, 1.08v)	on(HV, 1.08v)	on(HV, 1.0
s01	on(HV, 1.08v)	on(HV, 1.08v)	off(OFF, 0v)	on(HV, 1.0
s02	on(HV, 1.08v)	on(HV, 1.08v)	on(HV, 1.08v)	on(HV, 1.0
s03	on(HV, 1.08v)	on(HV, 1.08v)	on(HV, 1.08v)	on(HV, 1.0
s1	on(HV, 1.08v)	on(LV, 0.864v)	on(LV, 0.864v)	on(LV, 0.8
s11	on(HV, 1.08v)	on(LV, 0.864v)	off(OFF, 0v)	on(LV, 0.8
s12	on(HV, 1.08v)	on(LV, 0.864v)	on(LV, 0.864v)	on(LV, 0.8
s13	on(HV, 1.08v)	on(LV, 0.864v)	on(LV, 0.864v)	on(LV, 0.8

3.1.4 Power State Annotation in nSchema

The **nSchema** pane supports highlighting power domains. The highlight can also be turned on/off with the **View** > **Highlight Power Aware Object** > **Power Domain Color and View** > **Highlight Power Aware Object > Power Aware Object Color** menu commands in the **nSchema** pane.

The power state is annotated on top of each instance after turning on the **Schematic** > **Active Annotation** menu command, and the instance with an off state is grayed out, as illustrated in the following figure:

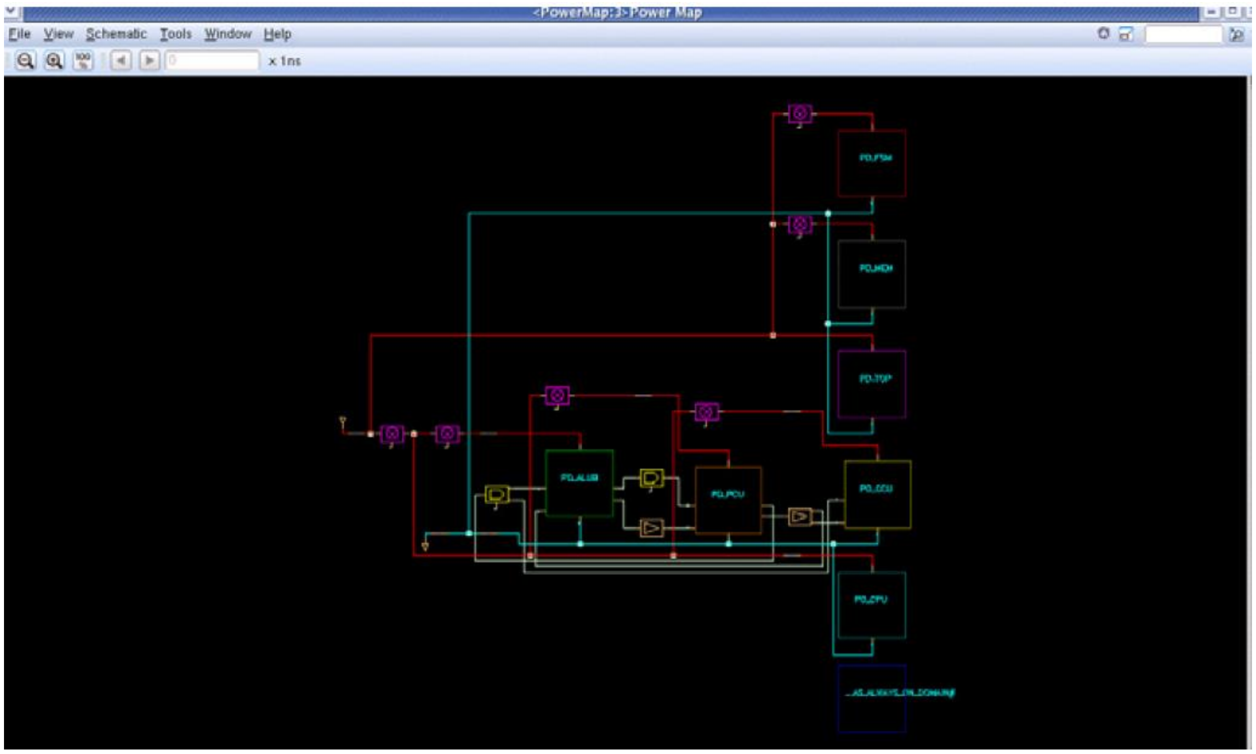

<nSchema:7>View Trace Connectivity Result (Hierarchy Mode) - Power Aware
File Edit View Schematic Trace Tools
By:
x 1ns
system
CPU
PCU
ALUB
NORMAL(HV, 1.08v)
default
IDB[7:0]
addr[7:0]

Power State:
NORMAL(HV, 1.08v)

Power Domain:
i_cpu/i_PCU/PD_PCU

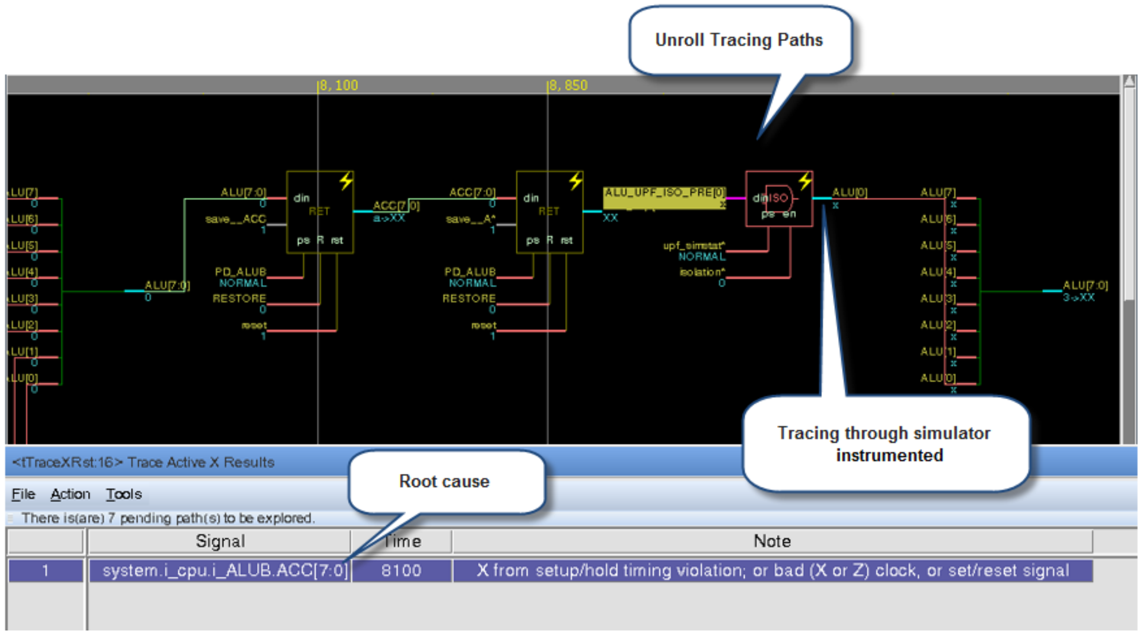
3.1.5 Power Map Pane

The **Power Map** pane provides the capability to visualize power intent in a flattened schematic view. The schematic view shows the structure of the power design (see the following figure), where the icon represents an isolation command, the icon represents a level-shifter command, and the icon represents a power switch cell. The **Power Map** pane provides support to highlight power domains.



3.1.6 Temporal Flow View Pane

The **Temporal Flow View** (TFV) pane supports power aware debugging. The Temporal Flow View contains the complete unrolled path of the low power hierarchy. The TFV window also supports power domain highlight and RET/ISO annotation for retention/isolation latches and flip-flops.



<tTraceXRst:16> Trace Active X Results

File Action Tools

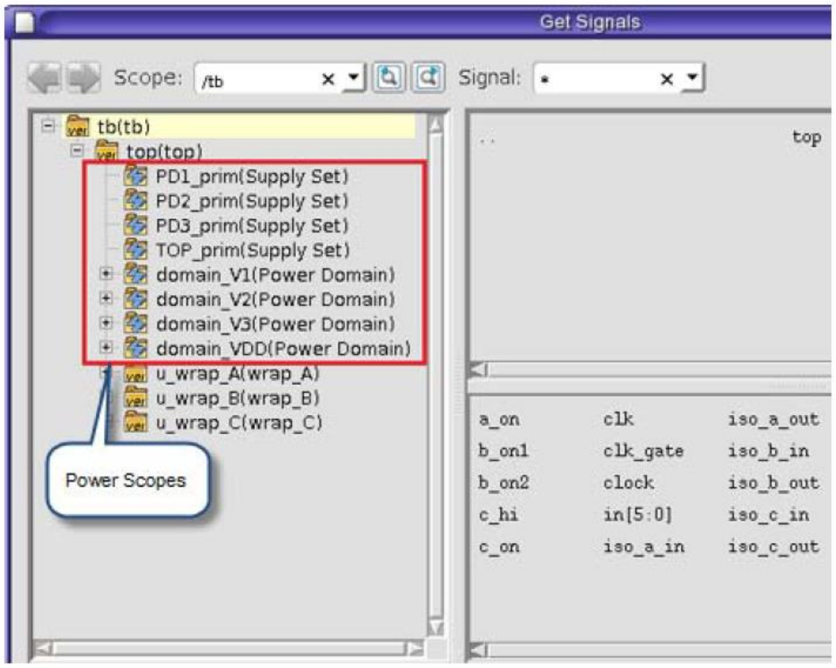
There is(are) 7 pending path(s) to be explored.

	Signal	Time	Note
1	system.i_cpu.i_ALUB.ACC[7:0]	8100	X from setup/hold timing violation; or bad (X or Z) clock, or set/reset signal

3.1.7 Viewing Power Objects in nWave

Verdi dumps power objects under the design scope. You can view the power objects in the nWave **Get Signals** form under the design scope. This feature is also supported with the standalone **nWave** window.

Power objects could be signal or power scope. Power scopes are represented with the icon, as shown in the following figure:

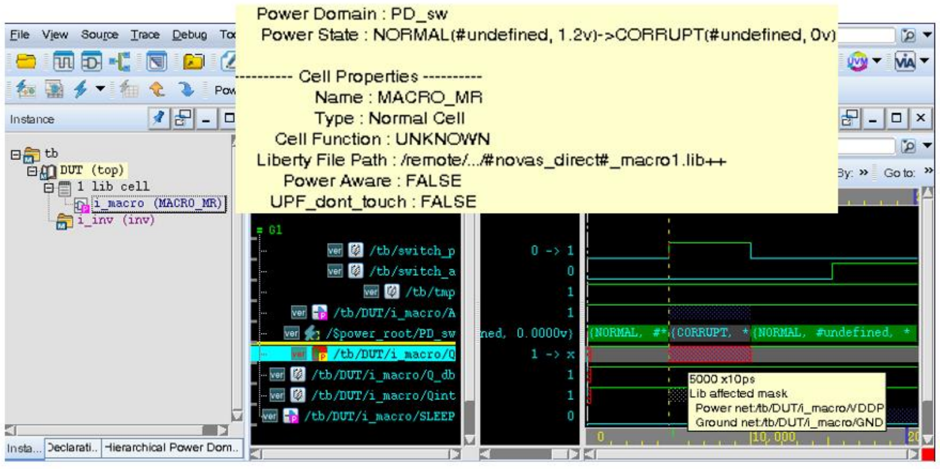


```
tb(tb)
  top(top)
    PD1_prim(Supply Set)
    PD2_prim(Supply Set)
    PD3_prim(Supply Set)
    TOP_prim(Supply Set)
    domain_V1(Power Domain)
    domain_V2(Power Domain)
    domain_V3(Power Domain)
    domain_VDD(Power Domain)
    u_wrap_A(wrap_A)
    u_wrap_B(wrap_B)
    u_wrap_C(wrap_C)
```

```
..                          top

a_on      clk        iso_a_out
b_on1     clk_gate   iso_b_in
b_on2     clock      iso_b_out
c_hi      in[5:0]    iso_c_in
c_on      iso_a_in   iso_c_out
```

3.2 Signal Shading

The following figure shows the shading of the DB/LIB affected signals:



Power Domain : PD_sw
Power State : NORMAL(#undefined, 1.2v)->CORRUPT(#undefined, 0v)
---------- Cell Properties ----------
Name : MACRO_MR
Type : Normal Cell
Cell Function : UNKNOWN
Liberty File Path : /remote/.../#novas_direct#_macro1.lib++
Power Aware : FALSE
UPF_dont_touch : FALSE

5000 x10ps
Lib affected mask
Power net:/tb/DUT/i_macro/VDDP
Ground net:/tb/DUT/i_macro/GND

3.3 List of Default Static Debug Reports Dumped by VCS NLP

The following reports can be used for debug purpose. These reports are dumped under the `mvsim_native_reports` directory.

Report Name	Description
`hierarchical_flop_inference.rpt`	Compile-time static report for Verilog. Captures the details of the inferred retention registers.
`hierarchical_flop_inference_vhdl.rpt`	Compile-time static report for VHDL. Captures the details of the inferred retention registers.
`isolation_association_summary.rpt`	Captures the details of the associated isolation cell instances and their associated isolation strategies
`library_mapping.rpt`	Captures the cell information.
`library_match.rpt`	Captures the matching/non-matching cell information.
`library_mapping_after_upf.rpt`	Captures the port-based corruption information of a cell.
`SDA/SDA_attribute_summary.rpt`	Captures information of the `set_design_attributes` command specified at element or model level.
`srsn_spa_association.rpt`	Captures information about the SRSN and SPA buffers associated with a port
`upf_state_machine.rpt`	Captures the objects with legal/illegal states/transitions being covered.
`upf_control_port_coverage.rpt`	Captures the functional coverage on the control ports/signals of a power switch, isolation strategy, and retention strategy.
`upf_powerswitch_mapping.rpt`	Captures the associated/Unassociated Power Switch Instances information
`upf_supply_set_power_state.rpt`	Captures information on all the power states of all the Supply Sets in the design.
`undefined_db_cell.rpt`	Lists the cells modules which are defined within `` `celldefine `` and `` `endcelldefine ``, but not having a corresponding matched liberty db cell.

Report Name	Description
`upf_supply_source_rsd.rpt`	Captures information on all the type of supply sources like root supply of a supply source and net/ports connected the source.
`upf_construct_statistics.rpt`	Captures information on the number of low power constructs used in the UPF file.

3.4 Verdi Database

Verdi platform provides the following two databases. All analysis engines and visualization tools use these databases.

- **Knowledge Database (KDB)**: As it compiles the design, the Verdi platform uses its internal synthesis technology to recognize and extract specific structural, logical, and functional information about the design and stores the resulting detailed design information in the KDB.

- **Fast Signal Database (FSDB)**: The FSDB stores the simulation results, including transaction data and logged messages from SVTB or other applicable languages in an efficient and compact format that allows you to access data quickly. Synopsys provides the object files that can be linked to common simulators to store the simulation results in FSDB format directly. You can generate FSDB either from the provided routines or after reading and converting your VCD file. In addition, FSDB read/write API routines are provided for customers and partners to use.

Copyright Notice and Proprietary Information

Destination Control Statement

Disclaimer

Trademarks

Third-Party Links

Free and Open-Source Software Licensing Notices

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com