Formality: Recognizing and Addressing Verification Problems in the UPF Flow in 2011.09

Bob Hatt

Formality CAE

September 2011

Objective

To give you skills to understand and debug low power UPF verifications in Formality.

Agenda

- Part 1: Low Power Static Verification and Library Requirements
- Part 2: Verification and Debugging in Formality
- Part 3: Resolving Common Issues

Agenda Part 1

- Part 1: Low Power Static Verification and Library Requirements
  - Formality and MVRC
  - Library cell requirements
  - Formality Library Checker
  - Consequences of Incomplete Models
- Part 2: Verification and Debugging in Formality
- Part 3: Resolving Common Issues

Formality and MVRC

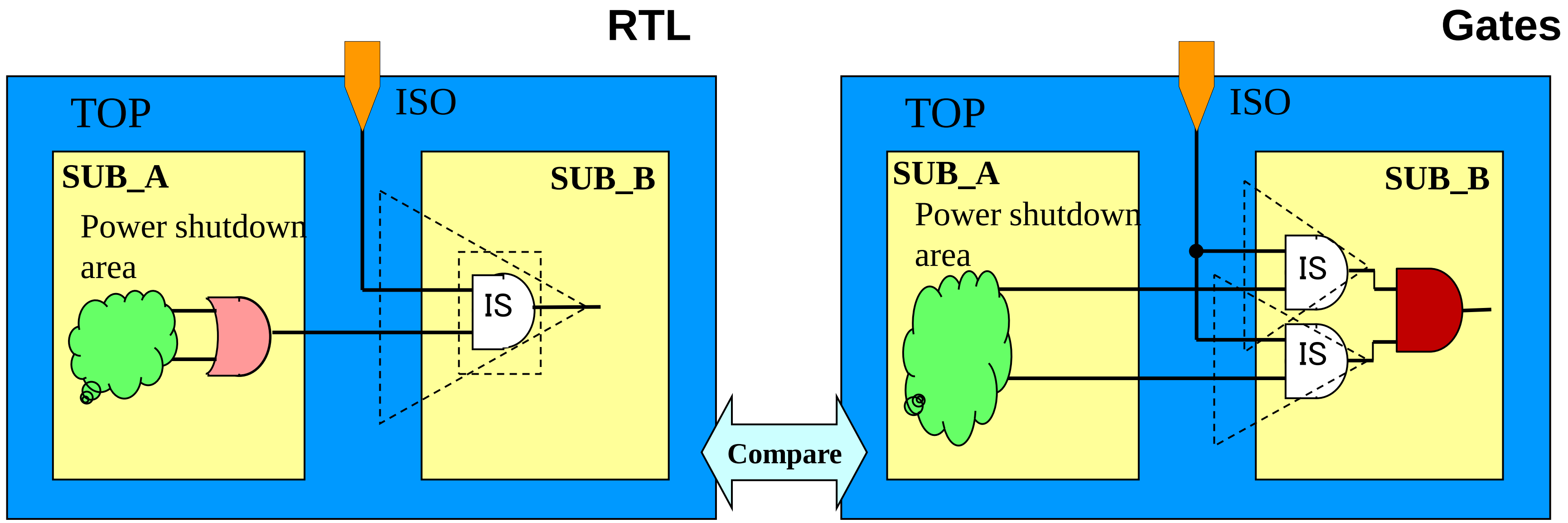
Complementary Tools for Low Power Checking

- Formality checks for functional equivalence between two designs
  - Example: Functional comparison of post layout PG netlist to the original low power design specification (RTL+UPF)
- MVRC checks for adherence to MV rules and power intent specification for a single design
  - MV Rules Examples: level shifter or isolation cell missing, level shifter / isolation cell interchange, illegal routing/placement, polarity of isolation signal
  - Power Intent Examples: check power up/down sequence, Validate control signal networks

Together, these tools provide the essential checks for MV and power gated designs

Formality Detects Functional Differences

- Both the RTL and Gates designs are properly isolated
- Formality will show a functional difference because the "OR" function in the RTL has changed to "AND" in the Gates

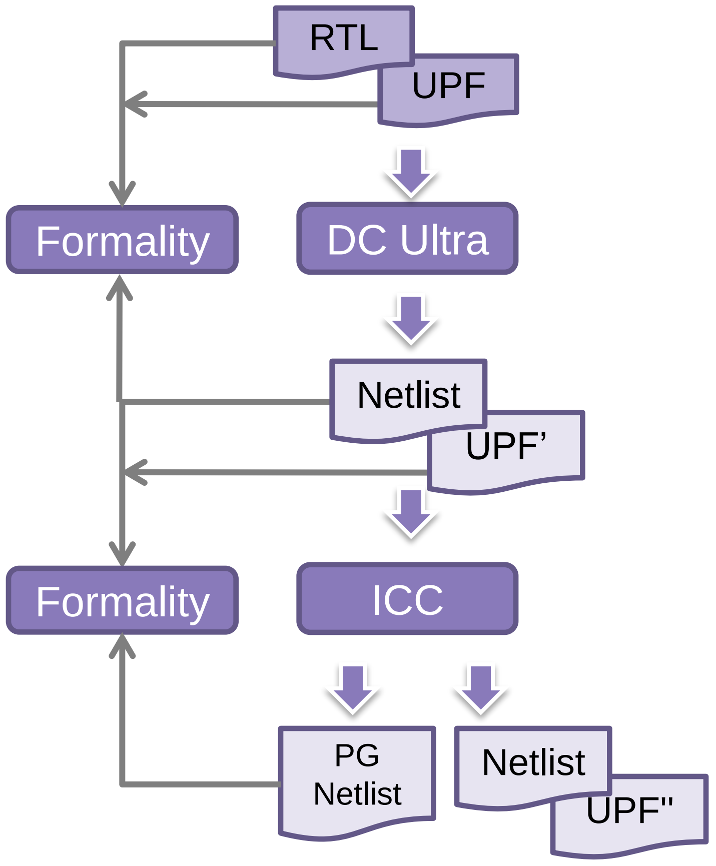


MVRC - Low Power Rule Checks

Power Consistency Check	• Validate power-intent to ensure it defines necessary policies across all power-modes
Library Check	• Ensure library contains cells necessary to implement the power-intent and the right cells get eventually inserted into the design
Architectural Check	• Ensures necessary and sufficient low-power structures are present in UPF+Design
Structural Check	• Ensures functionality of low-power structures in the design is consistent with power-intent (UPF) and cell library
Functional Check	• Ensures power domain partitions do not lead to functional errors in the design
PG Check	• Ensures power/ground pin connectivity in post-layout design is consistent with power-intent (UPF) and cell library
Transistor Check	• Ensures accurate structural checks based on transistor level schematic of cells

Formality in the Synopsys Low-Power Flow

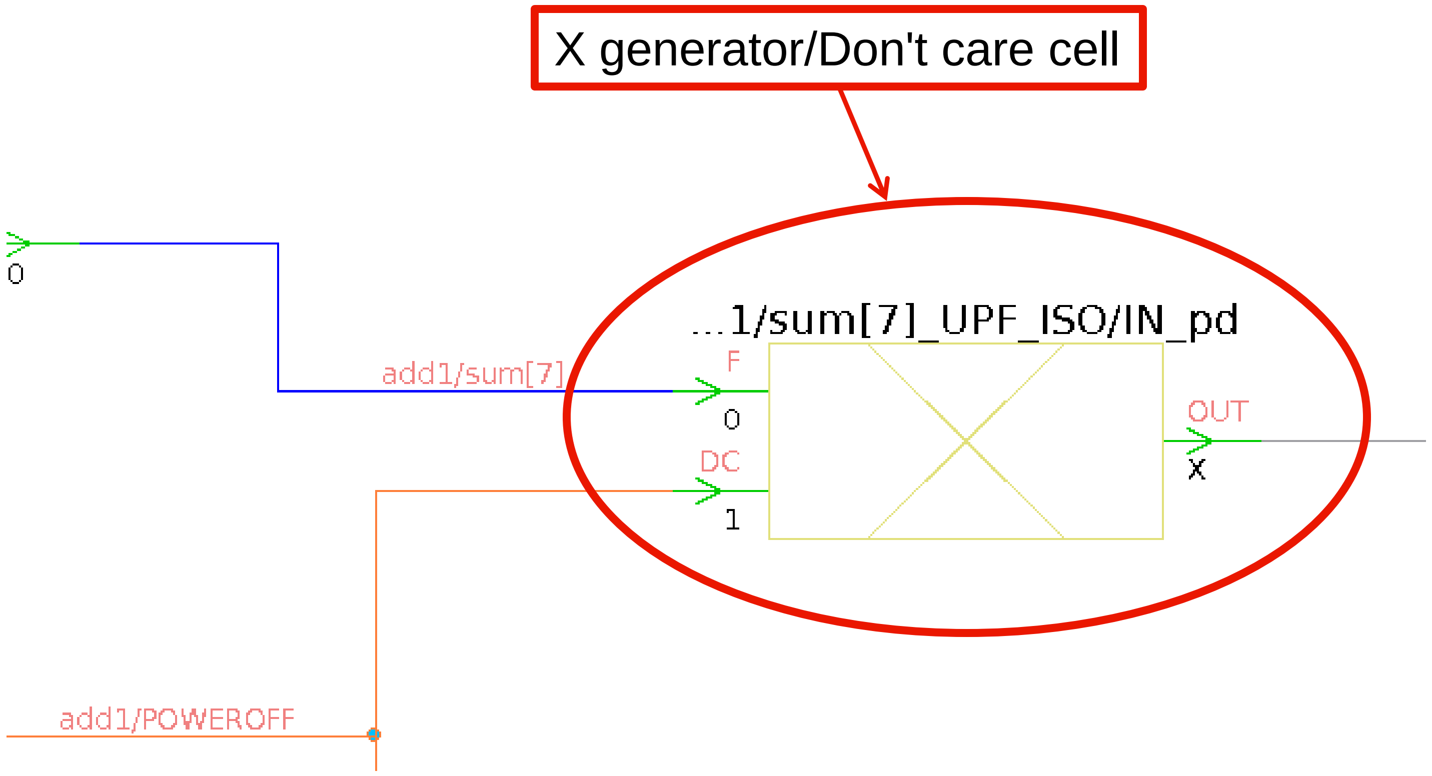
- RTL +UPF
    - Simulated with MVSIM
    - Checked with MVRC
- DC netlist can be Verilog or DDC
- UPF’ must be from DC `save_upf`
- ICC netlist can be pg_netlist (recommended) or netlist+UPF"
- Technology libraries
    - Library cells with power pins and power-down functions



UPF Effects on Verification

- Extends functional equivalency checking to include power effects on logic function
- Connects supply rails to power/ground pins
- Powers-down cell outputs and sequential elements
- Matches IEEE-1801 (UPF) simulation semantics
- Drives X into circuit using power-down functions triggered by power and ground pins

Power down inserts X into the design



Library Requirements

- Liberty attributes
  - related_power
  - related_ground
  - pg_pin
  - power_down_function
- Retention Register Models
- Liberty Syntax Example
- Verilog Syntax Example

Retention Register Modelling

- IEEE 1801 generic retention behavior modifies the clocked process and specifies a second process (balloon latch) to save/restore state.
- UPF should map to a retention cell model that has the same behavior (flop+balloon latch) that the RTL has.
- Analzye_points will detect retention models that are missing the balloon latch.

Liberty Syntax for PG Pins/PDF

PG Pin syntax:

```
pg_pin (VDD) {
   voltage_name : "V";
   pg_type : "primary_power";
}
pg_pin (VSS) {
   voltage_name : "VSS";
   pg_type : "primary_ground";
}
```

Power down function syntax for flip-flop/latch:

```
ff (IQ,IQN) {
   clocked_on : "CLK";
   next_state : "D";
   clear : "RSTB'";
   power_down_function : "!VDD + VSS";
}
```

Power down function syntax for an output pin:

```
pin (Z) {
   related_power_pin : "VDD";
   related_ground_pin : "VSS";
   power_down_function : "!VDD + VSS";
   function : "!(IN2*IN1)";
   …
```

Verilog Models with PG Pins

```
module AN2(A, B, Z, VDD, VSS);
input  A,B;
output Z;
(* pg_type = "primary_power" *) input VDD;
(* pg_type = "primary_ground" *) input VSS;

wire poweron = VDD & !VSS;
assign Z = poweron ? (A & B) : 1'bx;

endmodule
```

```
module HDRSWITCH(VDD, VSS, TVDD, EN, ENOUT);
input  EN;
output ENOUT;
(* pg_type = "primary_power" *) input VDD;
(* pg_type = "internal_power" *)output TVDD;
(* pg_type = "primary_ground" *) input VSS;

wire poweron = VDD & !VSS;

assign ENOUT = poweron ? (EN) : 1'b0;
assign TVDD = (poweron & EN) ? VDD : 1'b0;

endmodule
```

pg_pin attributes match the Liberty attributes

power down functionality

Formality Library Verification Mode for Power Aware Cells (F-2011.09)

```
module AN2(A, B, Z, VDD, VSS);
input  A,B;
output Z;
(* pg_type = "primary_power" *) input VDD;
(* pg_type = "primary_ground" *) input VSS;

wire poweron = VDD & !VSS;
assign Z = poweron ? (A & B) : 1'bx;

endmodule
```

EQUIVALENT?

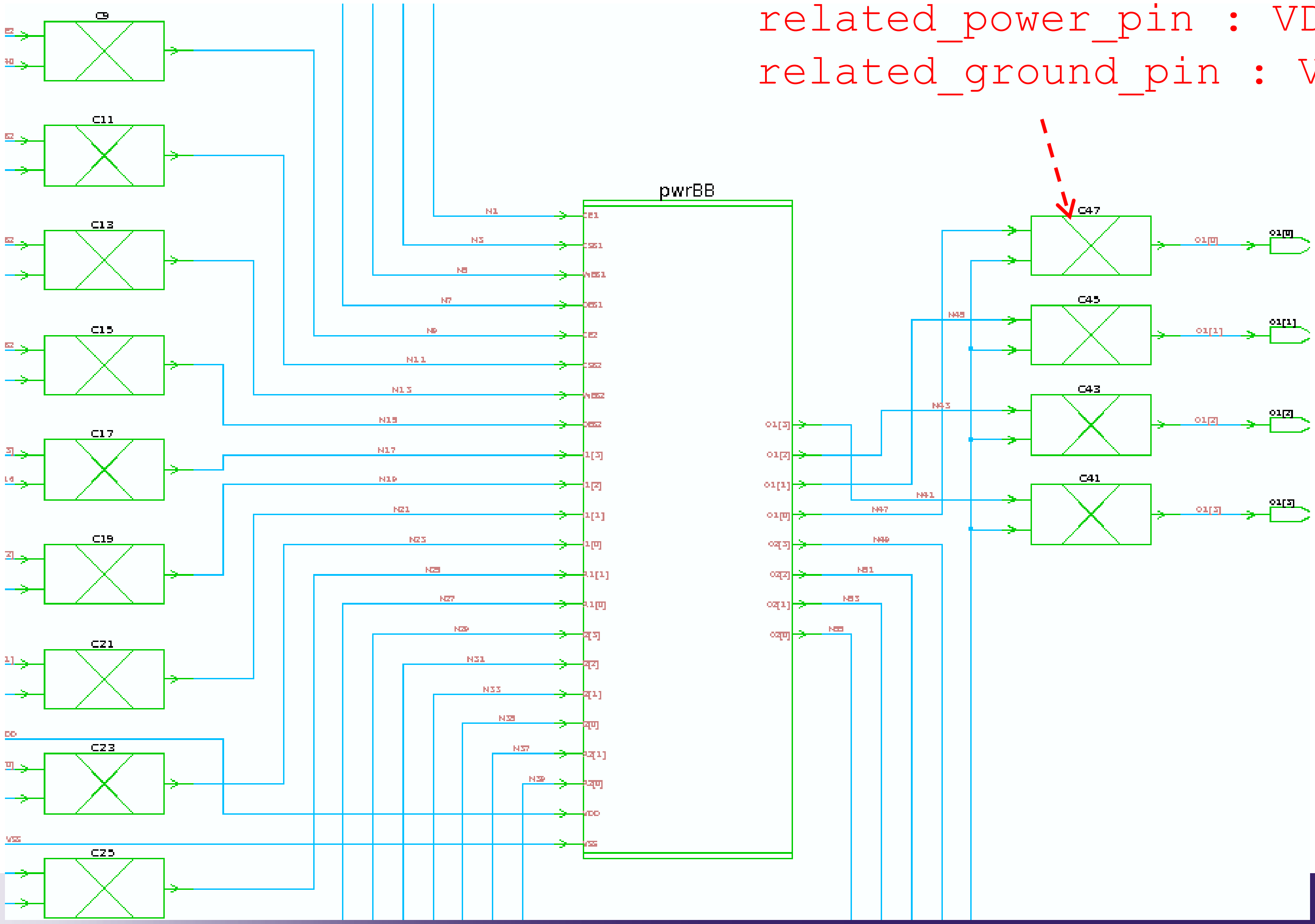
```
cell (AN2)
   pg_pin (VDD) {
   …
   pin (Z) {
   related_power_pin : "VDD";
   related_ground_pin : "VSS";
   power_down_function : "!VDD + VSS";
   function : " A * B ";
   …
```

Formality Library Verification Mode for Power Aware Cells

- Verify power aware Verilog models against .db versions
  - `library_verification VERILOG_PWRDB`
  - `library_verification PWRDB_VERILOG`
- Verify power aware .db models
  - `library_verification DB_DB`
    - Verifies both non power and power designs
    - Reports include unmatched (power) designs

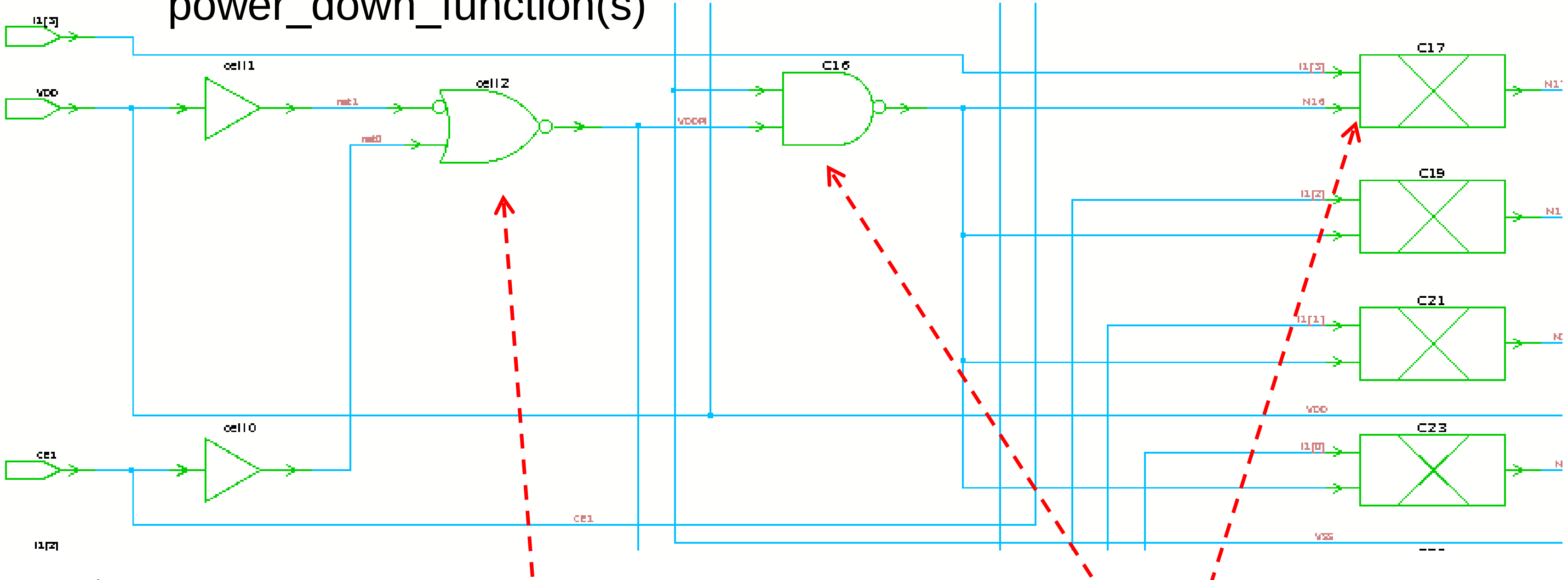
Macro (RAM) Cells

```
pin (O1[3:0]) {
    related_power_pin : VDDPI;
    related_ground_pin : VSS;
```



Macro Cells with Internal Switches

internal pg_pin, related_power_pin(s) ,
power_down_function(s)



```
pg_pin (VDDPI) {
     pg_type : internal_power;
     direction : internal;
     voltage_name     : VDDPI;
     switch_function : "CE1";
     pg_function     : "VDD" ;
}
```

```
pin (I1[3:0]) {
     related_power_pin : VDDPI;
     related_ground_pin : VSS;
      …
```

Low Power Library Checker

How to Spot an Incomplete Model

- Low power libraries may contain cells that have incorrect or incompletely modeled power behavior.
  - Unread power/ground pins
  - Missing power down functionality on flip-flops or output pins
  - Inconsistent pin directions
  - Missing or invalid switch function
  - Retention cells missing balloon latch

Low Power Library Checker

Library Checker Summary

- Example transcript from log file (after `set_top`)

```
************ Library Checking Summary ************
Warning:  60 unlinked power cell(s) with unread pg pins.
Warning:  111 unlinked power cell(s) with no power down
functions on outputs.
Warning:  60 unlinked power cell(s) marked as retention with
unread backup pg pins.
Warning:  408 unlinked power cell(s) with no power down
function
**************************************************
```

- For more details on each cell use
  - **`report_libraries -defects all`**
  - **`report_libraries -defects errors`**

Low Power Library Checker

Library Checker Summary

- Example transcript from log file (after `load_upf`)

```
************ Library Checking Summary ************
Warning:  50 unlinked power cell(s) with unread pg pins.
Warning:  10 linked power cell(s) with unread pg pins.
Warning:  105 unlinked power cell(s) with no power down
 functions on outputs.
Error:  6 linked power cell(s) with no power down functions
 on outputs.
Warning:  52 unlinked power cell(s) marked as retention
 with unread backup pg pins.
Error:  8 linked power cell(s) marked as retention with
 unread backup pg pins.
Warning:  389 unlinked power cell(s) with no power down
 function on an ff or latch.
Warning:  19 linked power cell(s) with no power down
 function on an ff or latch.
**************************************************
```

Low Power Library Checker

Detailed reports

```
################################################################
####    TECH LIB - r:/SAED90NM_MAX_HTH
################################################################
Library Cell /Defect List                        Attributes
-------------------------                        ----------
AOBUFX1#PWR                                        u
   port VDD : unread pg_pin
AOBUFX2#PWR                                        u
   port VDD : unread pg_pin
AOBUFX4#PWR                                        u
   port VDD : unread pg_pin
AODFFARX1#PWR                                      su
   port VDD : unread pg_pin
DFFARX1#PWR                                        su
   cell *dff.00* : no power_down_function : copied from Q/QN
DFFARX2#PWR                                        su
   cell *dff.00* : no power_down_function : copied from Q/QN
```

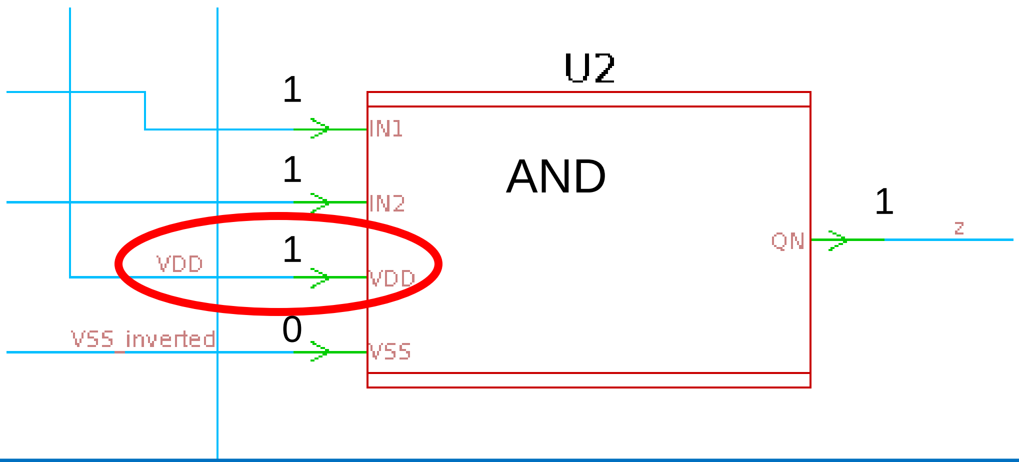
Low Power Library Checker

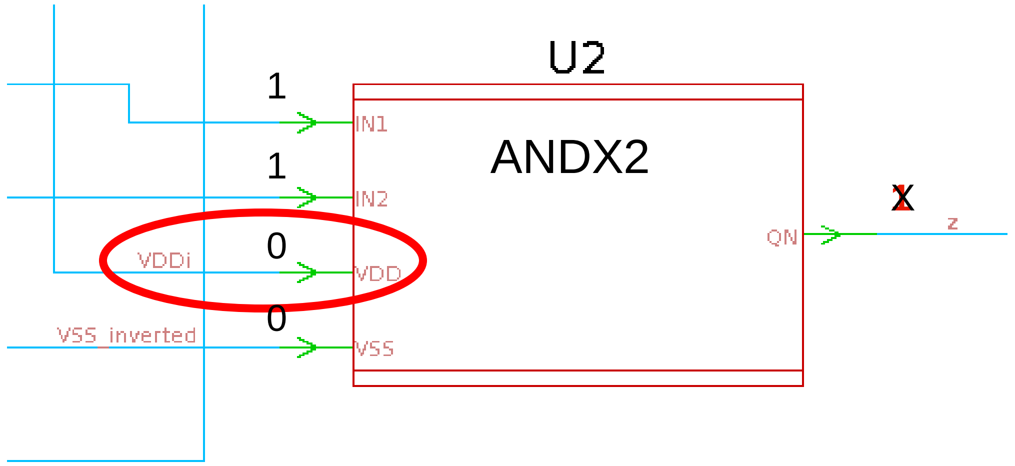
- Let FM attempt to fix common issues
  - `set hdlin_library_auto_correct true`
  - To allow progress while library model is being fixed.
- Make library errors cause set_top or load_upf to fail
  - `set hdlin_library_ignore_errors true`

Consequences of Incomplete Models

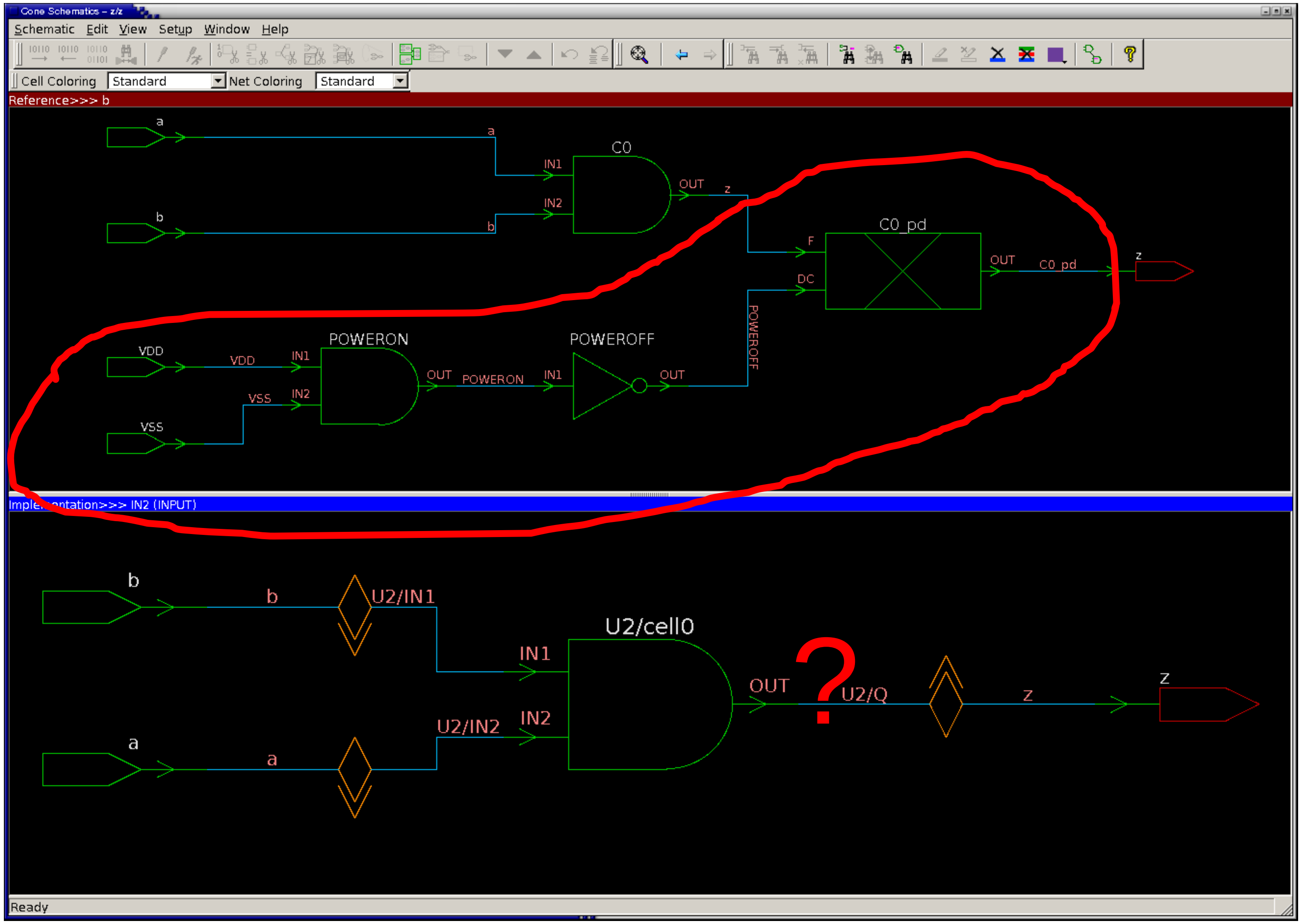
- Worst case
  - Verification succeeds when it should not
- Best Case
  - Verification succeeds because the cell is connected to the correct supplies and never turns off incorrectly.
  - Verification fails
    - Hard to debug missing power down functions
- Pay attention to library checker Error and Warning messages.

Verification SUCCEEDED because of a missing a power_down_function





Logic cones inside of technology cells



Predictable Success

Summary

Low Power Static Verification and Library Requirements

- Run MVRC to check the quality of the UPF
- Simulate the design (MVSIM)
- Fix Libraries
  - pg_pin, power_down_function etc.
- Low Power Library Checks
    - `report_libraries -defects <all | error>`
  - optionally autocorrect library cells
    - `set hdlin_library_auto_correct true`

SYNOPSYS®

Predictable Success

Agenda

- Part 1: Low Power Static Verification and Library Requirements
- Part 2: Verification and Debugging in Formality
- Part 3: Resolving Common Issues

SYNOPSYS® Predictable Success

UPF related issues in Verification

- UPF Effects on Verification
  - New variable settings in 2011.09
- Different Flavors of UPF
- Failure Symptoms
- Debugging techniques

UPF Effects on Verification

- Extends functional equivalency checking to include power effects on logic function
- Matches IEEE-1801 (UPF) simulation semantics
  - Drives logic X into circuit when a domain/cell is powered off.
- Connects supply nets to power/ground pins
- Powers down cell outputs and sequential elements using power down functions
  - triggered by power and ground nets

All Supplies are Forced on by Default

- In Formality 2011.09 the variable `verification_force_upf_supplies_on` defaults to value `true`
  - In previous versions default value was "false"
- Verification runs faster out of the box
- Allows detection of non-power related flow/optimization issues more quickly
- This is only a partial verification result! Set this variable to `false` to get a complete verification of all power states

Formality Inserts Cutpoints at the Power Domain Boundary

- New variable in 2011.09
  - `verification_insert_upf_isolation_cutpoints true`
- Improves verification performance by reducing verification complexity
- Prevents X from escaping a power off domain into a power on domain
  - Reduces failures due to X propagation differences

Look at the Verification Results ATTENTION Messages

- **verification_insert_upf_isolation_cutpoints true**

```
    ********************** Verification Results ***********************
    Verification FAILED
       ATTENTION: 8 failing PDCut compare point(s)
                  represent UPF power domain boundary differences.
                  UPF power domain boundary verification is enabled by the variable
                  verification_insert_upf_isolation_cutpoints.
    ******************************************************************************

------------------------------------------------------------------------------------------------
Matched Compare Points       BBPin     Loop    BBNet      Cut     Port    PDCut      DFF      LAT    TOTAL
------------------------------------------------------------------------------------------------
Passing (equivalent)             0        0        0        0       32        0       33        0       65
Failing (not equivalent)         0        0        0        0        0        8        5        0       13
```

- **verification_force_upf_supplies_on true**

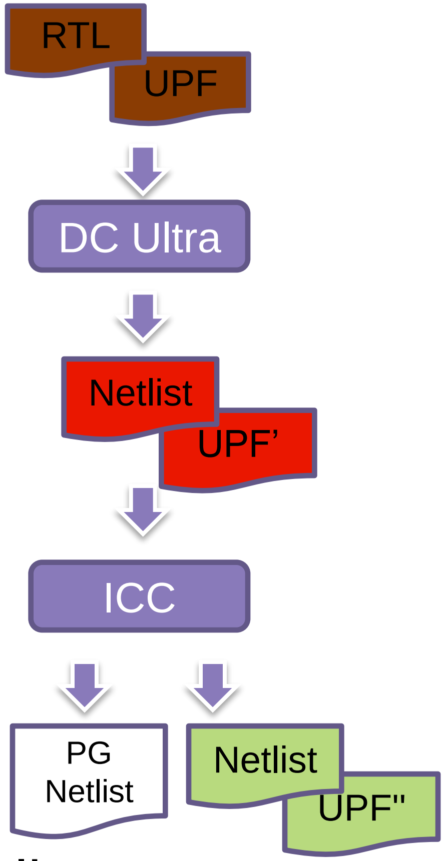
```
    ********************** Verification Results ***********************
    Verification SUCCEEDED
       ATTENTION:  Verification was run with all UPF supplies constrained
                   to their ON state.
                   This is only a partial verification result as it does
                   not cover all operational states.
```

Formality Supports the Synopsys Low Power Design Flow

- Formality supports the UPF commands allowed in the Synopsys Low Power UPF flow.
- New for 2011.09
  - set_isolation –location fanout
  - query*/find_objects commands
  - create_logic_net/create_logic_port, connect_logic_net

Neapolitan UPF
Chocolate, Strawberry, Pistachio

- UPF (Chocolate)
  - Implement all UPF constructs (like simulation)
- UPF' (Strawberry)
  - #Create by Design Compiler …
  - Implement supplies and switches
  - Do not implement isolation/retention
- UPF" (Pistachio)
  - #Created by IC Compiler ...
  - Implement supplies
  - Used by PrimeTime and other tools.
- PG Netlist (Vanilla)
  - Nothing to implement, it is all explicit in the netlist



Messages from load_upf

- RTL+UPF

```
Loading upf file 'test.upf'
Info:  load_upf: /u/…/training2011/example3/test.upf: implementing all UPF
constructs.
Info:  load_upf: implemented 0 retention registers and 0 isolation cells.
Info:  load_upf: completed.
```

- UPF' (#Generated by Design Compiler …)

```
Info:  load_upf: /u/…/training2011/example3/test.mapped.upf  Generated by
Design Compiler(E-2010.12) on Tue Jan 18 09:52:21 2011, implementing
supply network and connecting retention and isolation supplies.
```

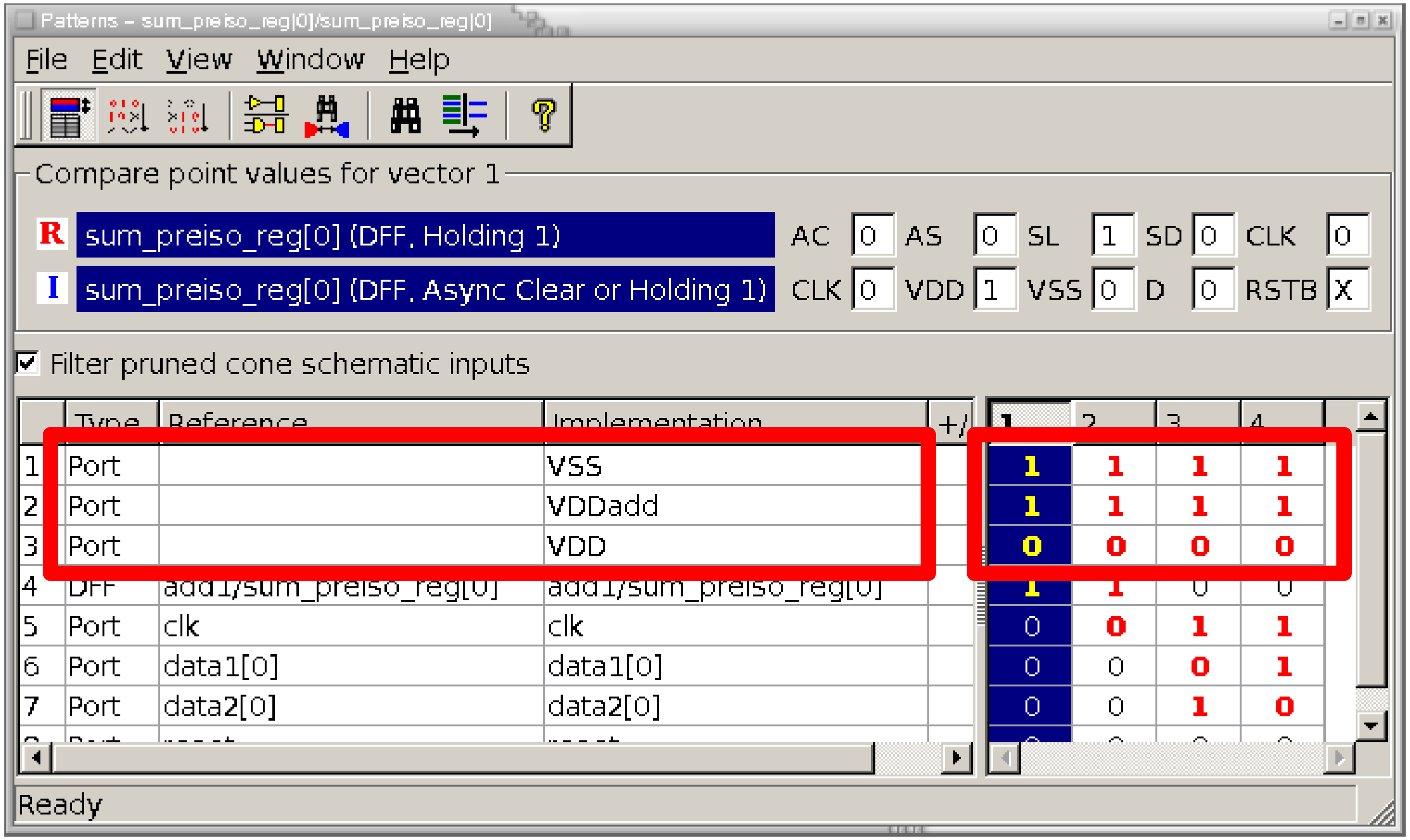
- UPF'' (#Generated by IC Compiler …)

```
Info:  load_upf: /u/…/training2011/example3/test.icc.upf: Generated by IC
Compiler(E-2010.12) on Tue Jan  4 21:23:10 2011, connecting primary
supplies.
```

Failure Symptoms

- Failing patterns show X differences at compare point
  - CutPin, Primary output, Black box inputs, Registers
  - Something in the netlist is powered off causing X to propagate
- Failing patterns show 1/0 differences at the compare point
  - maybe not a power-off related problem.

Pattern Window
UPF cones are unusual



Patterns – sum_preiso_reg[0]/sum_preiso_reg[0]

File Edit View Window Help

Compare point values for vector 1

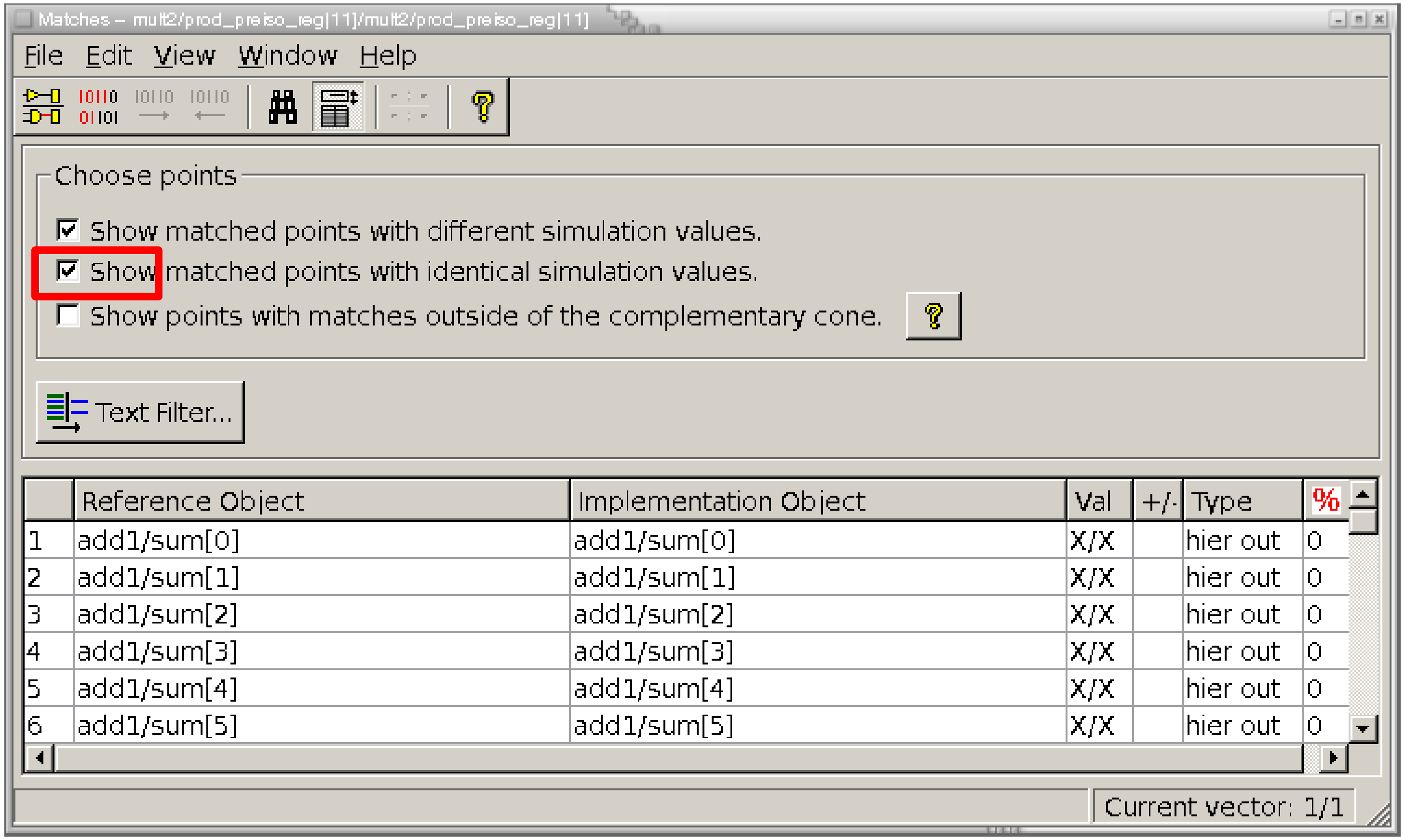
R	sum_preiso_reg[0] (DFF, Holding 1)	AC	0	AS	0	SL	1	SD	0	CLK	0
I	sum_preiso_reg[0] (DFF, Async Clear or Holding 1)	CLK	0	VDD	1	VSS	0	D	0	RSTB	X

☑ Filter pruned cone schematic inputs

	Type	Reference	Implementation	1	2	3	4
1	Port		VSS	1	1	1	1
2	Port		VDDadd	1	1	1	1
3	Port		VDD	0	0	0	0
4	DFF	add1/sum_preiso_reg[0]	add1/sum_preiso_reg[0]	1	1	0	0
5	Port	clk	clk	0	0	1	1
6	Port	data1[0]	data1[0]	0	0	0	1
7	Port	data2[0]	data2[0]	0	0	1	0

Ready

Matched point window can also quickly pinpoint X on hierarchical pins


Matches – mult2/prod_preiso_reg[11]/mult2/prod_preiso_reg[11]

File Edit View Window Help

Choose points

- [✓] Show matched points with different simulation values.
- [✓] Show matched points with identical simulation values.
- [ ] Show points with matches outside of the complementary cone.

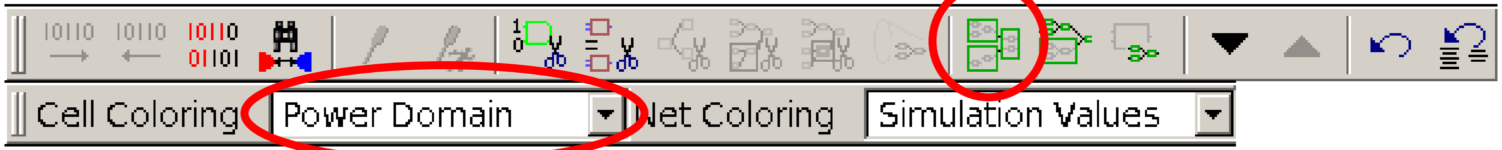
Text Filter...

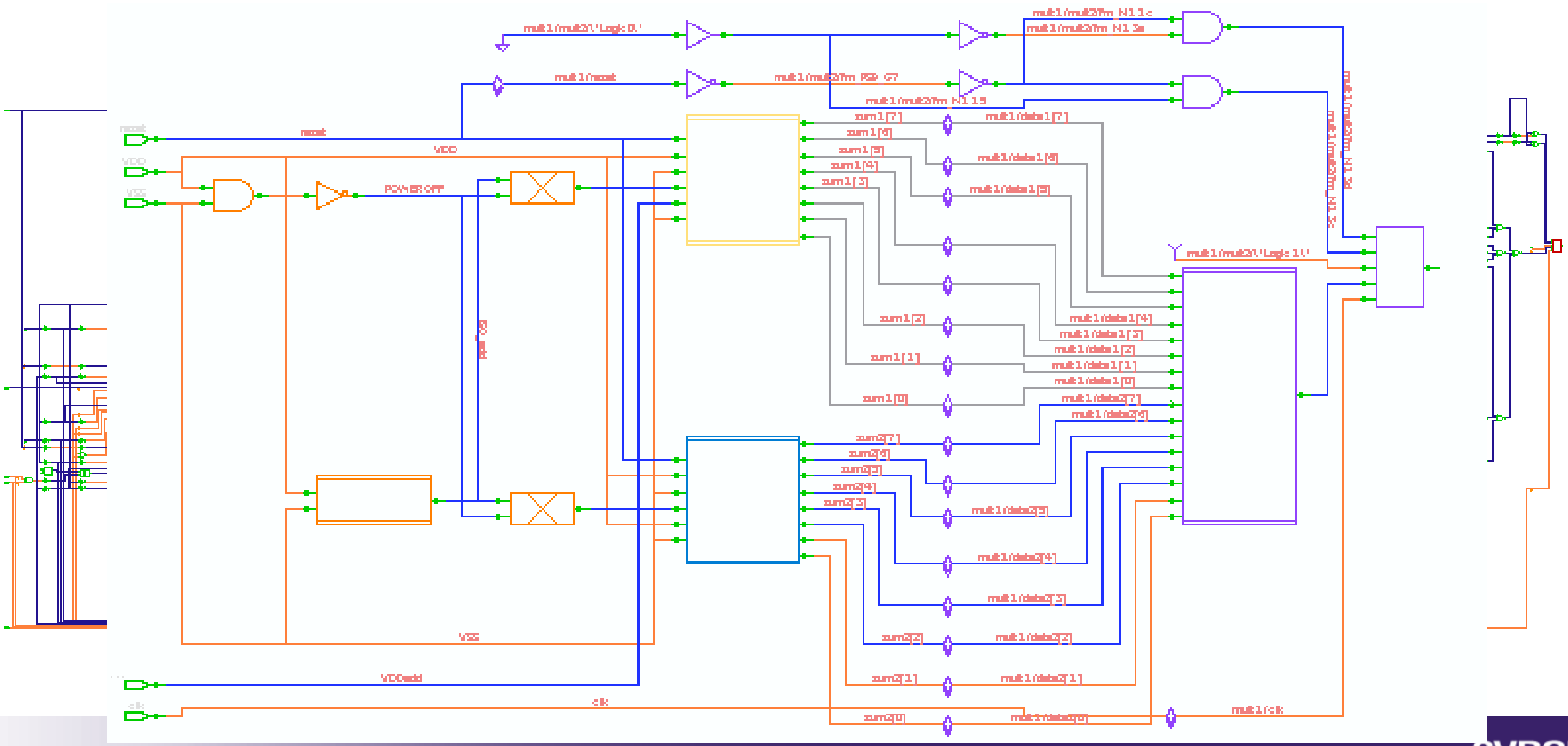
	Reference Object	Implementation Object	Val	+/-	Type	%
1	add1/sum[0]	add1/sum[0]	X/X		hier out	0
2	add1/sum[1]	add1/sum[1]	X/X		hier out	0
3	add1/sum[2]	add1/sum[2]	X/X		hier out	0
4	add1/sum[3]	add1/sum[3]	X/X		hier out	0
5	add1/sum[4]	add1/sum[4]	X/X		hier out	0
6	add1/sum[5]	add1/sum[5]	X/X		hier out	0

Current vector: 1/1

Best UPF Cone Schematic View

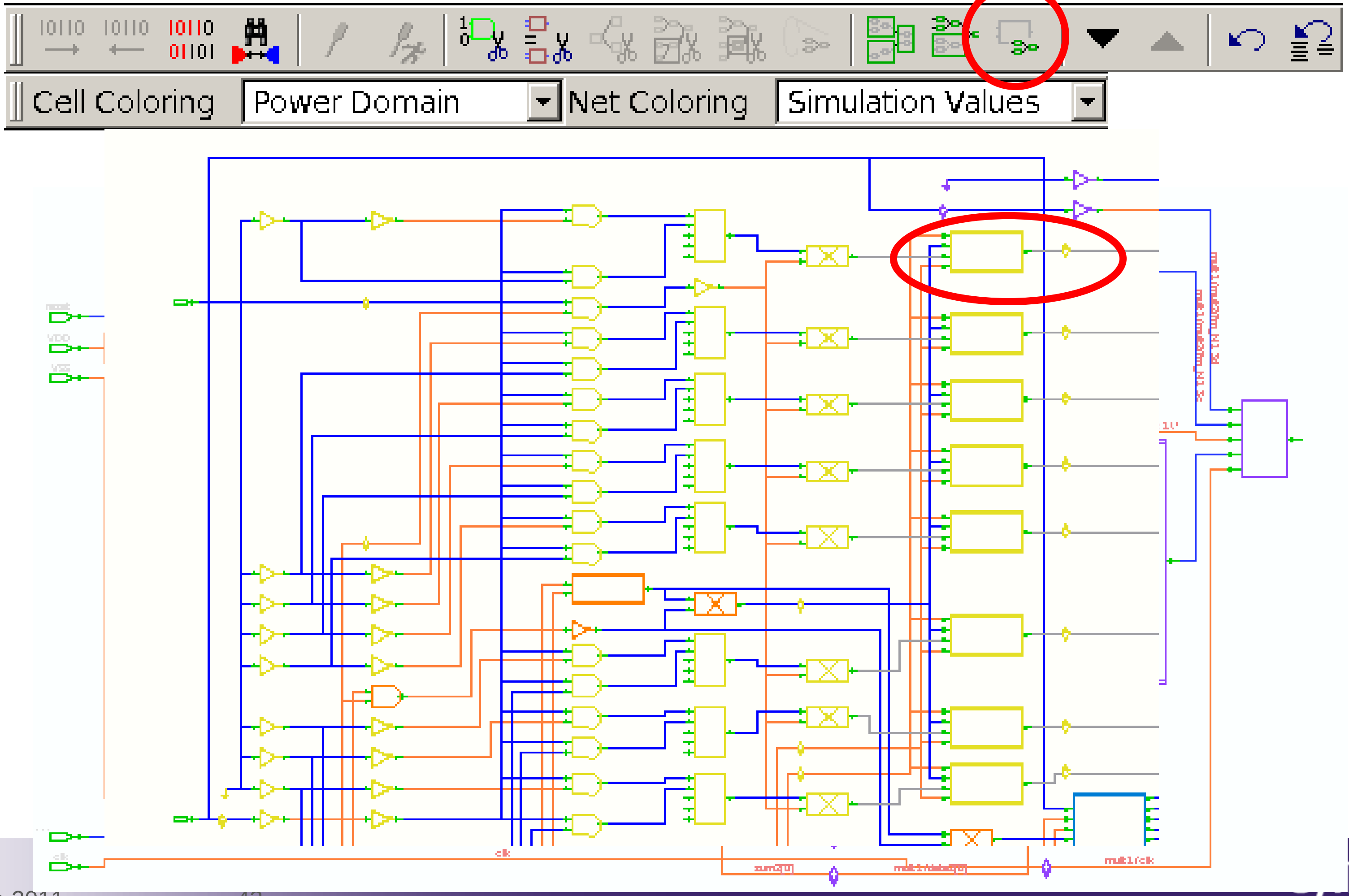
Group All by Parent


Cell Coloring: Power Domain
Net Coloring: Simulation Values

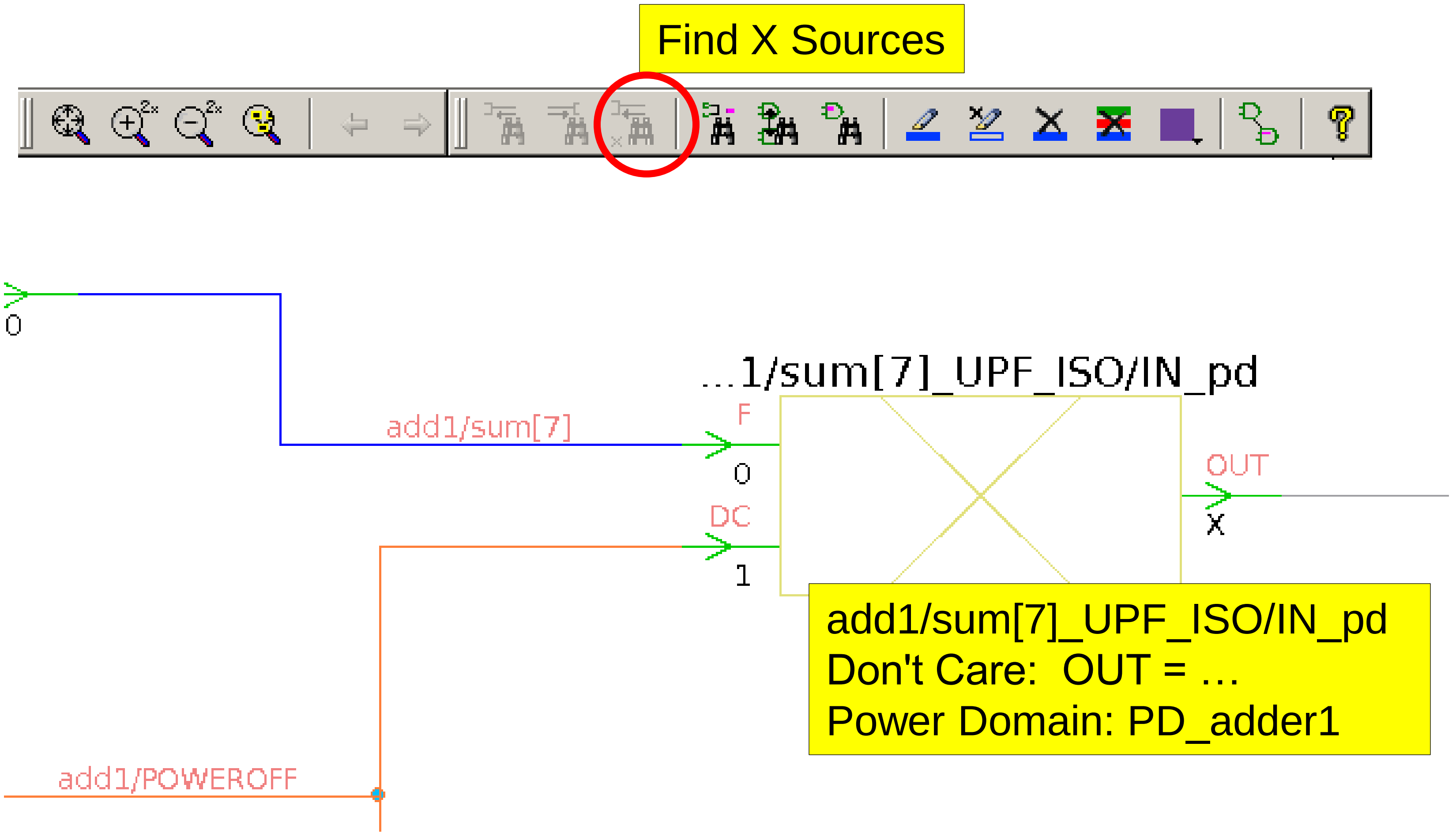


Ungroup to Look Inside a Block

Ungroup Selected



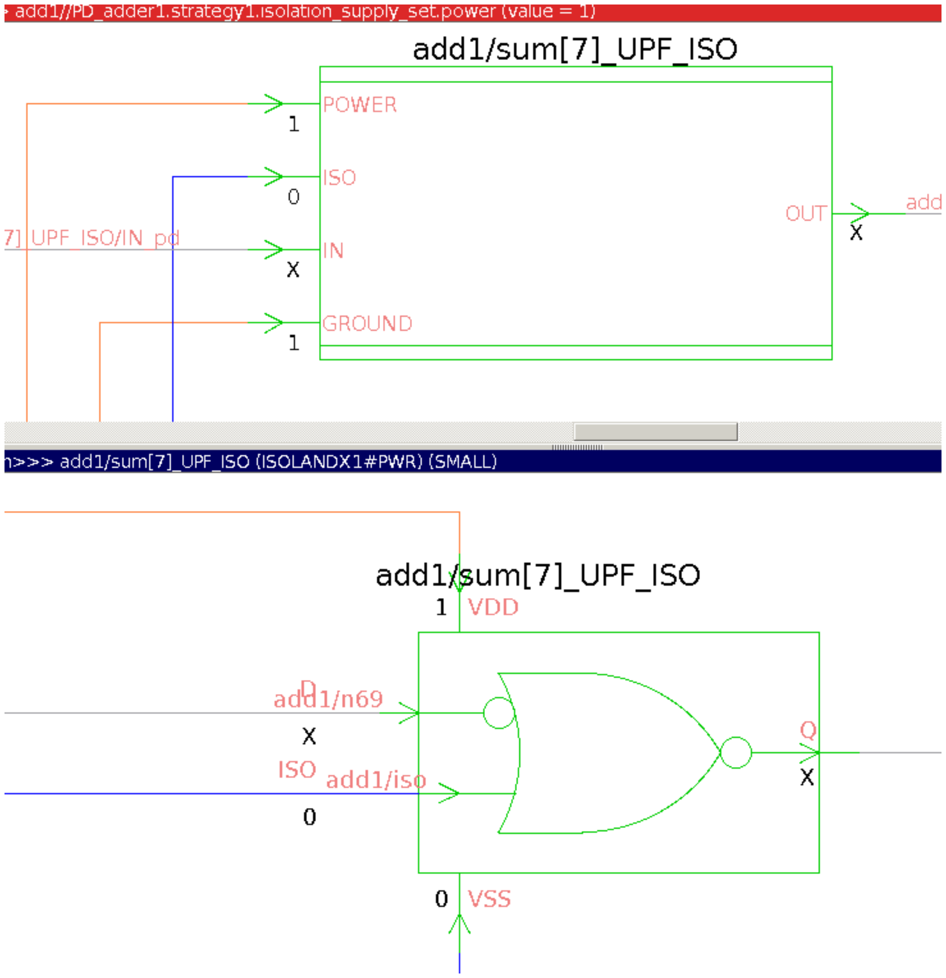
Finding Sources of X



add1/sum[7]_UPF_ISO/IN_pd
Don't Care: OUT = …
Power Domain: PD_adder1

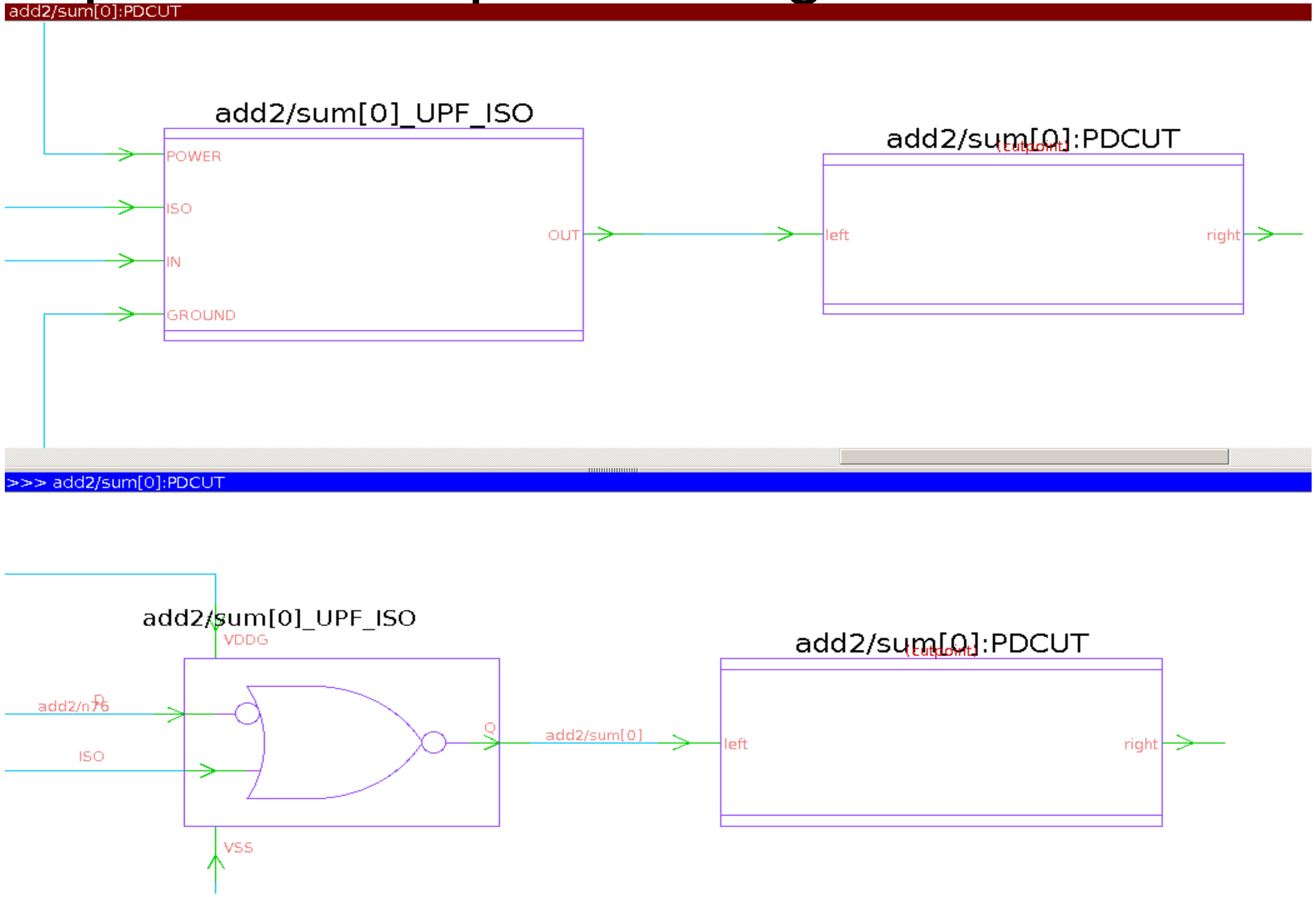
Separate Power and ISO Controls

X propagates through ISO cell



Power Domain Boundary Cutpoints
PDCUT CutPins

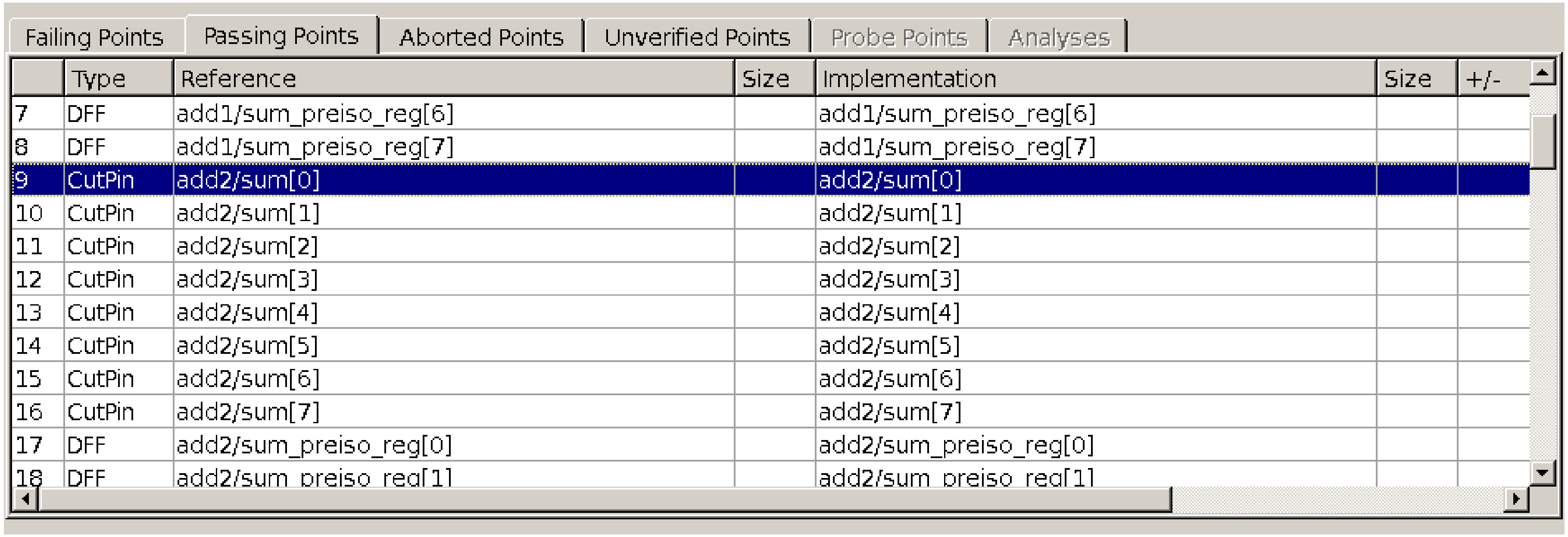
- Cutpins show up in the logic cone schematics



Power Domain Boundary Cutpoints

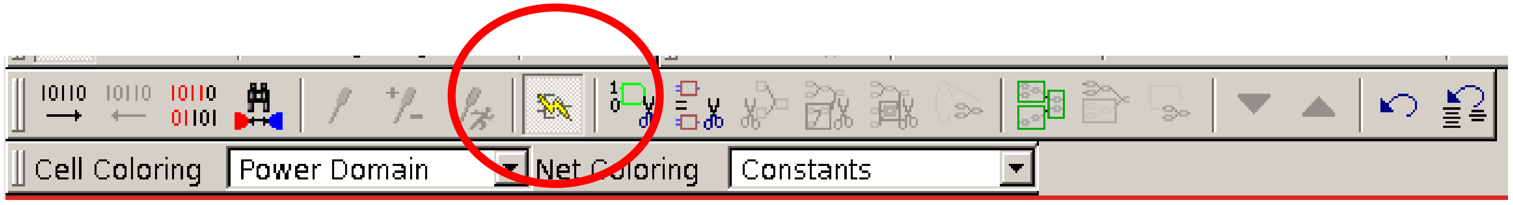
Compare Points

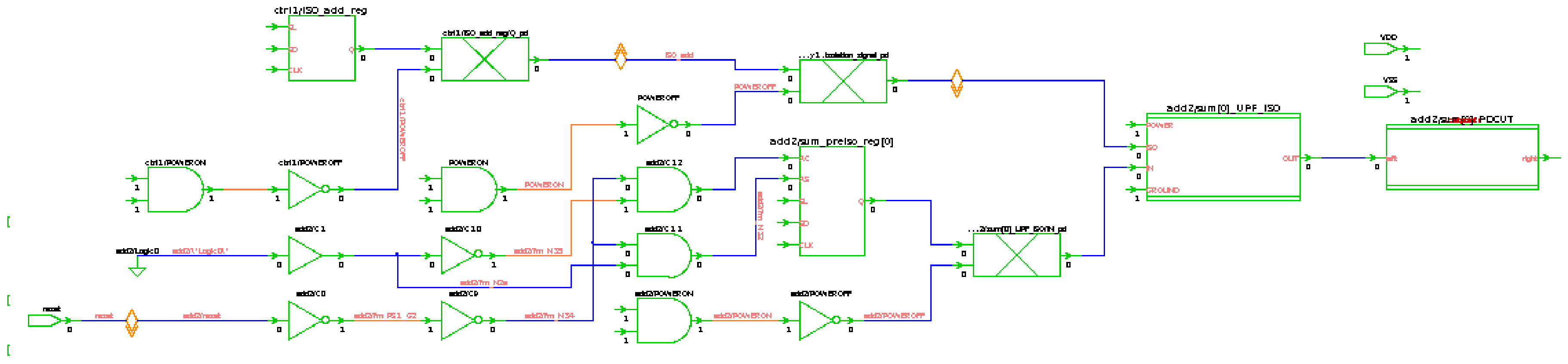
- CutPin compare points show up in the reports



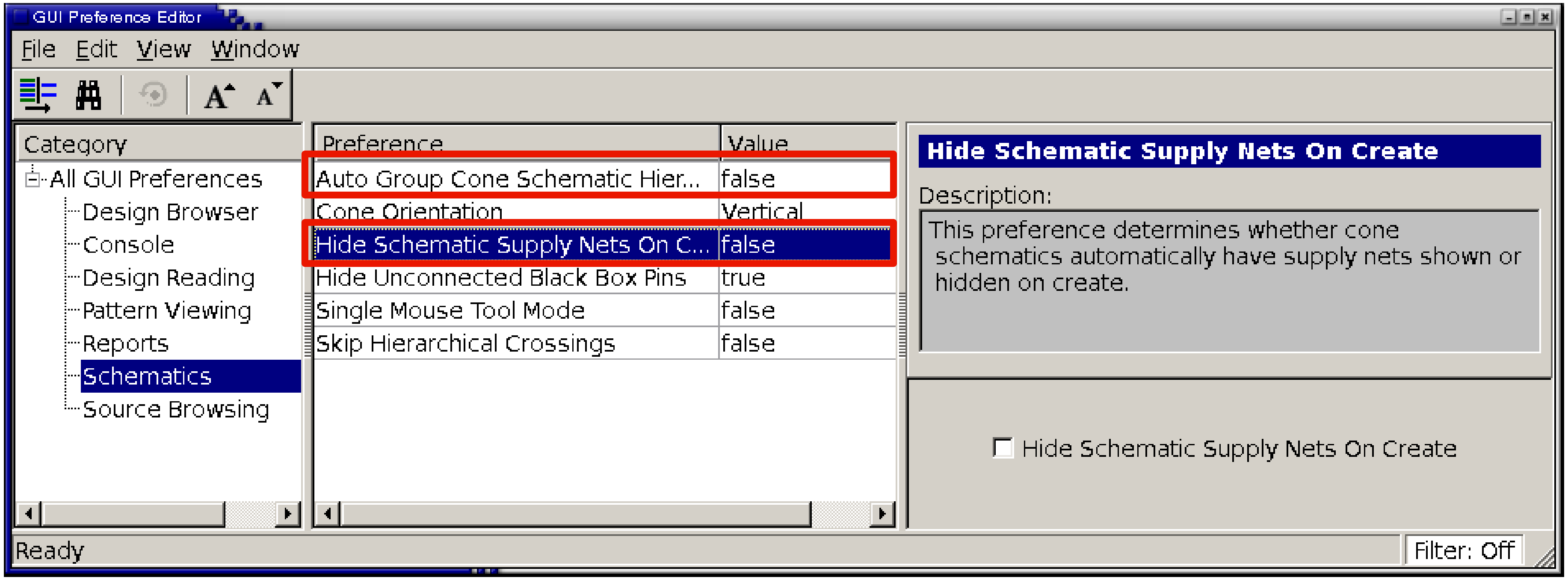
	Type	Reference	Size	Implementation	Size	+/-
7	DFF	add1/sum_preiso_reg[6]		add1/sum_preiso_reg[6]		
8	DFF	add1/sum_preiso_reg[7]		add1/sum_preiso_reg[7]		
9	CutPin	add2/sum[0]		add2/sum[0]		
10	CutPin	add2/sum[1]		add2/sum[1]		
11	CutPin	add2/sum[2]		add2/sum[2]		
12	CutPin	add2/sum[3]		add2/sum[3]		
13	CutPin	add2/sum[4]		add2/sum[4]		
14	CutPin	add2/sum[5]		add2/sum[5]		
15	CutPin	add2/sum[6]		add2/sum[6]		
16	CutPin	add2/sum[7]		add2/sum[7]		
17	DFF	add2/sum_preiso_reg[0]		add2/sum_preiso_reg[0]		
18	DFF	add2/sum_preiso_reg[1]		add2/sum_preiso_reg[1]		

Hiding Supply Nets


Cell Coloring: Power Domain
Net Coloring: Constants



GUI Prefs simplify setup



Category
All GUI Preferences
Design Browser
Console
Design Reading
Pattern Viewing
Reports
Schematics
Source Browsing

Preference	Value
Auto Group Cone Schematic Hier...	false
Cone Orientation	Vertical
Hide Schematic Supply Nets On C...	false
Hide Unconnected Black Box Pins	true
Single Mouse Tool Mode	false
Skip Hierarchical Crossings	false

Hide Schematic Supply Nets On Create

Description:

This preference determines whether cone schematics automatically have supply nets shown or hidden on create.

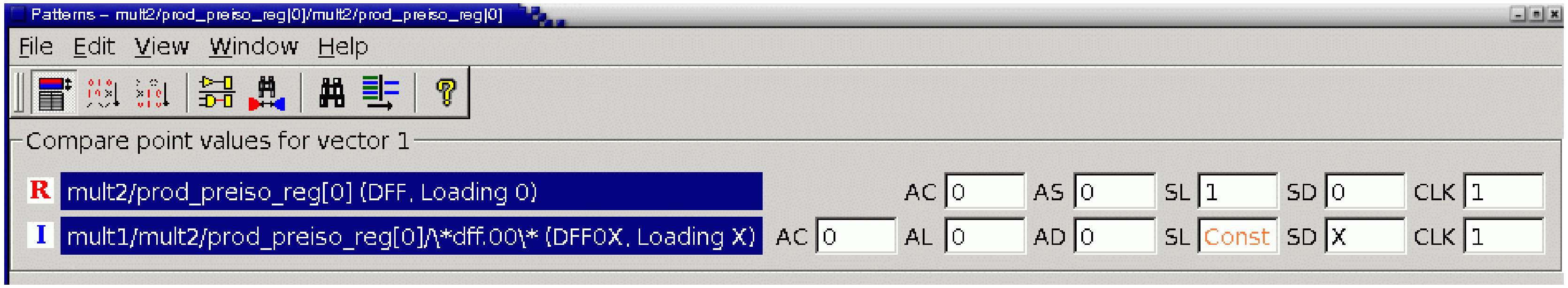
☐ Hide Schematic Supply Nets On Create

Ready

Filter: Off

X Differences

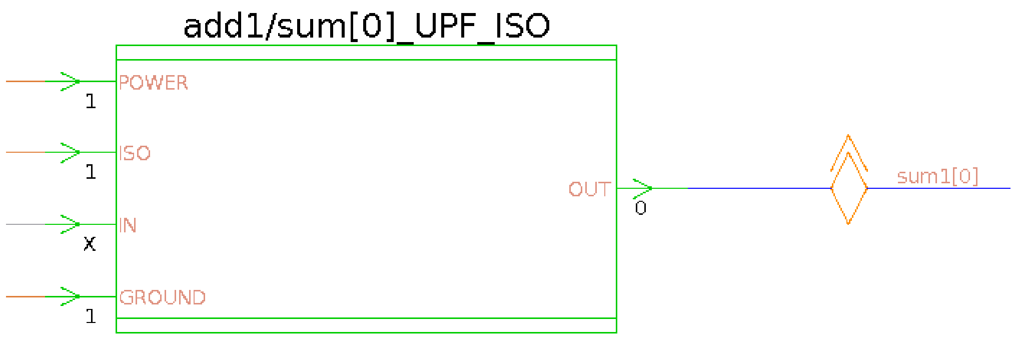
- If you see an X difference at a compare point when you are have set verification_force_upf_supplies_on true
  - You probably have a model with a bad power down function (nothing should be off)
  - Or something is undriven.

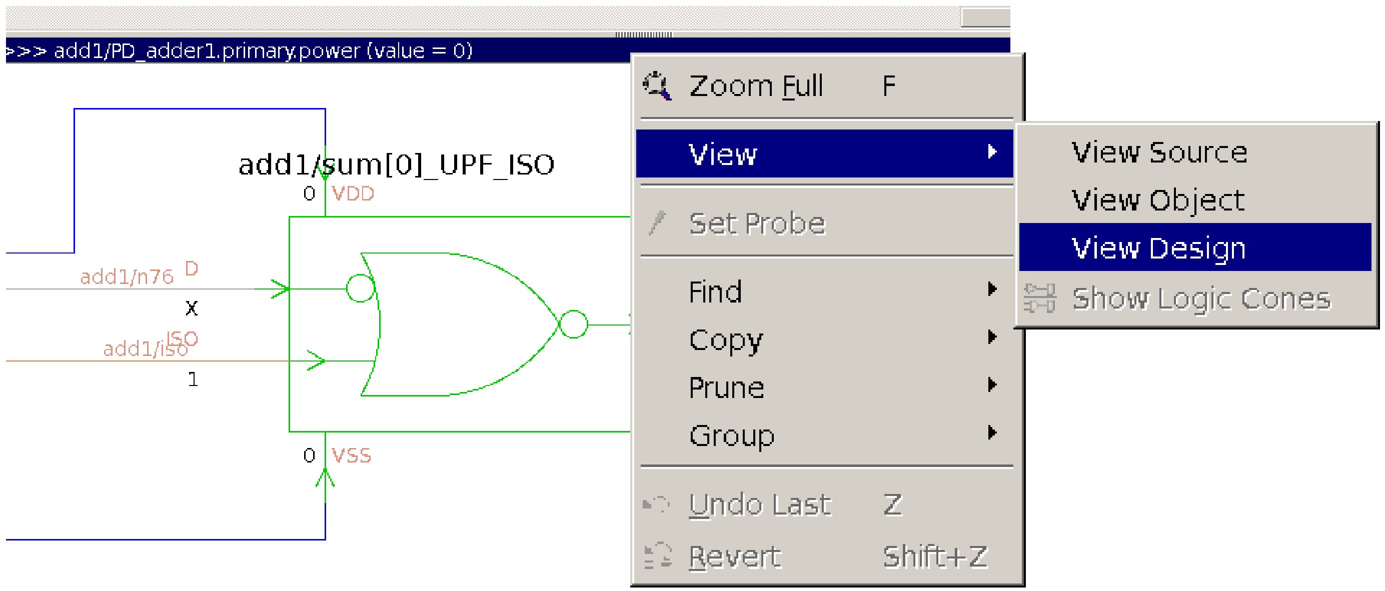

Patterns – mult2/prod_preiso_reg[0]/mult2/prod_preiso_reg[0]
File Edit View Window Help

Compare point values for vector 1

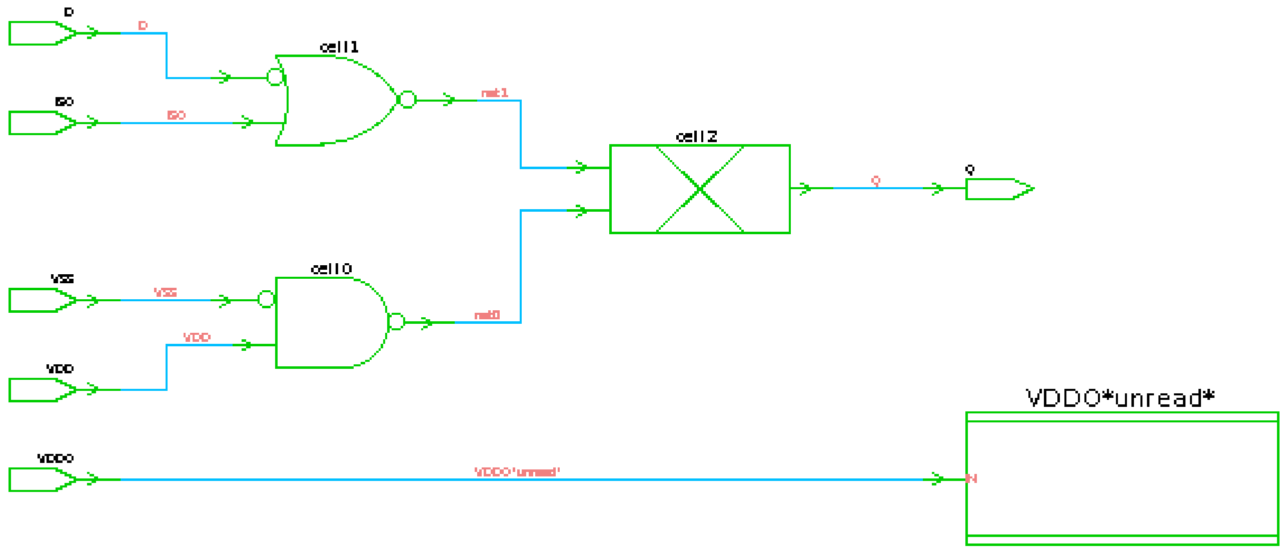
```
R mult2/prod_preiso_reg[0] (DFF, Loading 0)                                 AC 0  AS 0  SL 1      SD 0  CLK 1
I mult1/mult2/prod_preiso_reg[0]/\*dff.00\* (DFF0X, Loading X)      AC 0  AL 0  AD 0  SL Const  SD X  CLK 1
```

Looking inside tech cells checking the power down function




>>> add1/PD_adder1.primary.power (value = 0)

Looking inside tech cells checking the power down function



Verification and Debugging Summary

- New variable values in Formality 2011.09
  - – force all supplies on by default
  - – insert cutpoints at power domain boundary
- Pattern Window
  - – Look for supplies are on/off
- Match points window
  - – Quickly find which hierarchical boundaries are X
- Cone schematics
  - – coloring, grouping, hiding supplies …

Predictable Success

Agenda

- Part 1: Low Power Static Verification and Library Requirements
- Part 2: Verification and Debugging in Formality
- Part 3: Resolving Common Issues

UPF related issues in Verification

- Errors during load_upf
- Top level (and block level) port related supply assumptions in synthesis
- Retention Register models
- Test Insertion
- Handling IC Compiler designs
- …

Power State Table Defines the Legal Power States

- Poorly written port states

```
add_port_state VDDadd -state {ADD1_ON 1.1}
add_port_state VDDadd -state {ADD1_OFF 0.0}
add_port_state VDD -state {TOP_ON 1.1}
add_port_state VDD -state {TOP_OFF 0.0}
add_port_state VSS -state {GND off}
```

ON!

OFF!

```
create_pst TOP_PST -supplies                {VDD     VDDadd    VSS}
add_pst_state allon  -pst TOP_PST -state {TOP_ON  ADD1_ON   GND}
add_pst_state addoff -pst TOP_PST -state {TOP_ON  ADD1_OFF  GND}
add_pst_state alloff -pst TOP_PST -state {TOP_OFF ADD1_OFF  GND}
```

- Usually indicates the design has not been simulated.

Power State Table
Defines the Legal Power States

```
add_port_state VDDadd -state {ADD1_ON 1.1}
add_port_state VDDadd -state {ADD1_OFF off}
add_port_state VDD -state {TOP_ON 1.1}
add_port_state VDD -state {TOP_OFF off}
add_port_state VSS -state {GND 0.0}
```

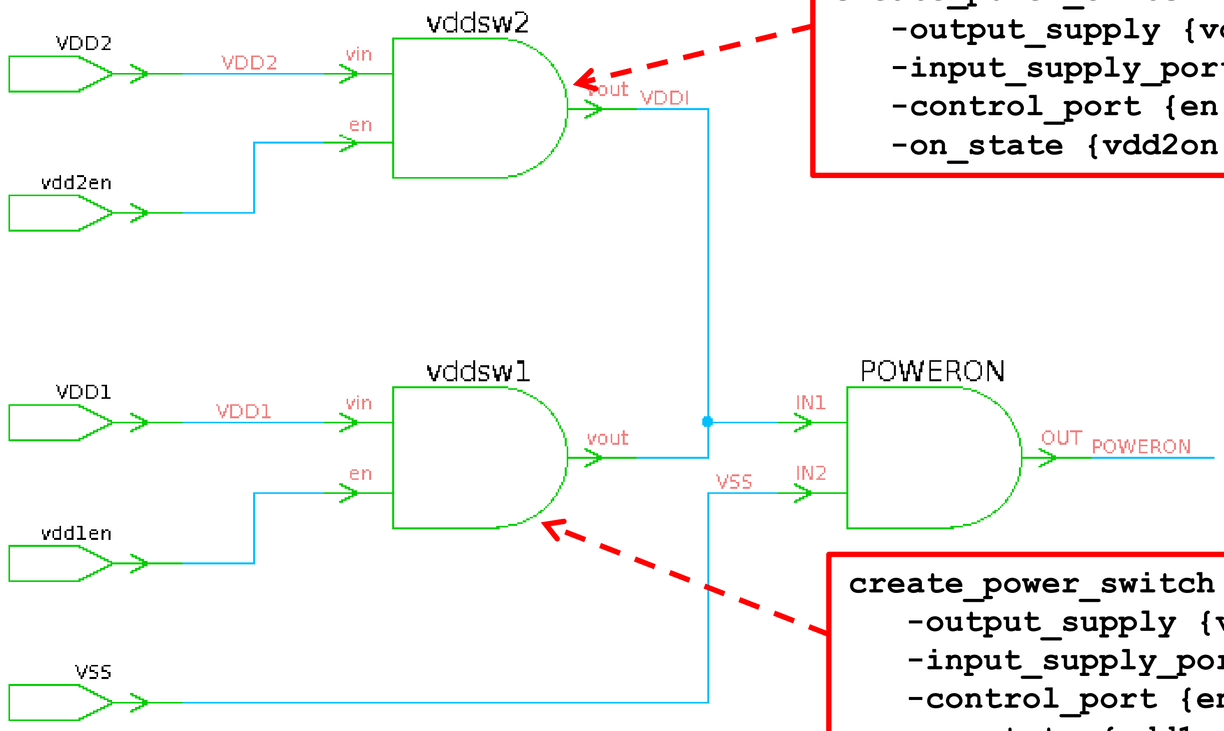
OFF

ON

```
create_pst TOP_PST -supplies                {VDD     VDDadd    VSS}
add_pst_state allon  -pst TOP_PST -state {TOP_ON  ADD1_ON   GND}
add_pst_state addoff -pst TOP_PST -state {TOP_ON  ADD1_OFF  GND}
add_pst_state alloff -pst TOP_PST -state {TOP_OFF ADD1_OFF  GND}
```

Parallel Resolved Supply Nets

- You have two switches driving the same supply net.



```
create_power_switch vddsw2 -domain TOP
   -output_supply {vout VDDI}
   -input_supply_port {vin VDD1}
   -control_port {en vdd2en}
   -on_state {vdd2on vin {en}}
```

```
create_power_switch vddsw1 -domain TOP
   -output_supply {vout VDDI}
   -input_supply_port {vin VDD1}
   -control_port {en vdd1en}
   -on_state {vdd1on vin {en}}
```

Parallel Resolved Supply Nets

- UPF

```
create_supply_net VDDI  -resolve parallel
…
create_power_switch vddsw1 -domain TOP -output_supply {vout VDDI} \
   -input_supply_port {vin VDD1} -control_port {en vdd1en}
   -on_state {vdd1on vin {en}}
create_power_switch vddsw2 -domain TOP -output_supply {vout VDDI} \
   -input_supply_port {vin VDD2} -control_port {en vdd2en}
   -on_state {vdd2on vin {en}}
```

- FM issues this Error message because Formality cannot verify the supply voltages are the same.

```
Error:  load_upf: parallel-resolved supply net /VDDI has different
root drivers; can't verify voltage semantics.
```

- If the nets are truly parallel driven
  - `set upf_warn_on_failed_parallel_resolved_check true`

Check for drivers on (supply) nets

New commands in 2010.12-SP1

- Useful for checking your Design+UPF.
- Use on the PG netlist from ICC.
- report_multidriven_nets/report_undriven_nets
  - **`report_multidriven -substring "VDD"`**
  - **`report_undriven -substring "VDD"`**

```
2 Multiply driven nets:
…
Net    r:/WORK/top/VDD
Drivers (3)
    r:/WORK/and_multi/vddsw1/OUT
    r:/WORK/and_multi/vddsw2/OUT
    r:/WORK/top/mid/out      (inv)   (TECH_WORK)
```

Black Box Supply Pins

Connecting to inout pins

- Instantiated power cells require specific connect_supply_net commands in the UPF. (e.g. pad cells)

```
Error:  load_upf: /home/testcase/design.dc.upf, line 147:
connect_supply_net: UPF receiver supply port
/top/pads/pad_inst23/VDD12 would drive supply net
/top/VDD12 contrary to declared direction; can't connect.
```

- Logical pin direction for supply ports in IN
  - Fix the library model
  - In Formality use the command **`set_direction`**
    - **`set_direction -shared -type pin /libname/cell#PWR/pin IN`**

Top Level Port Assumptions

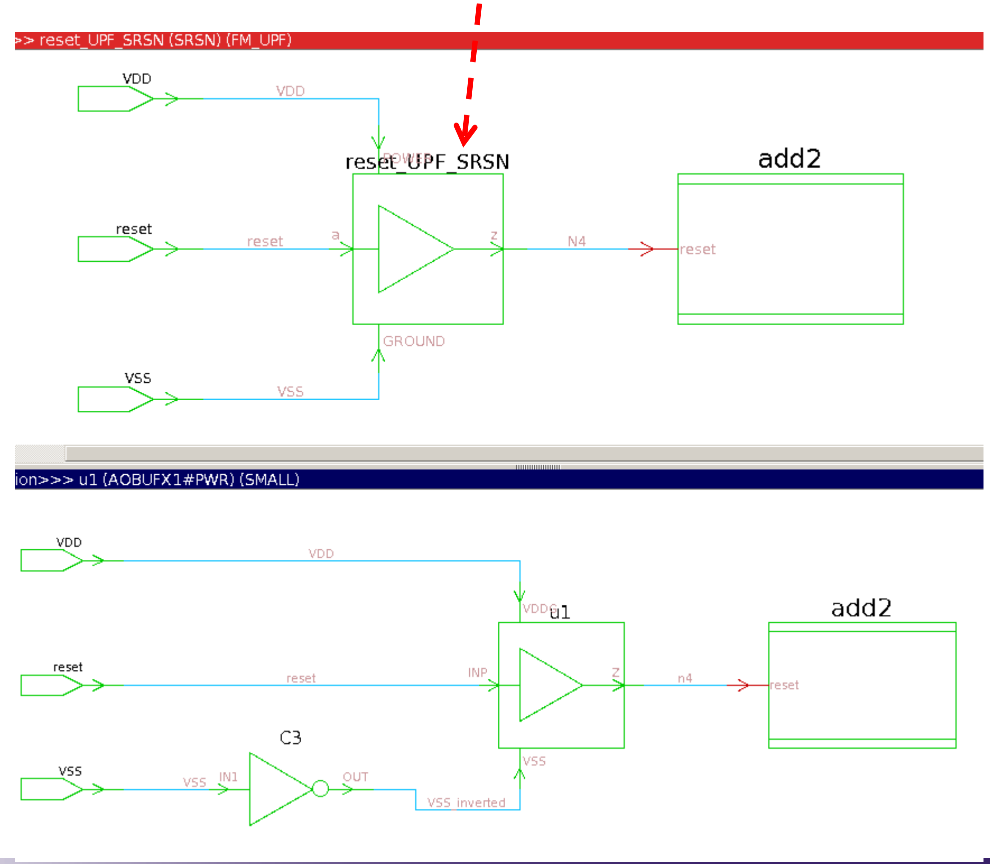
- By default Design Compiler assumes the when you do compile_ultra the top level ports are driven(and read) by the top level domain UPF supply
- The following UPF command will make that assumption explicit for all tools in the flow.

```
– set_port_attribute -elements {.} –attribute \
      related_supply_default_primary true
```

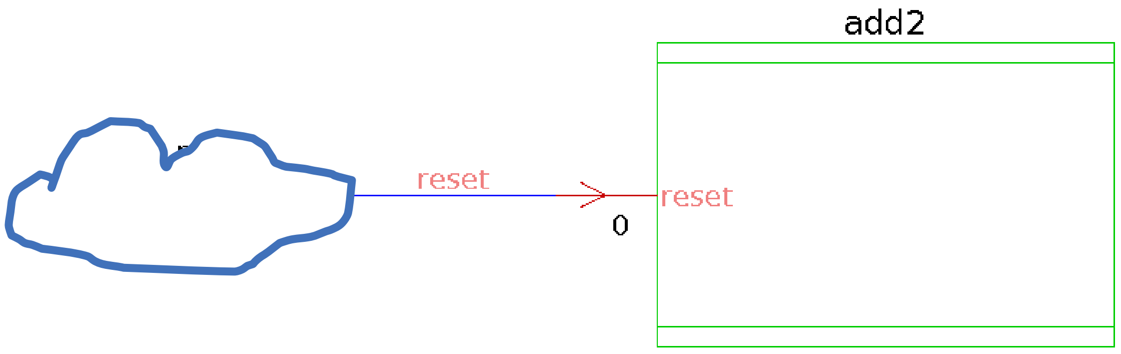
Top Level Port Assumptions

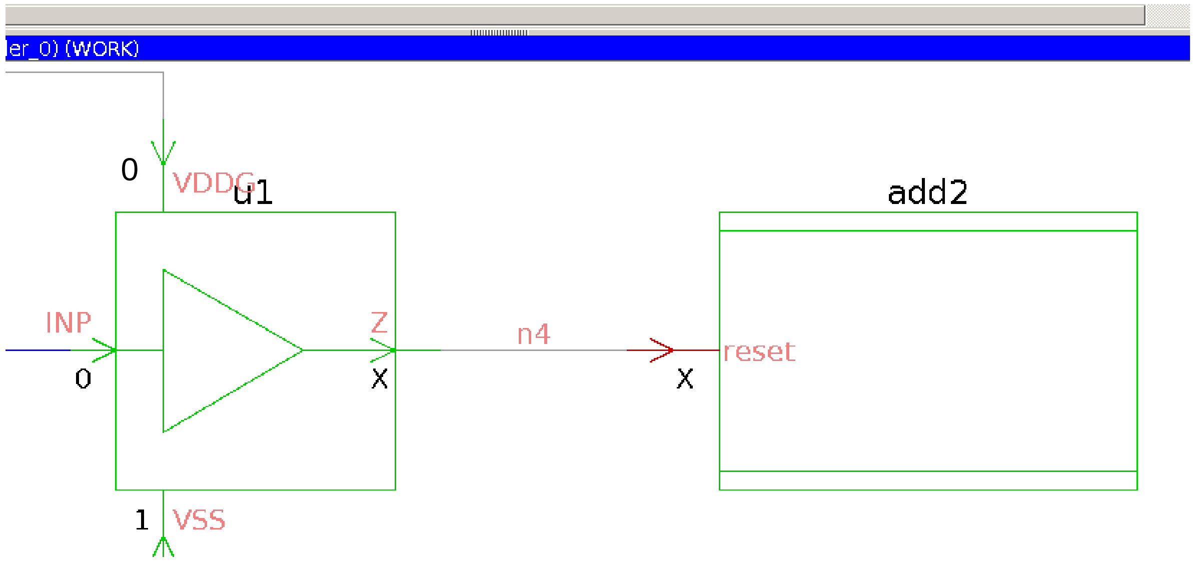
Related Supply Nets

```
    set_port_attribute -elements {.} -attribute \
        related_supply_default_primary true
```



Failures at Black Box Pins

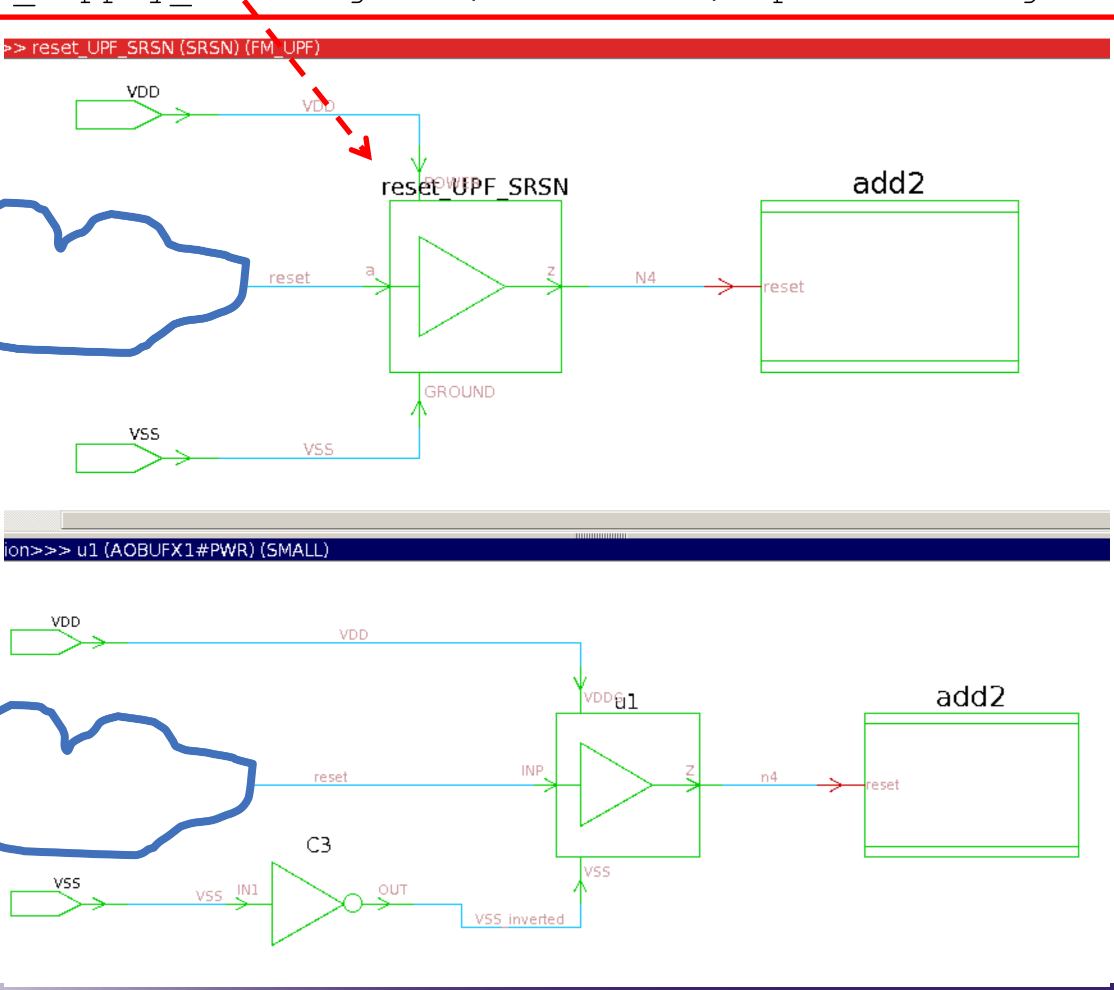




Failures at Black Box Pins

`set_related_supply_net`

```
set_related_supply_net -object {add2/reset} -power VDD -ground VSS
```



Failures at Black Box Pins

Causes and solutions

- Macro cells models need related power pin info
  - Update macro cell .lib/.db with related power/ground pin information
- For true black boxes, you can use set set_related_supply_net in the UPF on specific objects
  - Black box pin functional pin
  - `set_related_supply_net  -object_list {add2/reset}\`
    `-power VDD -ground VSS`
  - `set_related_supply_net  -object_list {reset} \`
    `-power VDD -ground VSS`
- Black Box power supply pin – real problem ?

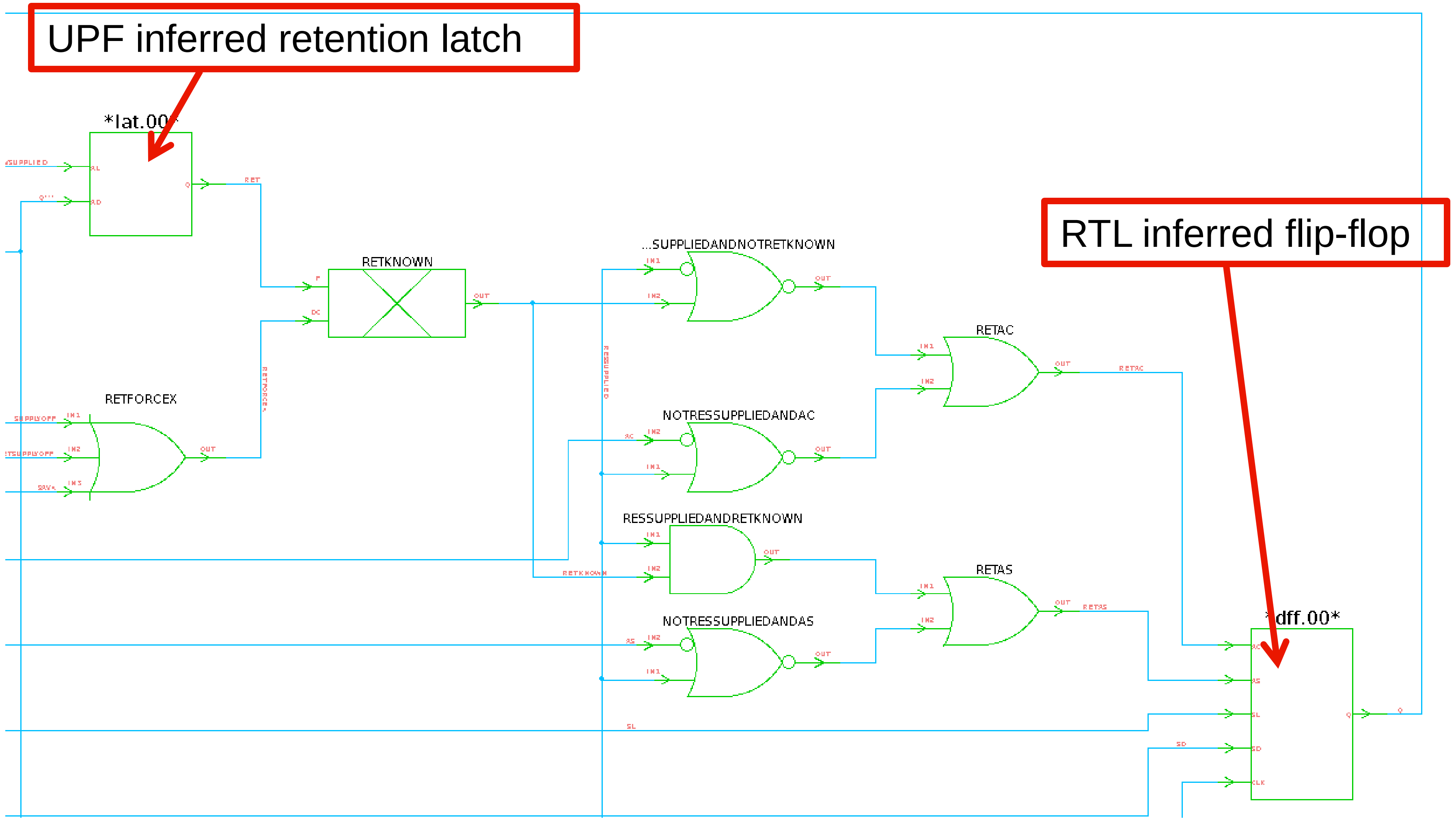
Retention Register Issues

UPF set_retention effects

- RTL + UPF infers a second process for each retained flip-flop which infers a latch.
- Technology cell models should also have a flip-flop+latch to match the RTL+UPF
- You will see verification failures if you have invalid retention register models
  - analyze_points
- Check Solvnet for the latest articles on handling Retention Registers

Formality RTL+UPF Retention Flop

Inferred Model Structure



Retention Register Issues

Liberty function for flop+latch retention cell

- Matches the structure of RTL+UPF

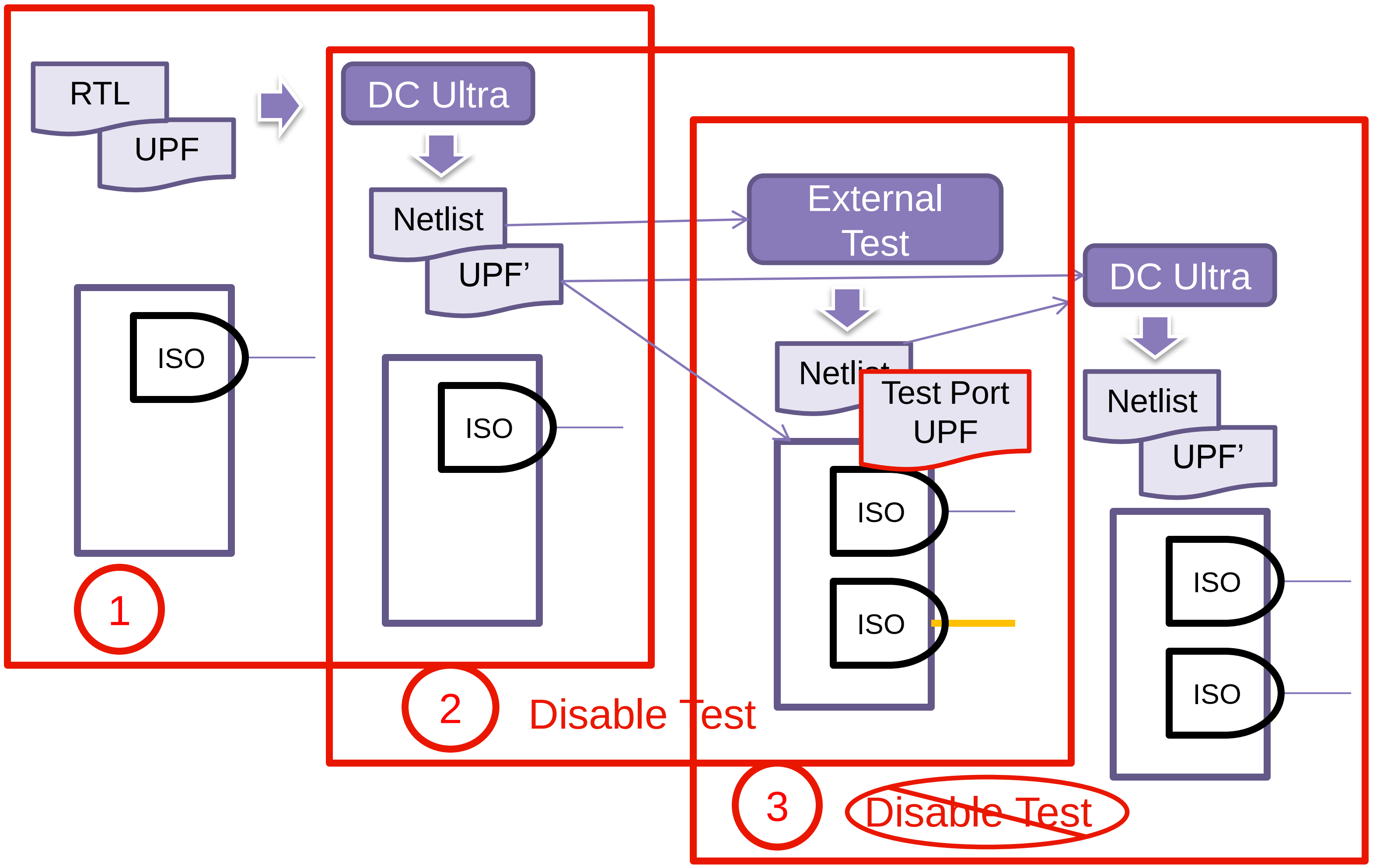
```
ff (IQ,IQN) {
     clocked_on : "CLK";
     next_state : "D";
     clear :  "!RSTB +  (!CLK & (!RETN & !IQ2))";
     preset : " (!CLK & (!RETN & IQ2))";
     clear_preset_var1 : L;
     clear_preset_var2 : H;
     power_down_function : "!VDD + VSS";
}

latch (IQ2,IQ2N) {
     enable : "RETN";
     data_in : "IQ";
     power_down_function : "!VDDG + VSS";
}
```

Retention Register Issues

- Instantiated clock gate cells in RTL+UPF will get retained.
  - The UPF should use `set_retention –no_retention…`
  - If you cannot modify the UPF use the variable `set upf_use_additional_db_attributes true`
- Retiming of retention devices not yet supported

Non-Synopsys Test Insertion

in the Synopsys flow



Non-Synopsys Test Insertion

Test Port UPF

- Read in a top level UPF which loads the UPF' and new test port isolation strategies

```
fm_shell> load_upf -r top.upf
```

```
##### top.upf
load_upf design.mapped.upf
load_upf new_ports_to_isolate.upf
```

```
##### new_ports_to_isolate.upf
set_isolation eco1  -domain PD_adder2  -isolation_power_net VDDiso \
  -isolation_ground_net VSS  -clamp_value 0  -elements {add2/sum[0]}
set_isolation_control eco1  -domain PD_adder2  -isolation_signal \
   ctrl1/ISO_add  -isolation_sense high  -location self
```

new isolation strategy

hierarchal ports to isolate

Inconclusive verifications

- Use `analyze_points`
- Check the svf guidance summary report
- What were the results with `verification_force_upf_supplies_on true` ?
- Loading UPF can affect svf processing
  - Rejected svf can cause inconclusive results
  - Check the svf guidance summary report and compare with the all supplies on report
  - File a STAR if loading UPF caused svf rejections (and an inconclusive verification)

Handling designs from IC Compiler

Supply0/Supply1 in ICC PG Netlist

- When writing out the PG netlist verilog
  - You don’t want supply0/supply1 statements for verification or simulation

```
output tbo_DATA__0 ;
input  VSS ;
input  VDD_SW ;

supply0 VSS ;
supply1 VDD_SW ;
```

```
## For comparison with a Design Compiler netlist, the option -diode_ports is removed
write_verilog -no_physical_only_cells -pg -supply_statement none
    $RESULTS_DIR/$DESIGN_NAME.output.pg.dc.v
```

Reading UPF'' from IC Compiler

- Errors during load_upf with UPF''

```
Error:  load_upf: /home/test/top_after_icc.upf, line 43:
  add_port_state: port /SW3/VDD does not exist.
```

- Addressed in IC Compiler and Formality 2011.09-SP1
- Workaround for earlier versions
  - in the UPF'' change the switch (and pin) name in the add_port_state command to the name of one of the implemented switches (and pin) in the ICC netlist

Handling designs from IC Compiler

PG netlist vs. PG netlist

- Formality Symptom: Failing points in a power state that is not legal.
  - Clock gating (changes) can cause X to move to D and CLK pins
- All power states are verified (no UPF is read) causing illegal combinations of power up/down
  - – User defined constraints to mimic power state table can eliminate these failures in unreachable states.

Hierachical Flows

- Each hierarchical block should have its own upf. The upf should be self contained.
  - do not reference objects outside the design
  - no set_isolation -location fanout/parent
- The top level upf should use connect_supply_net to connect to lower level blocks
- In 2011.09 UPF support includes create_logic_net/port, connect_logic_net

Additional Formality Flow support Non-UPF MV flows

- You can load a design that already has primary ports connected to cell PG pins
  - set hdlin_allow_partial_pg_netlist true
- UPF in ref has some/no isolation, Isolation cells are inserted in ICC
  - Set constants on isolation controls
  - Single point verify the isolation control signals/registers

Other Possible Issues using UPF
Known FM issues as of 2011.09

- If you have properly set constants to disable the scan paths and you have lockup latches that fail then
  - Likely a Formality bug. Please file a STAR
- set verification_clock_gate_edge_analysis true may cause erroneous failures when using UPF
  - Should be resolved in 2011.09-SP2

Other Issues using Formality variables

what is the difference between these two variables

- upf_derive_pst_constraints = "true"
  - Formality will automatically constrain away states that violate the PST. (without this you may get failing points that can never be reached)
- upf_derive_supply_constants = "true"
  - If a supply in the PST is on/off in every possible state, formality will set it to a constant.
  - You may see only constants driving that net, it will no longer be a cone input!
  - Often happens with VSS. (value c1 in the cones)

Resolving Common Issues Summary

- Errors during load_upf
- Top level (and block level) port related supply assumptions in synthesis
- Retention Register models
- Test Insertion
- Handling IC Compiler designs
- Hierarchical Flows

- Read the new upf

Predictable Success