The need for combining static and dynamic LP verification

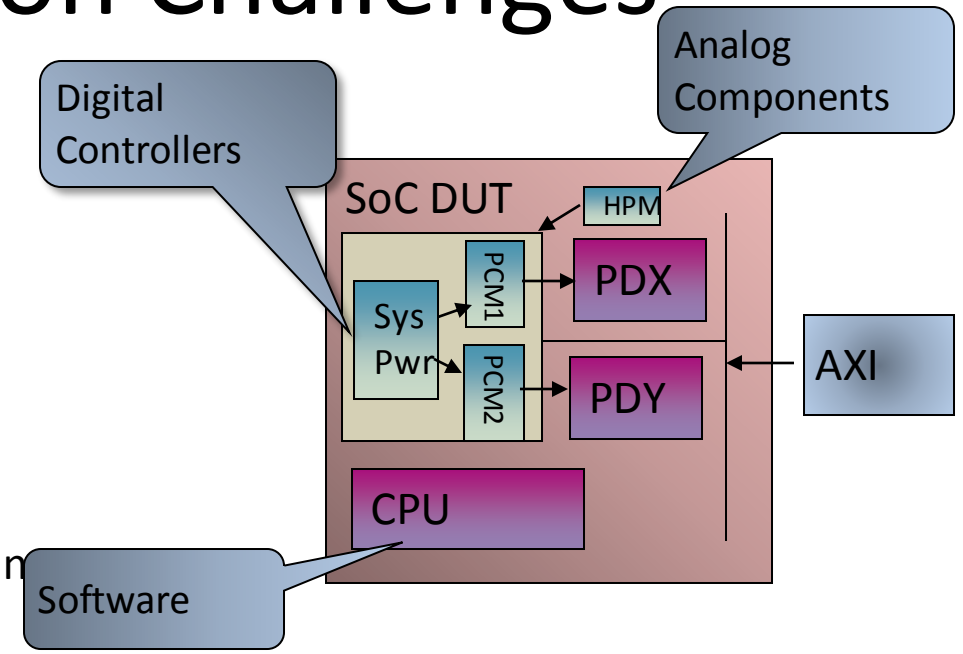
Himanshu Bhatt, Prashanth M, Adiel Khan
(Synopsys)

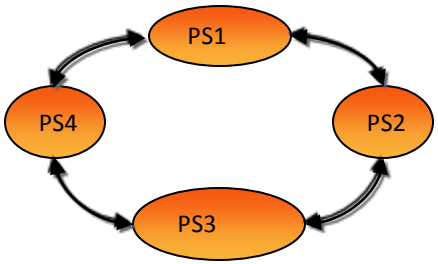
accellera SYSTEMS INITIATIVE

2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

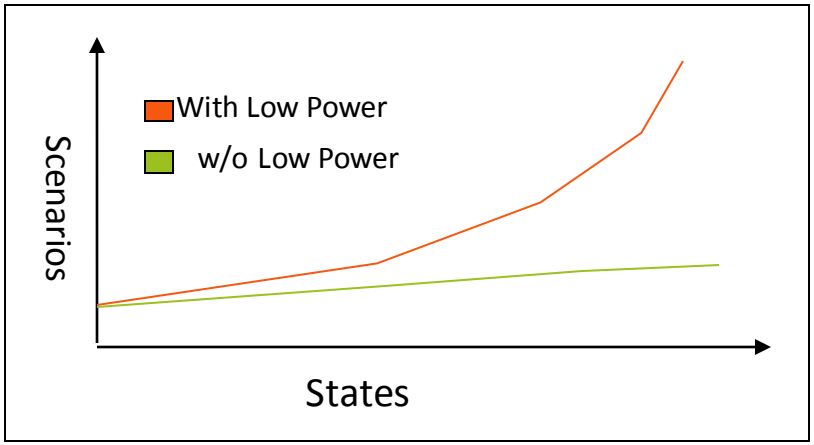
LP Verification Challenges

- Huge verification space
  - Large number of power states
  - Large number of transitions
    - Software applications
    - Firmware
    - Digital Hardware
- System level verification
  - Reuse in larger system
  - Often requires HW/SW simulation
- LP specification extensive
  - New versions





Power State Transitions



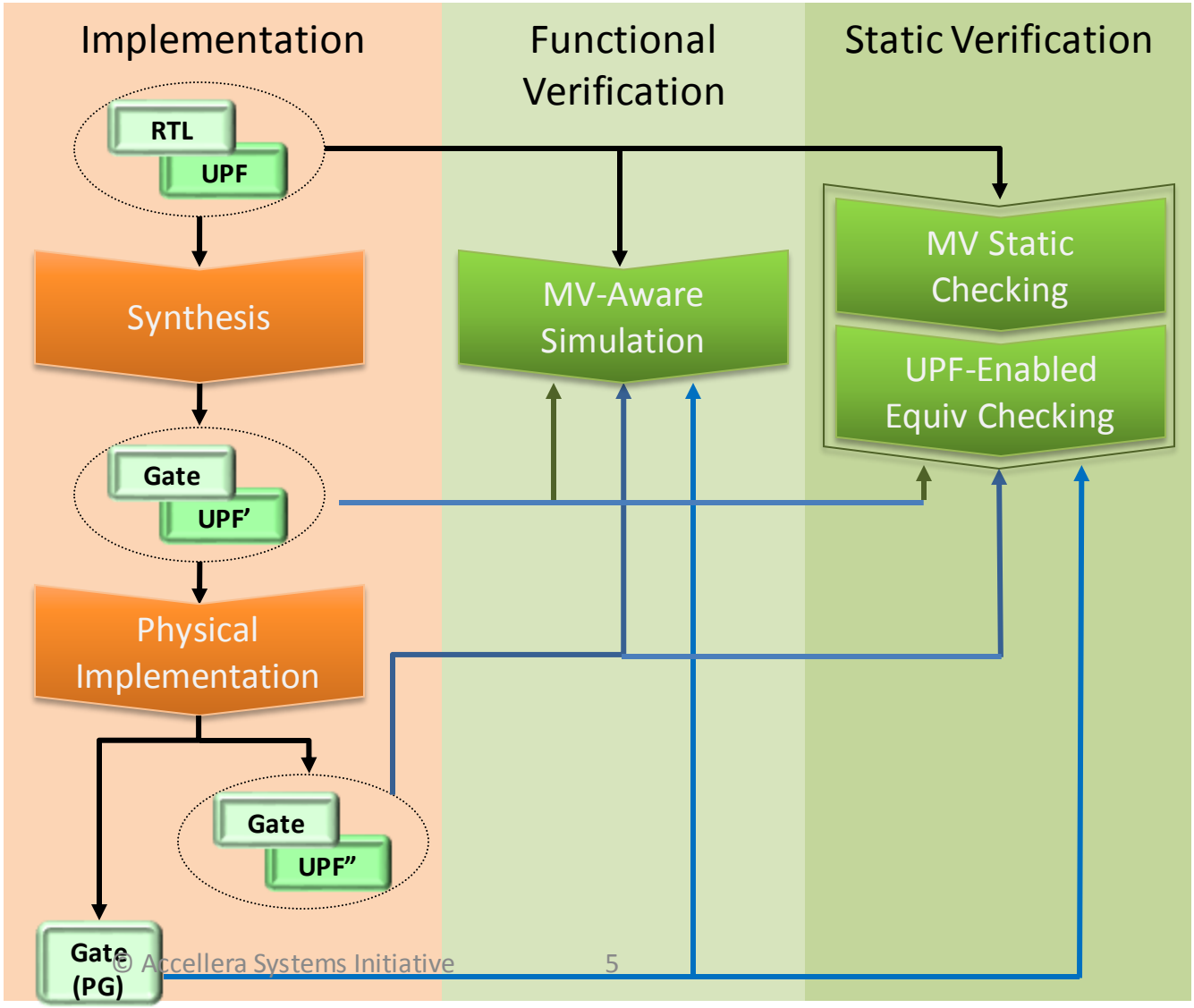
LP Verification Challenges (Contd.)

- Does that make sense to validate simultaneously the UPF and the design?
- How do you verify your PG netlist is electrically safe?
- How do you verify your PST coverage?
- Is there a way to find a bug in my PST?
- Will you really run all tests in all LP modes at RTL/netlist/PG netlist stage?
- Where are the critical paths (clock, reset, scan-enable …) that requires specific attention?
- Soc PST is the key, did we say merging?
- Was my UPF properly understood and implemented?
- Is retention working? Is isolation working? How to validate this?
- Are my LP tests passing?

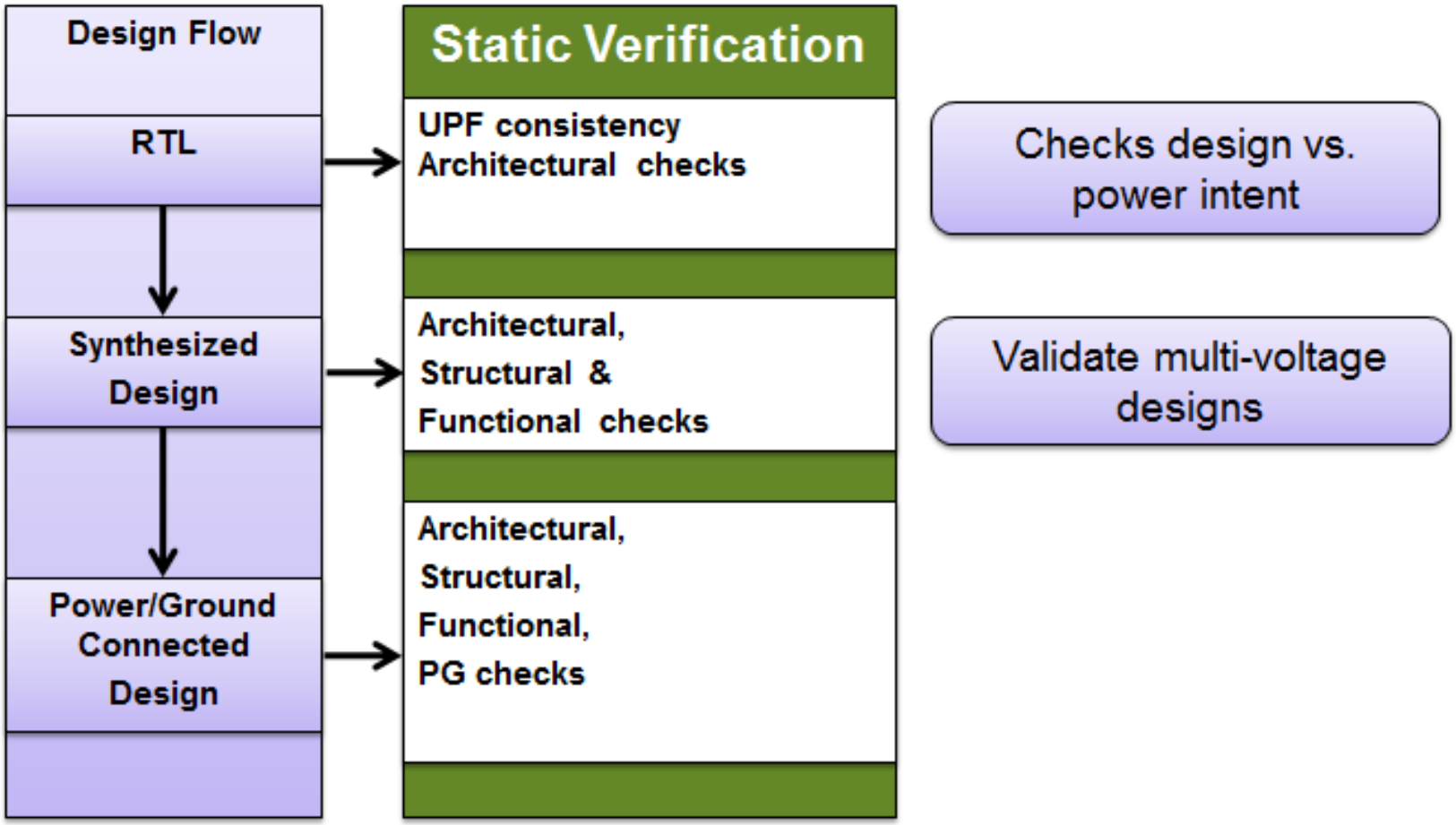
Dynamic Verification v/s Static Verification

Dynamic Verification	Static Verification
Main focus is on functionally correct power related "sequences"	Main focus is on design related low power checks
The abstraction level is mostly RTL (GLS is both time consuming and has a huge performance impact)	This is very useful in validating RTL netlist or PG netlist
Requires creation of test stimulus (verification can not start unless a power aware test-bench is in place)	This can be very useful in flow-flushing UPF related issues (does not require a test-bench)
Dynamic verification is needed to validate that the output values match with the expected values (e.g. if isolation clamp value is specified as '1', the isolation port should propagate '1'. Similarly the simulation output should show the correct value that needed to be retained on power up)	Static verification can not perform such checks
Thanks to both coverage and assertions techniques, one can verify that the device is operating properly in all defined low power modes. Furthermore, still with respect to the PST, corruption can be done when the power value is not in the range specified by the PST.	Useful for PST merging

Combined Dynamic and Static Low Power Flow



Static Verification flow across design flow



What is a PST?

- Power State Table (PST) is a construct in UPF that captures the legal combinations of power states for a set of supply ports/nets.

- One or multiple PST can be defined in each scope of the UPF.
- Create Port in the present scope : `create_supply_port P1`
- Add states to the Port :

```
add_port_state P1 -state {HV 1.0} \
                  -state {LV  0.5} \
                  -state {OFF off}
```

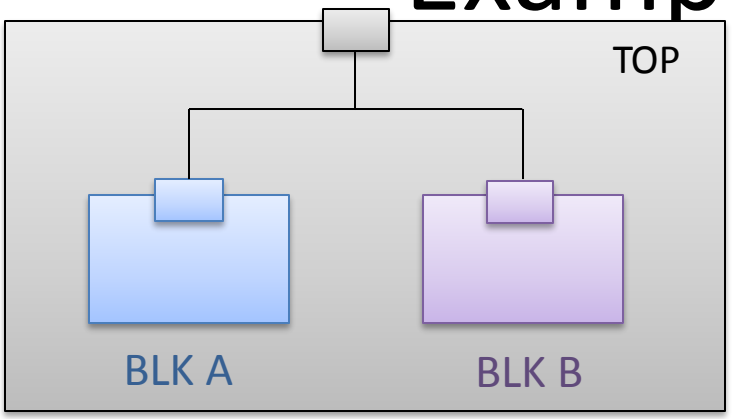
- Create a PST specifying the port/nets which are involved

```
create_pst PST_1 -supplies { P1 P2 P3 }
```

- Different states of the PST

```
add_pst_state s1 -pst PST_1 -state { HV   HV   LV }
add_pst_state s2 -pst PST_1 -state { HV   HV   LV }
add_pst_state s3 -pst PST_1 -state { HV   LV  off }
```

Example PST Merging



	TOP
TOPV 1_2	1.2
TOPV 1_0	1.0
TOPV 0_8	0.8

	BLK A
BLKAV 1_2	1.2
BLKAV 0_8	0.8
BLKAV 0_6	0.6

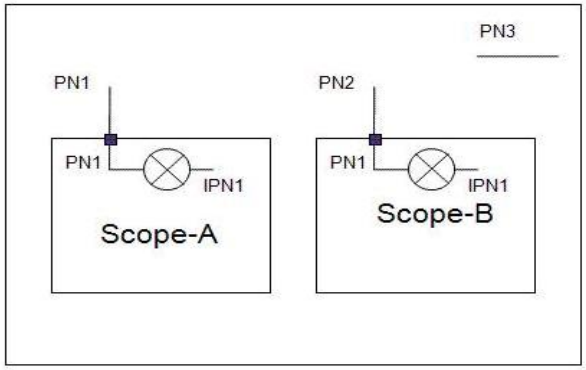
	BLK B
BLKBV 1_2	1.2
BLKBV 1_0	1.0
BLKBV 0_8	0.8

Merged PST

	TOP	BLK A	BLK B
merged_pst_st0	1.2	1.2	1.2
merged_pst_st1	0.8	0.8	0.8

PST merging 1

- At SOC level, PST merging will highlight IP/SOC PST inconsistency in identifying missing PST states at IP level



Top-level PST:

PN1	PN2	PN3	
1.0	2.0	2.0	(ps-top-1)

Scope-A PST

A/PN1	A/IPN1	
1.0	1.0	(ps-A-1)
1.0	off	(ps-A-2)

Scope-B PST

B/PN1	B/IPN1	
1.0	1.0	(ps-B-1)
1.0	off	(ps-B-2)
2.0	2.0	(ps-B-3)
2.0	off	(ps-B-4)

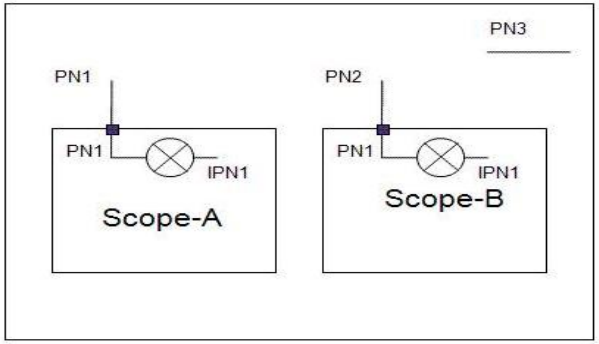
Warning: power states are not used for implementation

Derived PST (intersection of above 3 PSTs)

PN1	A/IPN1	PN2	B/IPN1	PN3
1.0	1.0	2.0	2.0	2.0
1.0	1.0	2.0	off	2.0
1.0	off	2.0	2.0	2.0
1.0	off	2.0	off	2.0

PST merging 2

- At SOC level, PST merging will highlight IP/SOC PST inconsistency in identifying missing PST states at IP level



Top-level PST:

PN1	PN2	PN3	
1.0	2.0	2.0	(ps-top-1)
2..0	2.0	2.0	(ps-top-2.)

Scope-A PST

A/PN1	A/IPN1	
1.0	1.0	(ps-A-1)
1.0	off	(ps-A-2)

Will be dropped as Scope-A has not been validated with PN1=2V

Scope-B PST

B/PN1	B/IPN1	
2.0	2.0	(ps-B-3)
2.0	off	(ps-B-4)

Derived PST (intersection of above 3 PSTs)

PN1	A/IPN1	PN2	B/IPN1	PN3
1.0	1.0	2.0	2.0	2.0
1.0	1.0	2.0	off	2.0
1.0	off	2.0	2.0	2.0
1.0	off	2.0	off	2.0

LS_INST_RANGEI/O

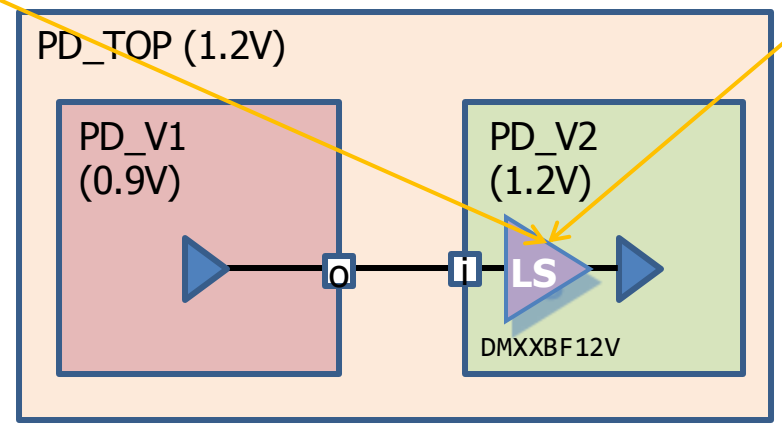
LS_INST_RANGEO

```
cell(DMLSUBF12V) {
  is_level_shifter : true;
  level_shifter_type : LH;
  input_voltage_range  (1.0,
1.2);
  output_voltage_range (1.0,
1.2);
```

LS_INST_RANGEI

```
cell(DMLSUBF12V) {
  is_level_shifter : true;
  level_shifter_type : LH;
  input_voltage_range  (1.0,
1.2);
  output_voltage_range (1.0,
1.2);
```

The low-to-high level shifter used has output voltage range of 1.0V ~ 1.2V, yet in design the output is at 1.3V.



The low-to-high level shifter used has input voltage range of 1.0V ~ 1.2V, yet in design the input is at 0.9V.

	PD_TOP (VDD2)	PD_V1 (VDD1)	PD_V2 (VDD2)
S1	1.3V	1.0V	**1.3V**

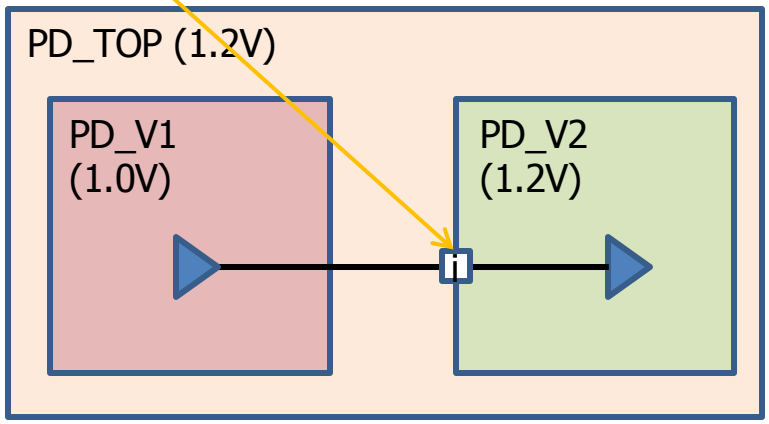
	PD_TOP (VDD2)	PD_V1 (VDD1)	PD_V2 (VDD2)
S1	1.2V	**0.9V**	1.2V

LS_THRESH_WRONG

LS_THRESH_WRONG

```
set_level_shifter LS_V2_in \
  -domain PD_V2 \
  -rule low_to_high \
  -applies_to inputs \
  -location self \
  -threshold 0.1
```

Threshold value specified is less than the actual voltage difference in design.



	PD_TOP	PD_V1	PD_V2
S1	1.2V	1.0V	1.2V

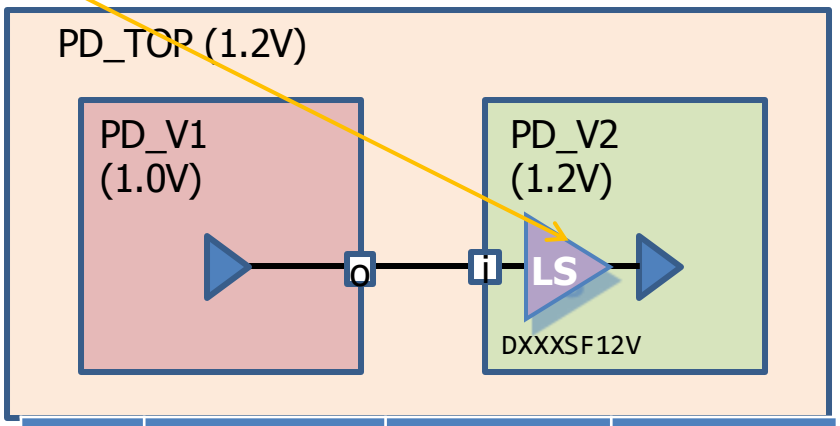
LS_INST_WRONG

LS_INST_WRONG

```
set_level_shifter LS_V2_in \
  -domain PD_V2 \
  -rule low_to_high \
  -applies_to inputs \
  -location self
```

```
cell(DMLSDBF12V) {
  is_level_shifter : true;
  level_shifter_type : HL;
  input_voltage_range  (1.0, 1.2);
  output_voltage_range (1.0, 1.2);
```

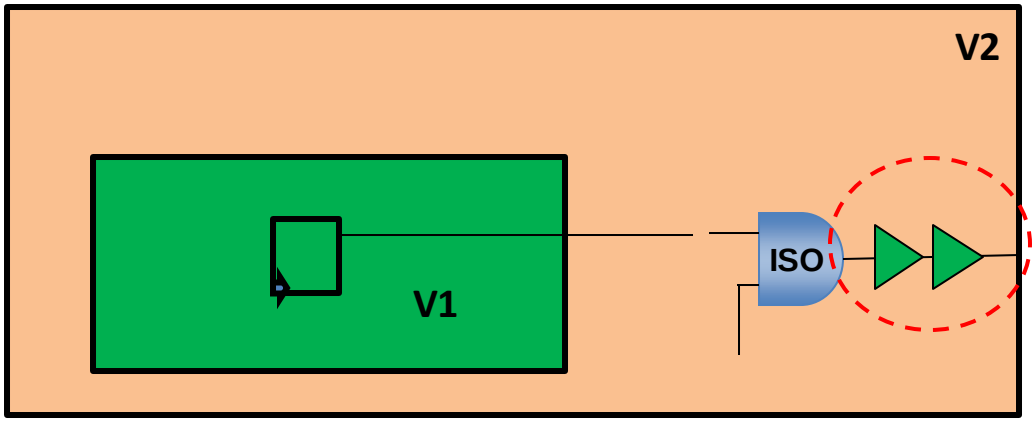
set_level_shifter command in UPF specifies a low-to-high level shifter, but there is a high-to-low type in design.



	PD_TOP (VDD2)	PD_V1 (VDD1)	PD_V2 (VDD2)
S1	1.2V	1.0V	1.2V

Rail-order checks: Corruption

A node driven by an OFF rail is driving the a node that is ON and the corruption is able to reaches destination (i.e, no isolation beyond the OFF node)

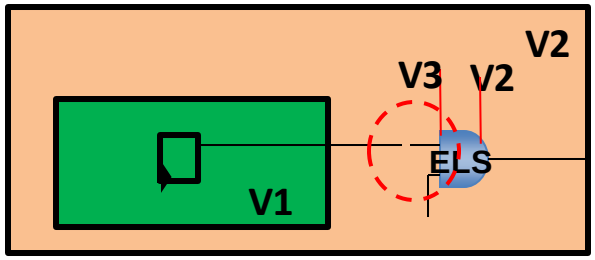


	V1	V2
S1	OFF	ON

ISO rail order violation:
Issue: Isolation required but isolation rail is OFF when destination is ON
Solution: Fix the incorrect rail connectivity or PST for isolation supply

Rail order checks: Leakage

- Logic source node power or ground off but load power and ground on.
- A node driven by an OFF rail is driving a node that is ON but the corruption does NOT reach the eventual destination (i.e., there is isolation beyond OFF node)



	V1	V2	V3
S1	OFF	ON	ON
S2	OFF	ON	OFF

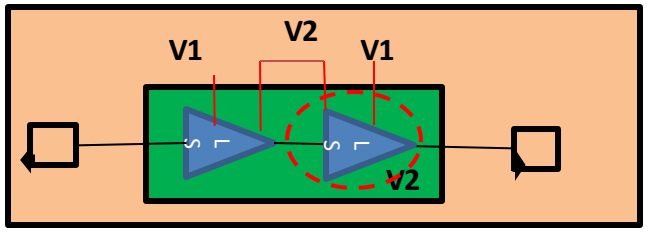
ISO rail order violation:

Issue: The supply net associated to the node is ON for some multivoltage state of legal state table, causing leakage in the path.

Solution: Fix the incorrect rail connectivity or PST for isolation supply

Rail order checks: Over consumption

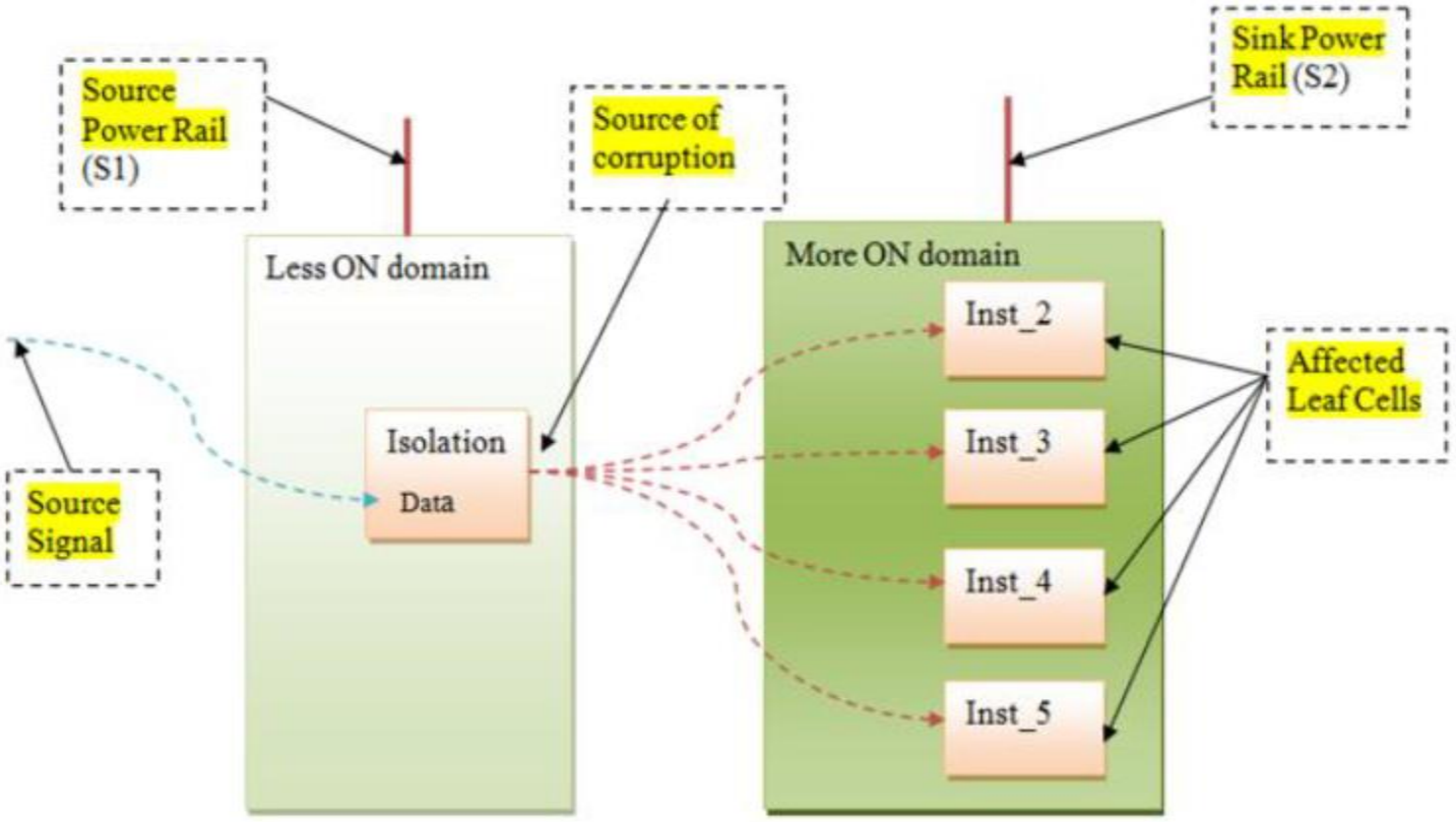
- Instance output power and ground on but logic sink power or ground off.
- A node driven by an ON rail is driving a node that is OFF unnecessarily



	V1	V2
S1	1.2	1.4
S2	OFF	ON

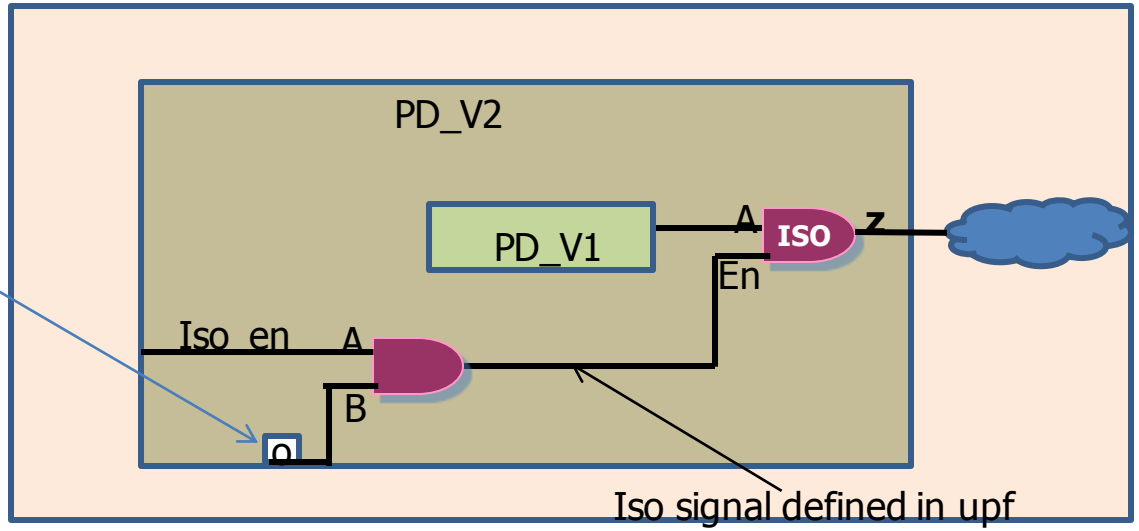
ISO rail order violation:
Issue: The supply net associated to the node is ON for some multi voltage state of legal state table, causing current over consumption in the path.
Solution: Fix the incorrect rail connectivity or PST for isolation supply

Identifying critical control signal that crosses multiple power domain

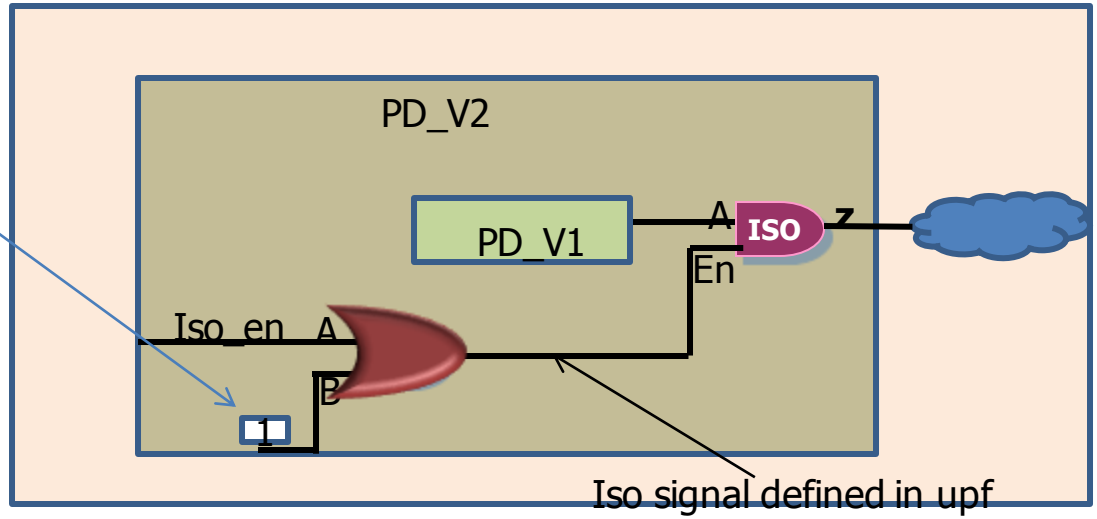


Constant propagation on enable path

AND gate on the iso_enable path tied to 1'b0



OR gate on the iso_enable path tied to 1'b1



Questions