February 25-28, 2013
DoubleTree, San Jose

accellera SYSTEMS INITIATIVE

SYNOPSYS®
Accelerating Innovation

A Systematic Approach to Power State Table (PST) Debugging

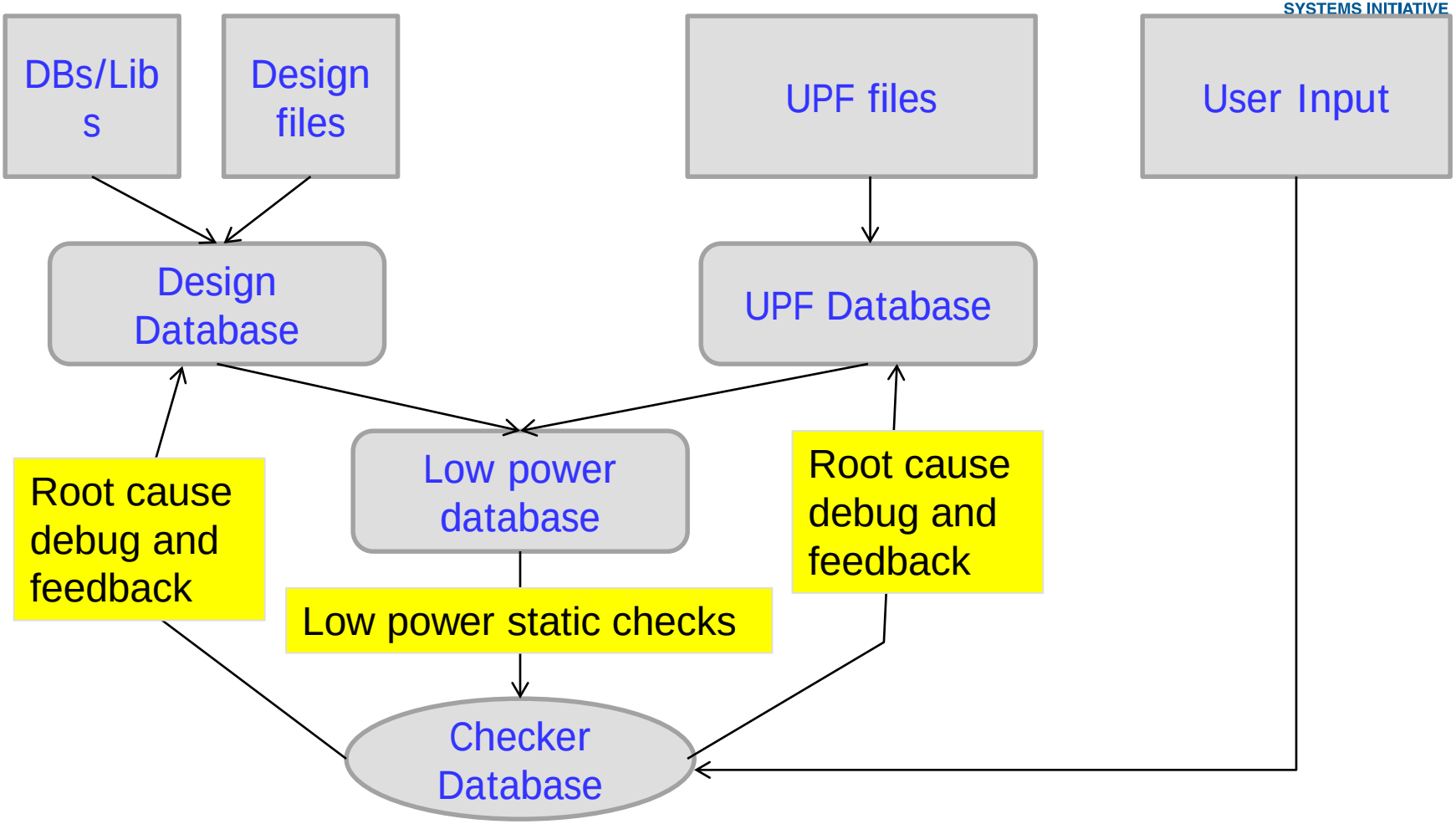
by
Bhaskar Pal
R & D Engineer
Synopsys

Overview:

The outline of the presentation

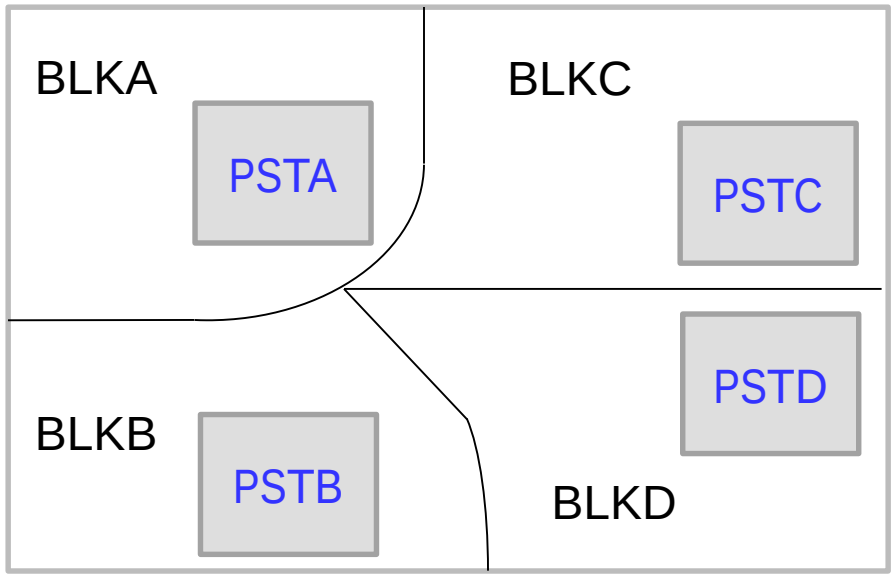
- Low power static checker flow
- Power State Table (PST) and its significance in the checker flow
- Completeness and consistency of PSTs
- Root cause debugging of incomplete and inconsistent PSTs
- Tool flow
- Results
- Conclusion

Low Power Static Checker Flow



Power State Table (PST)

- Usually a large design is partitioned into multiple power domains/blocks, each having multiple power states
- Power states are defined by voltage levels
- The PSTs of a block defines the valid power states and transitions of a block in terms of a set of supply ports/nets



Power State Table (PST) Contd..

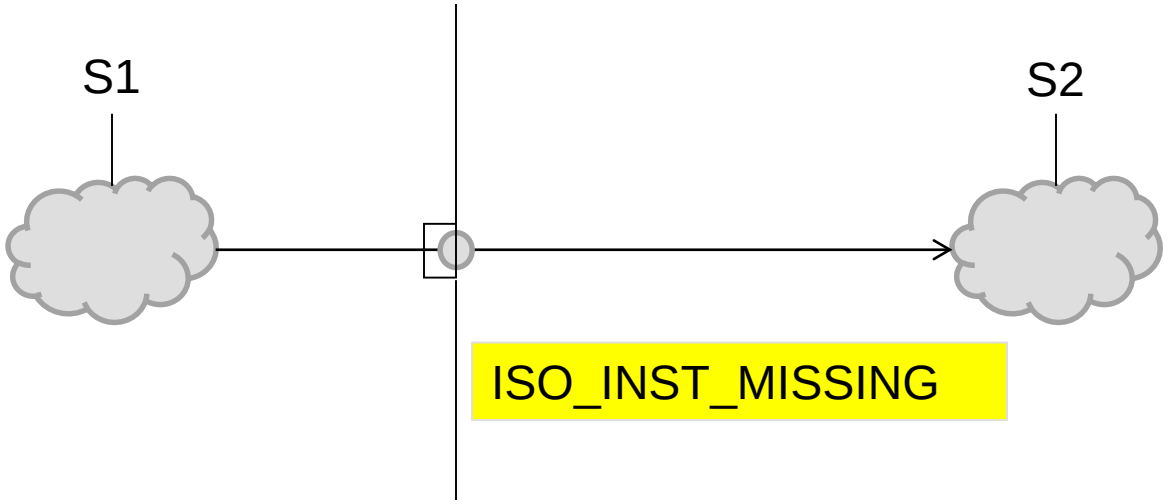
- The power/port states of supply net/ports are defined in terms of voltage levels
- A PST uses these port/power states to define a set of *supply relations* represented by the rows of the PST

PSTA		
	S1	S2
A11	off	ON
A12	ON	ON
A13	off	off

```
add_port_state S1 –state {ON 0.9} –state {OFF off}
add_port_state S2 –state {ON 0.9} –state {OFF off}
```

PSTs are golden to the checker

- Typically the low power static checker takes PST specification as golden and perform the electrical correctness checks on the design
  - Isolation requirement checks
  - Level shifter requirement checks, etc



PSTA		
	S1	S2
A11	off	ON
A12	ON	ON
A13	off	off

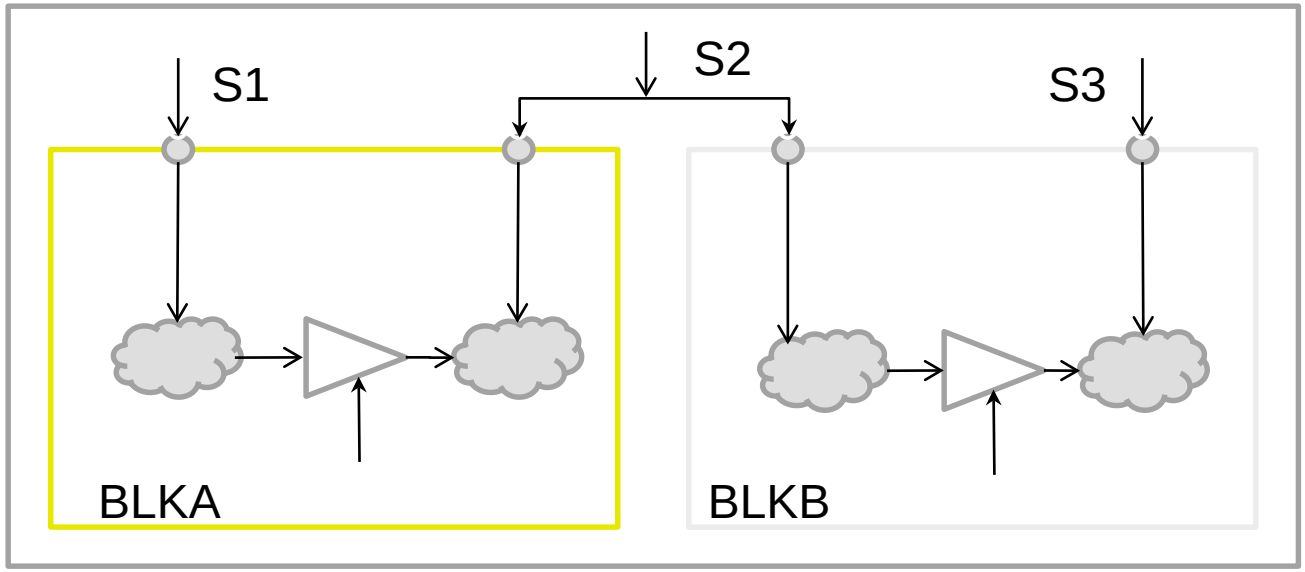
Global Power State Table and block level PSTs

- Ideally the global PST should represent states/transitions defined by conjunction of all the block level (local) PSTs.
- However, in reality the following scenario can happen
  - Designer can provide a global PST which can constrain the block level PSTs
  - A block level PST can constrain the global PST
  - A block level PST can constrain another block level PST
  - Merging of block level PSTs can introduce new *supply relations*
- The number of PSTs in design are becoming very large

Global Power State Table and block level PSTs

- Ideally the global PST should represent states/transitions defined by conjunction of all the block level (local) PSTs.
- However, in reality the following scenario can happen
  - Designer can provide a global PST which can constrain the block level PSTs
  - A block level PST can constrain the global PST
  - A block level PST can constrain another block level PST
  - Merging of block level PSTs can introduce new *supply relations*
- The number of PSTs in design are becoming very large

Inconsistencies in Block level PSTs



PSTA		
	S1	S2
P11	off	ON
P12	ON	ON
P13	off	off

PSTB		
	S2	S3
P21	off	ON
P22	off	off

Incomplete specification of port/power states in one block level PST can eliminate *supply relations* in another PST

Inconsistencies between Block level PST and Global PST

PSTG					
	S1	S2	S3	S4	S5
G11	ON	ON	ON1	ON	ON
G12	off	off	off	off	ON
G13	off	off	ON	off	ON
G14	off	off	off	off	off

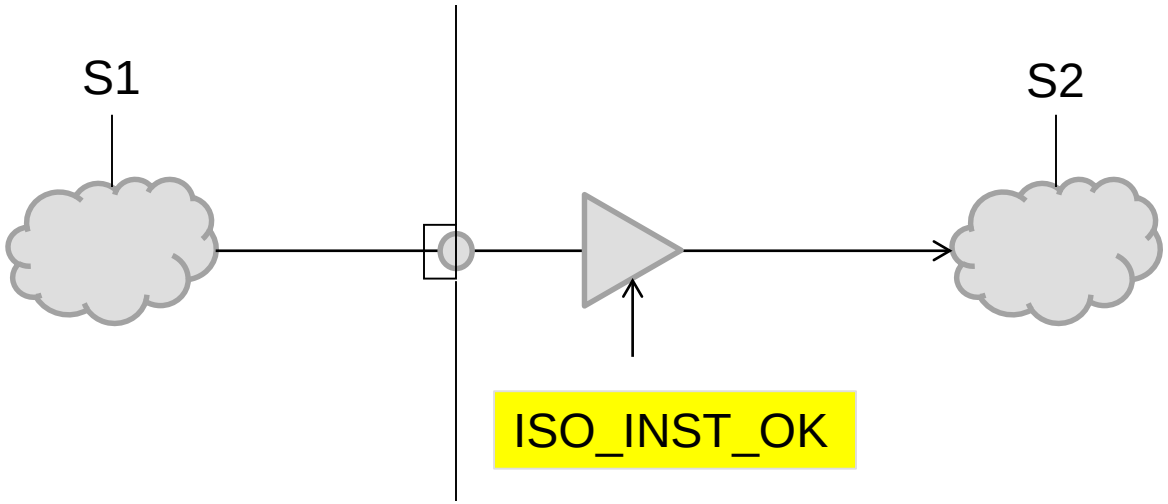
PSTA		
	S1	S2
P11	off	ON
P12	ON	ON
P13	off	off

PSTB		
	S2	S3
P21	off	ON
P22	ON	off

The global PST can override/constrain the block level PSTs resulting in elimination of *supply relations*

Violation debug issues with inconsistent PSTs

- Violation report generated during block level verification can change during System level verification making root cause analysis difficult
- A ISO_INST_OK violation in block level verification can be changed to ISO_INST_REDUND during system level verification

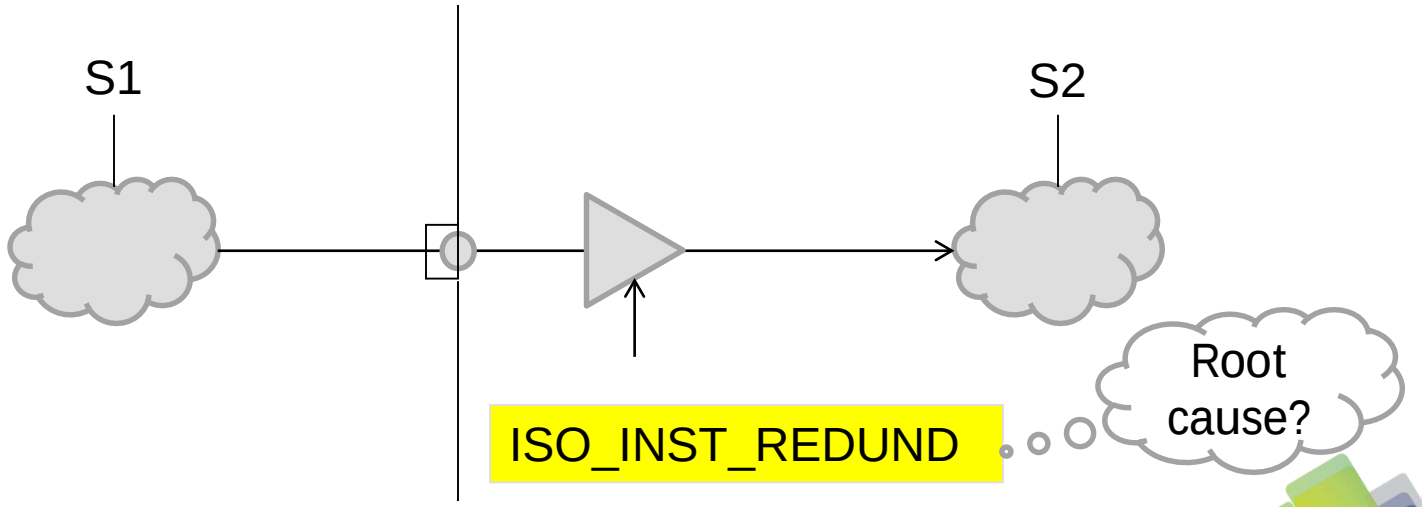


PSTA		
	S1	S2
A11	off	ON
A12	ON	ON
A13	off	off

Violation debug issues with inconsistent PSTs Contd...

PSTA		
	S1	S2
P11	off	ON
P12	ON	ON
P13	off	off

PSTG					
	S1	S2	S3	S4	S5
G11	ON	ON	ON1	ON	ON
G12	off	off	off	off	ON
G13	off	off	ON	off	ON
G14	off	off	off	off	off

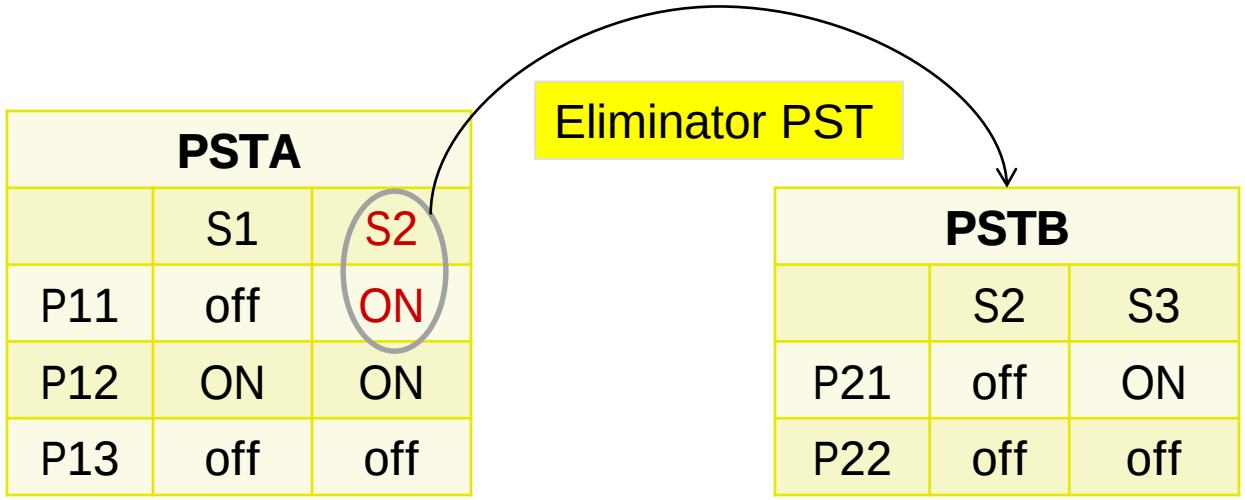


Contribution

- The large number of PSTs and specification of incomplete/inconsistent PSTs can lead to results which are difficult to debug
  - Framework for detecting and pre-compute database storing possible incompleteness/inconsistencies between PSTs
    - This can be achieved in a hierarchical manner
  - Linking violation database to the above pre-computed for root cause debugging

Eliminator PST

- A PST P is called an eliminator PST for a supplyStatePair (S, St) if supply S appears in P but the port/power state St of S has not been used in P



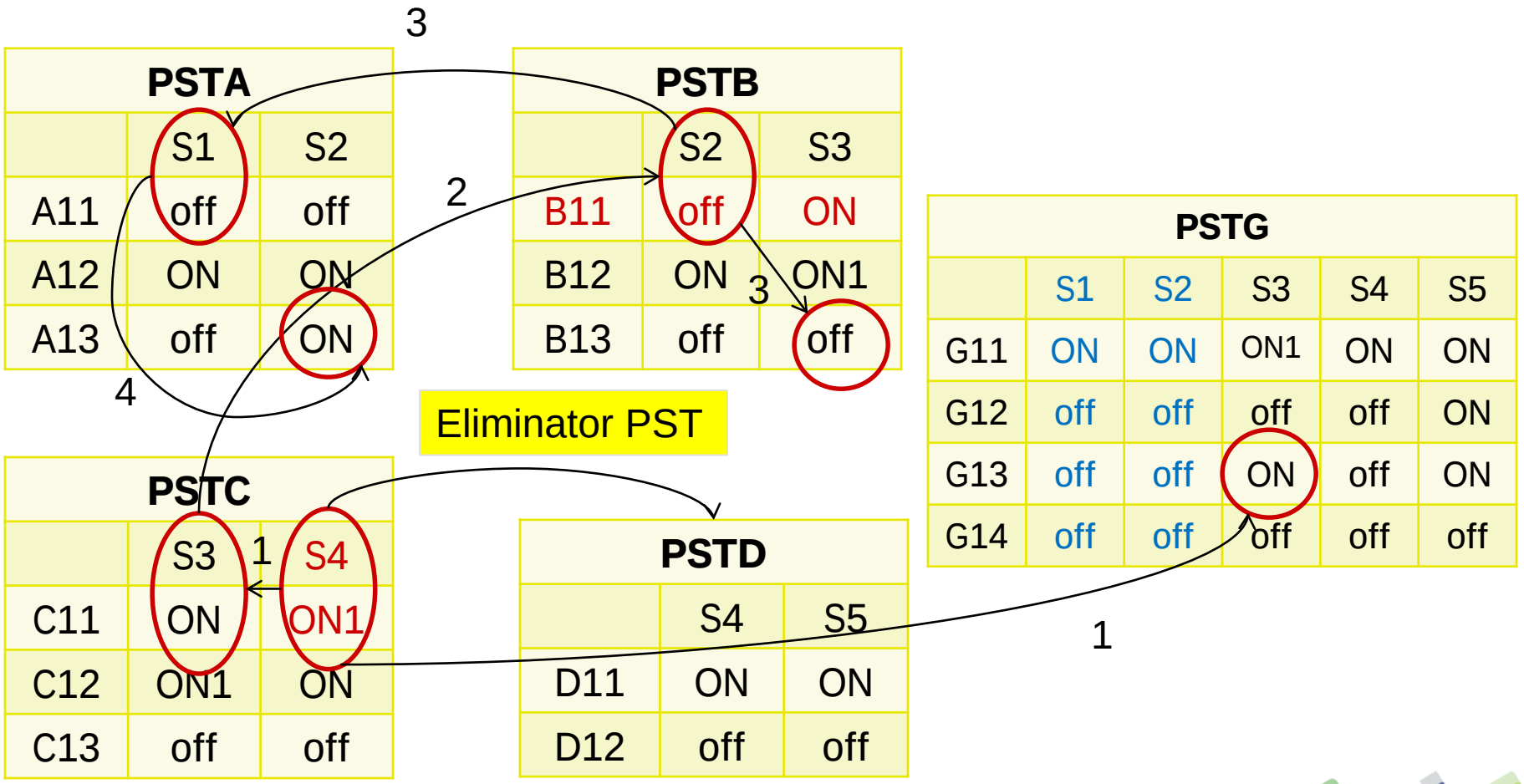
PSTA		
	S1	S2
P11	off	ON
P12	ON	ON
P13	off	off

PSTB		
	S2	S3
P21	off	ON
P22	off	off

- The supply relation {(S1, off), (S2, ON)} does not survive in the PST merge process because (S2, ON) has an eliminator PST and the eliminating sequence is PSTA → PSTB

Eliminator PST contd..

- Eliminating sequence may not be immediate



PSTA

	S1	S2
A11	off	off
A12	ON	ON
A13	off	ON

PSTB

	S2	S3
B11	off	ON
B12	ON	ON1
B13	off	off

PSTC

	S3	S4
C11	ON	ON1
C12	ON1	ON
C13	off	off

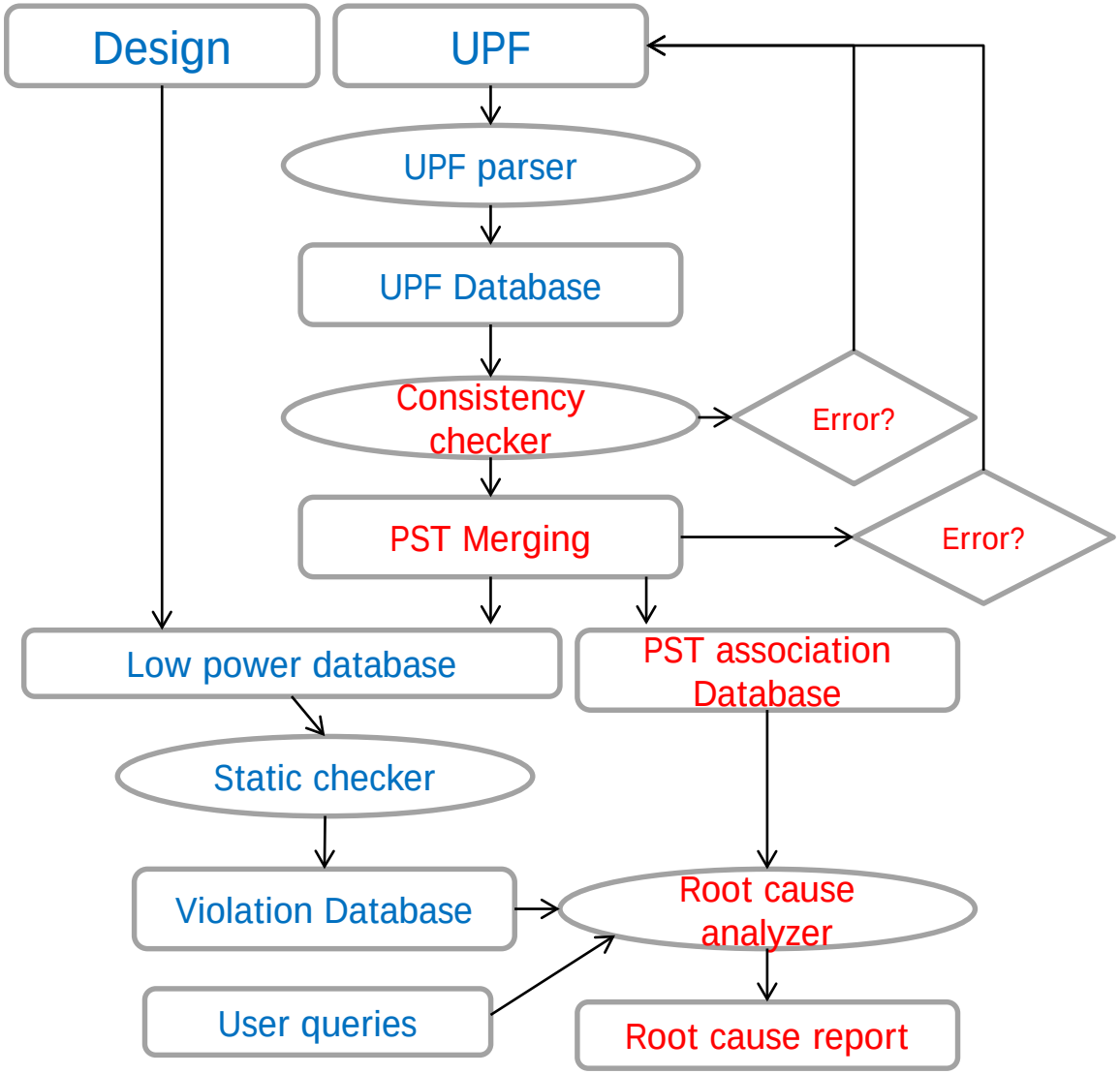
PSTD

	S4	S5
D11	ON	ON
D12	off	off

PSTG

	S1	S2	S3	S4	S5
G11	ON	ON	ON1	ON	ON
G12	off	off	off	off	ON
G13	off	off	ON	off	ON
G14	off	off	off	off	off

Tool flow



Results

Design	Size	Block level Issues	Chip level issues	Runtime overhead
Design-1	10 K	0	1	1 second
Design-2	2 M	2	2	< 5 seconds
Design-3	13.7 M	1	2	< 10 seconds
Design-4	45 M	1	5	< 1 min

Conclusion

- Developed a framework for detecting potential consistency/completeness issues between PSTs in a design
- Create a database of vulnerable PST states and their links with other states
- A checker that displays some of the critical PST issues
- A root cause analyzer that links checker violations that are triggered by some of the vulnerable PST states due to inconsistencies/incompleteness of PSTs