



Design & Verification Conference & Exhibition

**February 28 – March 3, 2011**

# Low Power Static Verification: Beyond Linting and Corruption Semantics

By

Kaustav Guha, Ankush Bagotra, Neha Bajaj

R&D Engineer, R&D Engineer, CAE

Synopsys India

The logo for Synopsys, featuring the word "SYNOPSYS" in a bold, blue, sans-serif font with a registered trademark symbol (®) at the end.



# Overview

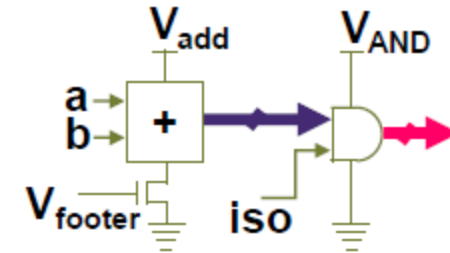
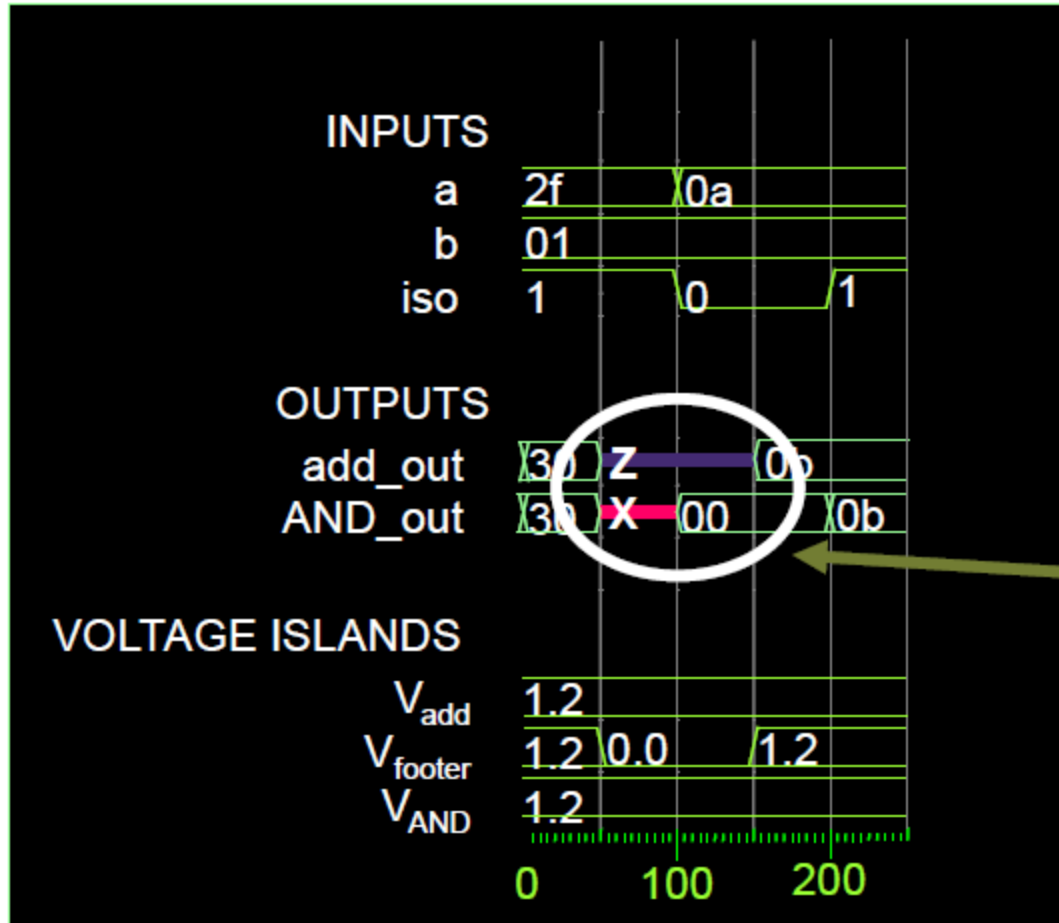
- Verifying a Low Power Design
- Problems with Verifying Low Power Design
- Defining "*Static*" in Static Verification
- High Level Low Power Verification (HLLPV) Framework
- Results

# Verifying a Low Power Design



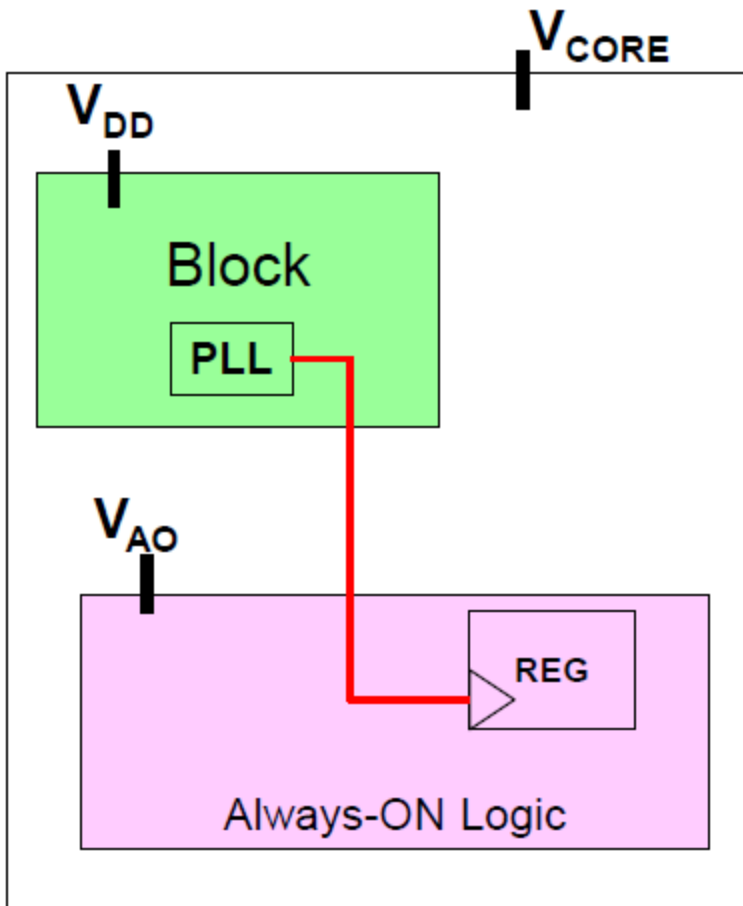
- The *Corruption Semantics* specified explicitly or implicitly through UPF triggers `X`'s or `Z`'s on signal paths whose supply is turned OFF unless protected by *Isolation, Level Shifter, Retention, and Power Switch* policies
- As a result, existing functional assertions/checkers fail.
- Simulator generates additional assertions to direct attention to a low power behavior
- Static checkers compare the design with structural templates called rules

# Verifying a Low Power Design-Simulation



Power gating sequence error caught

# Verifying Low Power Design-Static



Power State Table

STATE	$V_{AO}$	$V_{DD}$	$V_{CORE}$
OFF	0.0	0.0	0.0
BOOT	1.0	0.0	0.0
AON	1.0	1.0	0.0
ON	1.0	1.0	1.0

In "Boot" state, clock from shutdown block would be driving powered up register in always-on logic

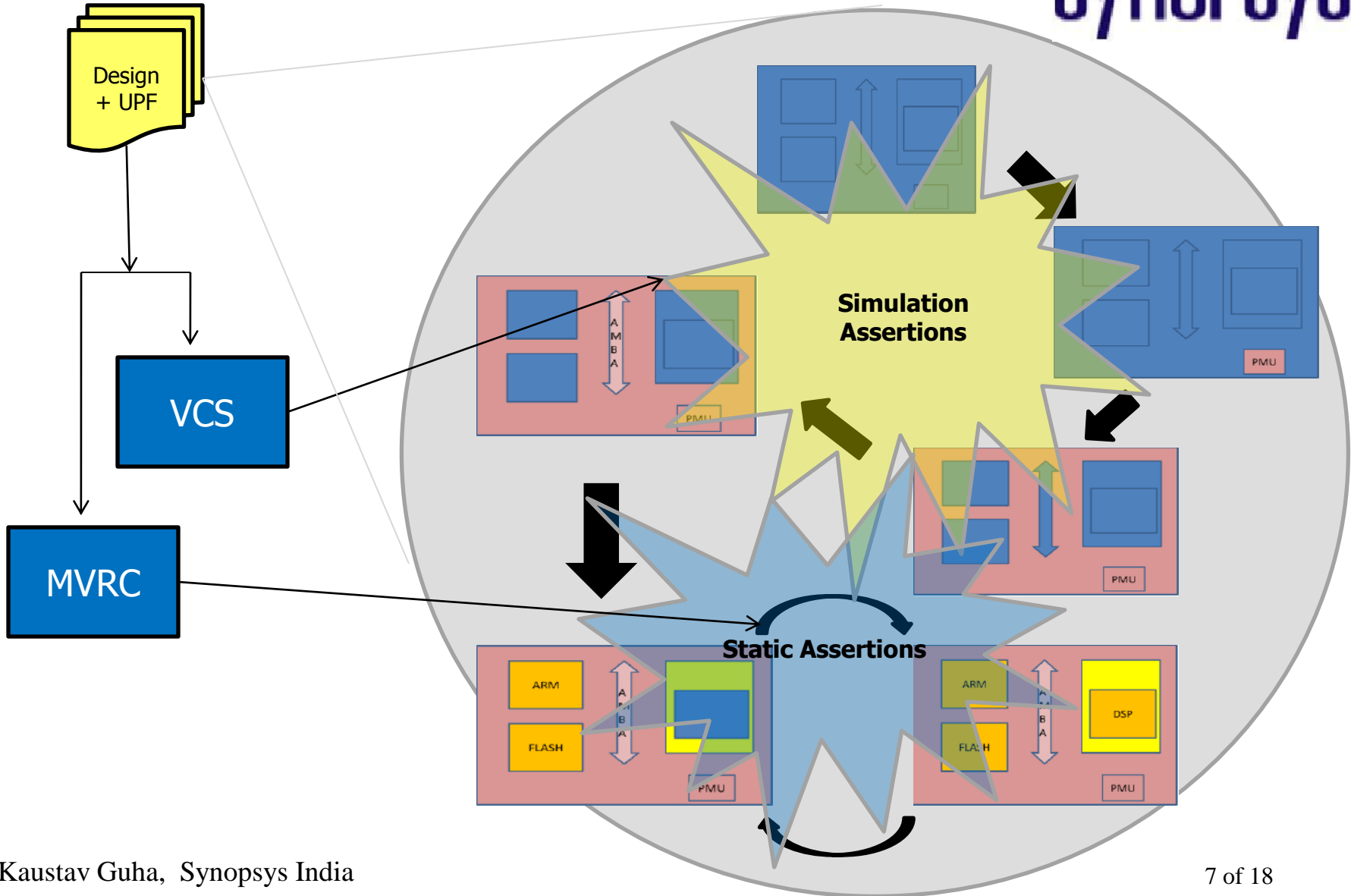


**Relative always-on error automatically flagged by MVRC**

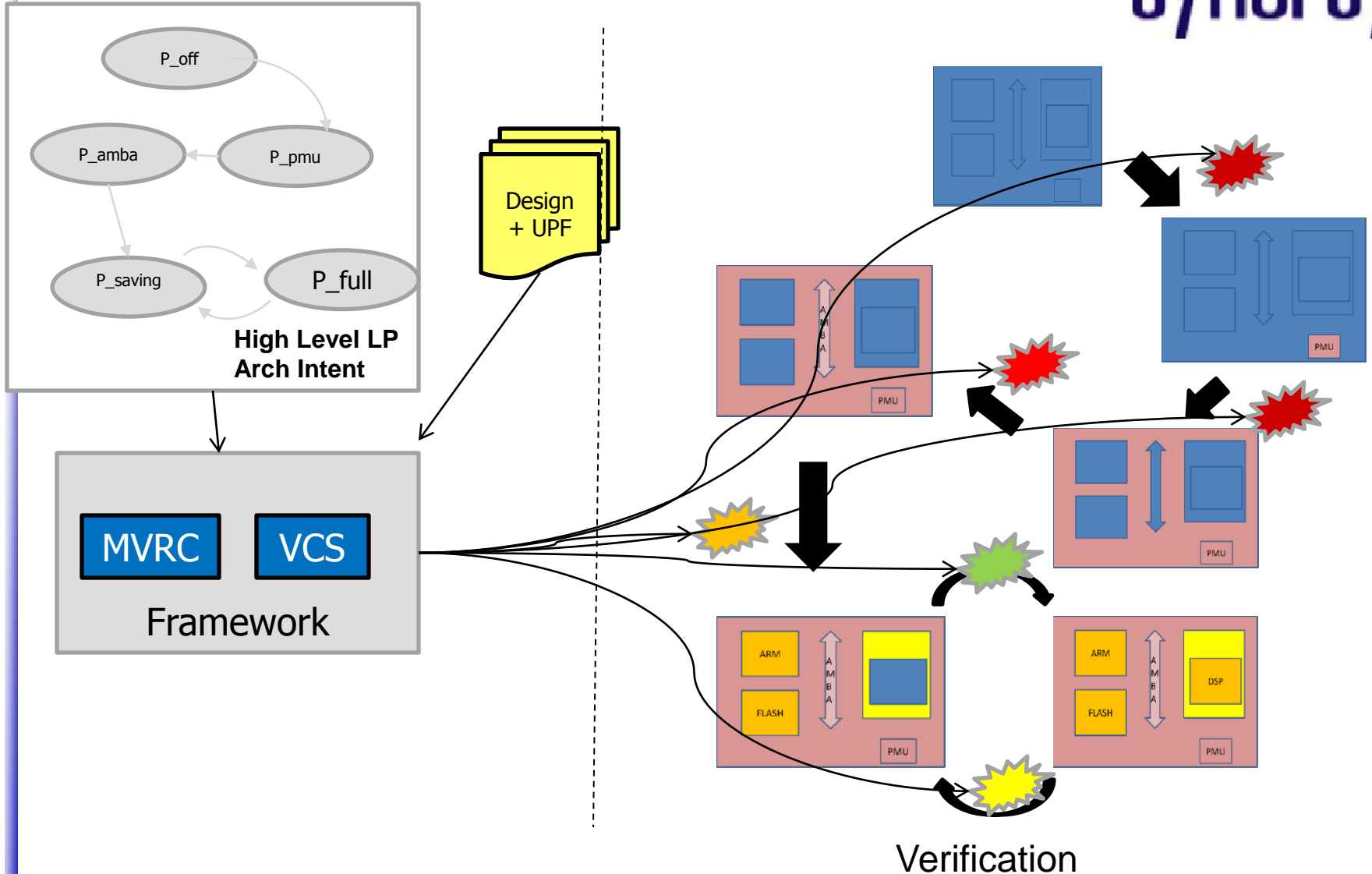
# Problems with Verifying Low Power Design

- *Corruption Semantics* will uncover low power issues however, debugging the root cause can be a problem
- Static rules and *Corruption Semantics* addresses very low design details rather than an architecture wide intent
- Consequence- Huge set of failing assertions and static rules hit verification productivity
- Deciding when to stop verifying the design becomes difficult

# Low Power Verification



# Low Power Verification-What If?

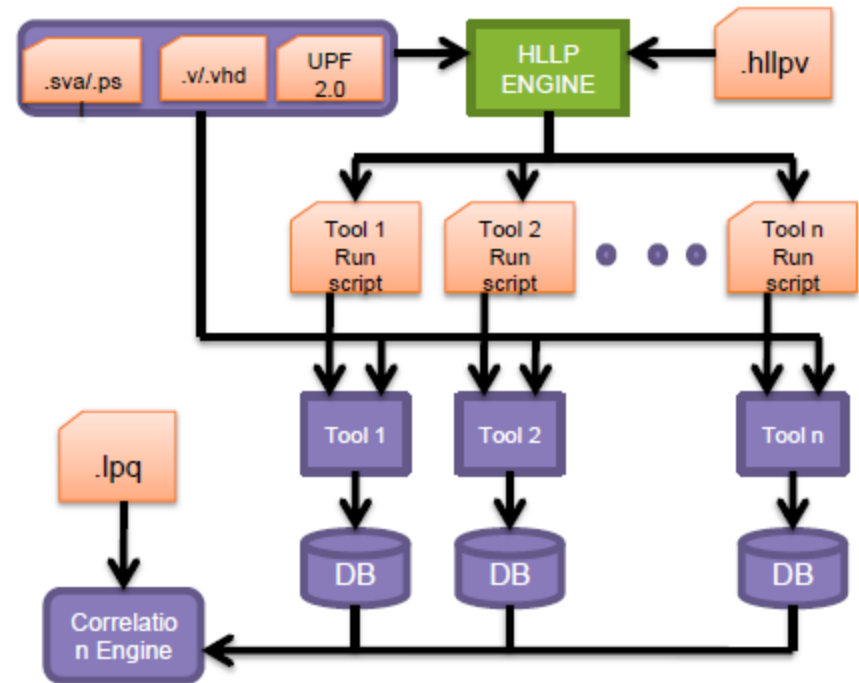


# Defining “Static” in Static Verification

*Static Verification is a verification of all specified desirable invariant properties and absence of all specified undesirable invariant properties in both the structure and functionality of the design.*

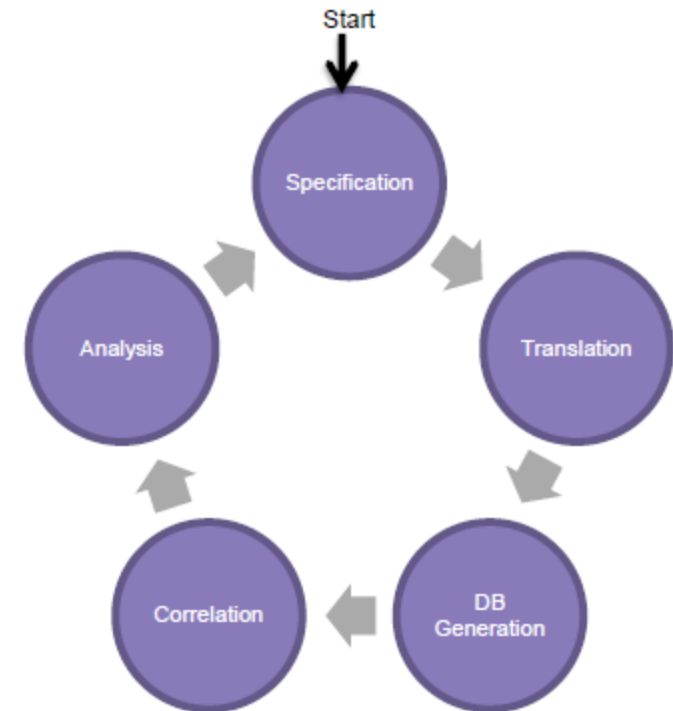
# HLLPV Framework

- Defines semantics to address low power verification flows and to enable non low power verification flows into power-aware ones.
- Brings together assertions from HDL specification, simulation tool, power states, and the static rules of different point tools to ensure that the low power flow is complete.
- User needs to specify a “Verification Objective” in terms of logical, low power states, and the static rules.

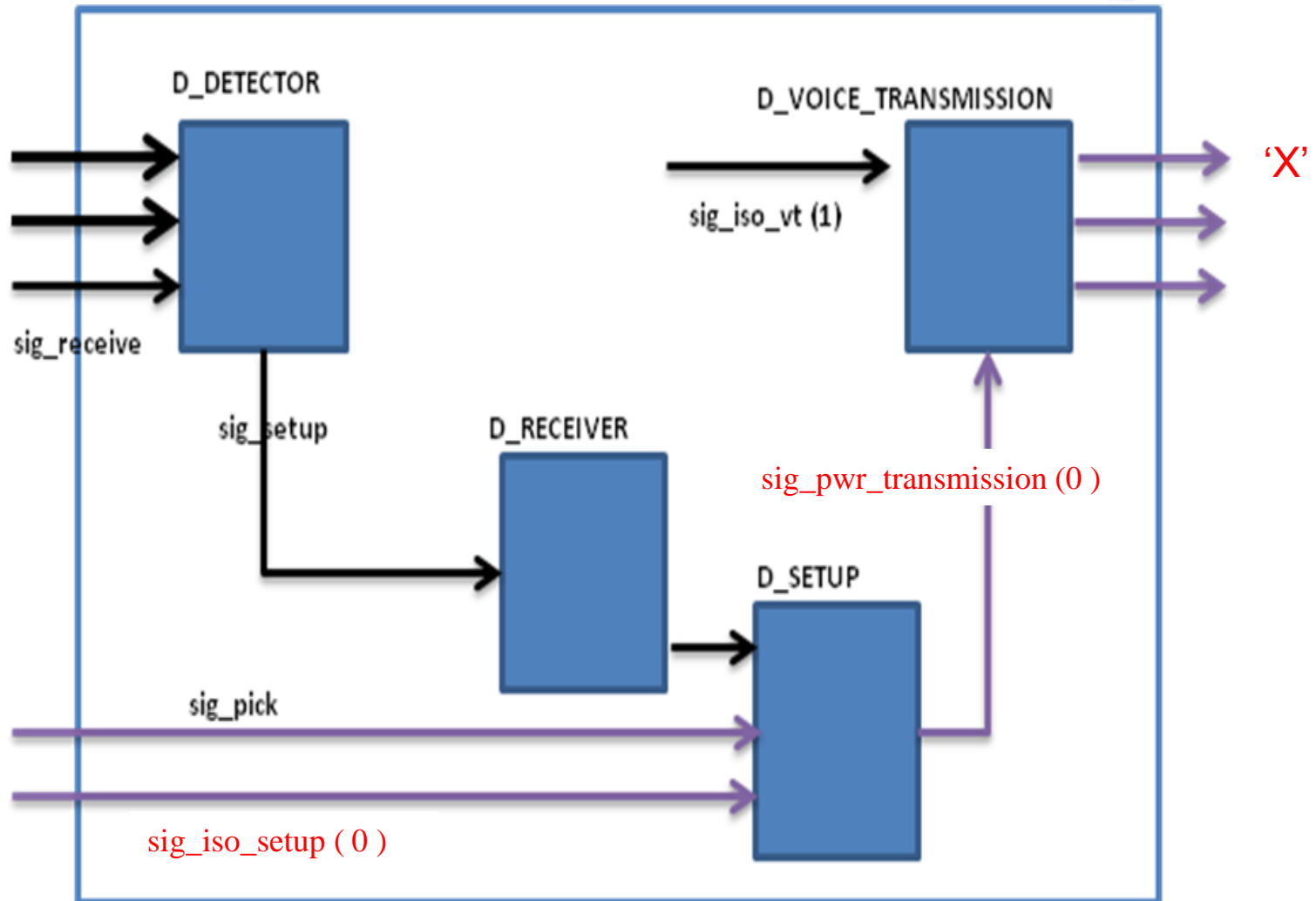


# HLLPV Stages

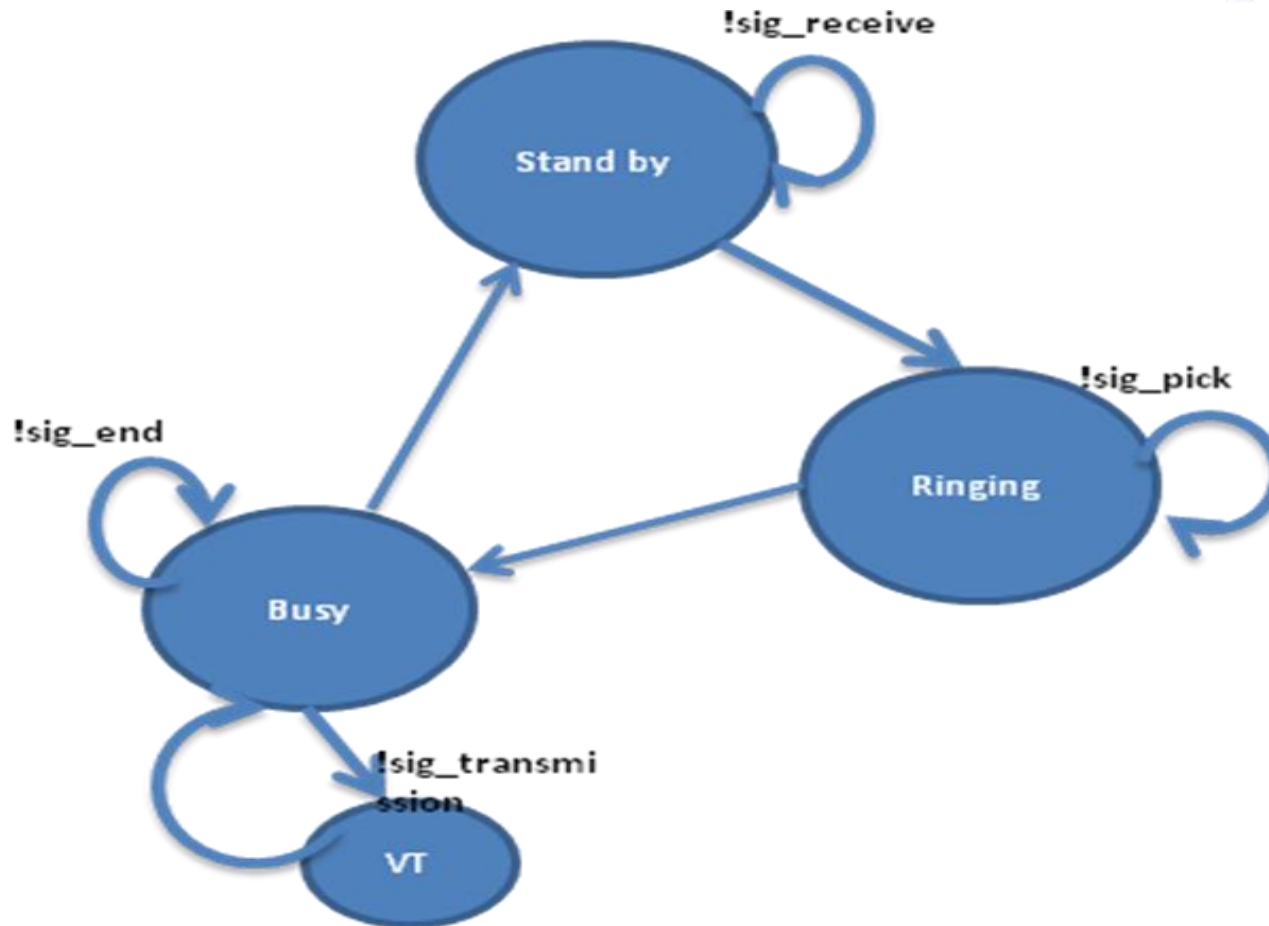
- **Specification:** Defining the verification objectives (.hllpv)
- **Translation:** Translating .hllpv into customized scripts for point tools to run customized checks
- **DB Generation:** Dumping Reports in SQL compliant format
- **Correlation:** Correlating results per say for a verification objective
- **Analysis:** Do what-if analysis, analyze the root cause and query the coverage metrics



# Case Study



# Design State Machines



# Power operation state machine

SYNOPSYS®

HLLPV snippets for the case study

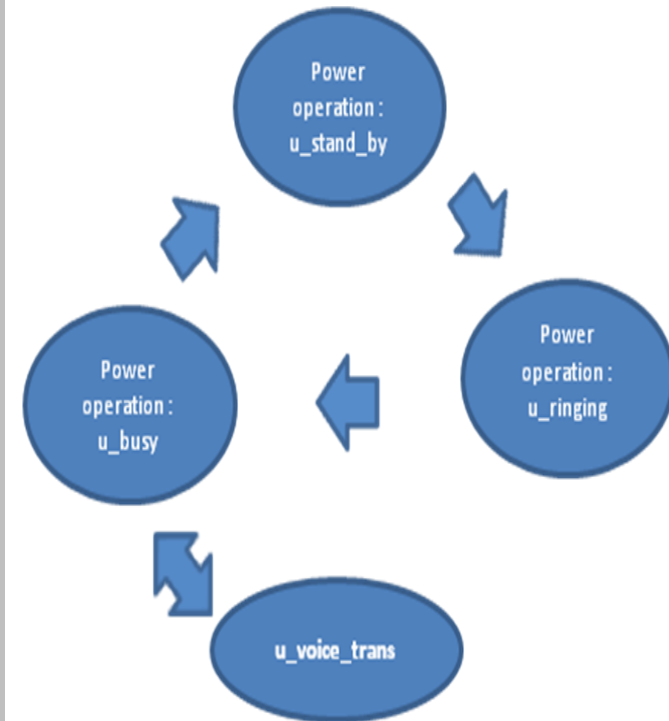
```

power_operation u_standby
  apply_when -domain_state {!D_RECEIVER}
  on_definition {D_RECEIVER.SLEEP_MODE}
  complete_upon { D_RECEIVER.ON_MODE}
end_power_operation

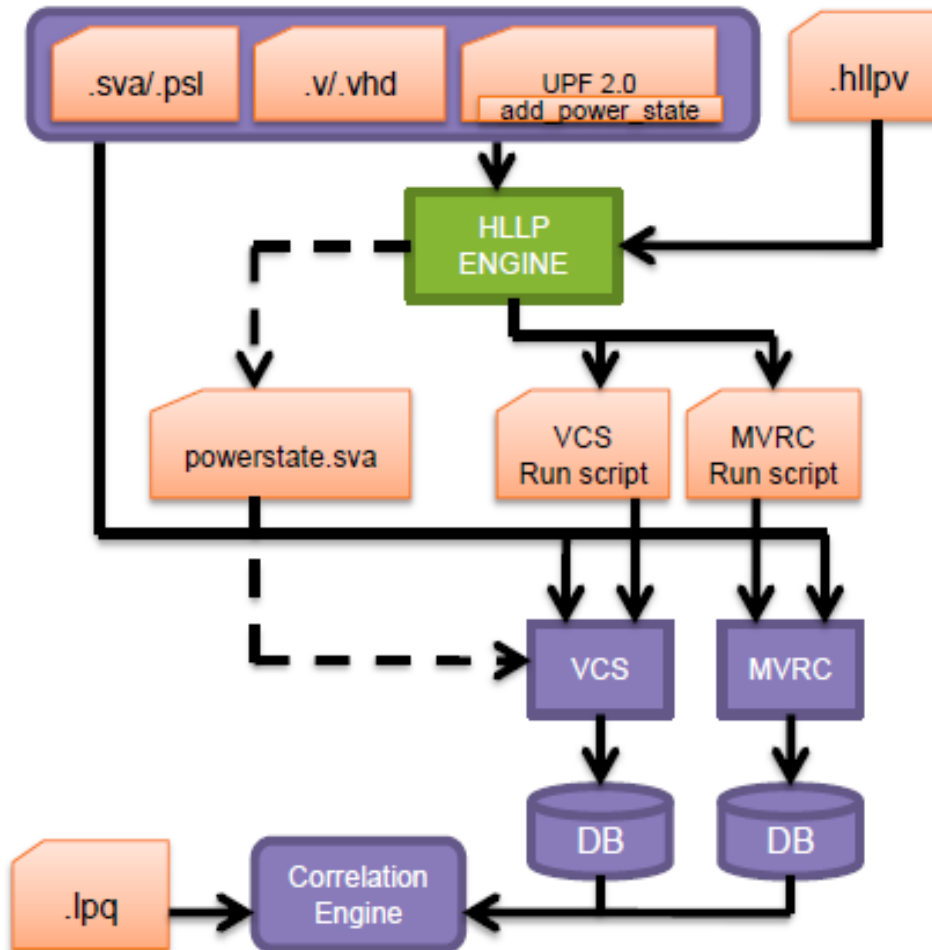
power_operation u_ringing
  apply_when -operation_completed u_standby
  on_definition {D_RECEIVER.ON_MODE}
  property_after {D_SETUP.ON_MODE, D_RECEIVER.ON_MODE}
end_power_operation

power_operation u_busy
  apply_when -operation_completed u_ringing -domain_state
{D_VOICE_TRANSMISSION.ON_MODE}
  property_after { D_VOICE_TRANSMISSION.ON_MODE, D_SETUP.RUN_MODE}
  assert_on_rule -rule {WRONG_SENSE} -tool MVRC
  user_property_when -state {D_VOICE_TRANSMISSION.ON_MODE} -checker
vcs_checker.sva -bind_to {D_VOICE_TRANSMISSION}
end_power_operation

```



# Case Study: The Flow



# Results



<b>Design</b>	<b># Domain</b>	<b># Total Issues in the Design</b>	<b># Assertions fired by MVRC</b>	<b># Assertions fired in assertions in VCS</b>	<b># Assertions fired by HLLPV</b>
Ref_design1	5	37	10	20	7
Ref_deisgn2	9	63	21	30	12

**Thank You**

**Questions?**