A Formal Approach for X-PROP and Low Power Verification-based Connectivity Verification for Control-intensive Static Design Trees

Jayashri Patil, Manu Yeeshu
(SoC Design Verification Engineers)
Intel India Pvt. Ltd., Bangalore

Problem Statement

- Low-power RTL simulations do not accurately predict the power consumption of the design if the simulation models used are not accurate.

- Switching power domains in presence of incorrect clock gating can be catastrophic

- The entire network fails if there is even a single failure anywhere in it because of the combinational nature. No escape route or recovery center exists.

Solution Space

1) We propose a technique that combines X-PROP (Formal x-propagation) verification with integration verification
2) The standard connectivity assertions are verified while we do X-PROP verification for a power-aware design.
3) This X-PROP integration verification technique guarantees strict interface connectivity as per specification even during power state transitions.

Applications of formal Connectivity in debug domains

DFD — Design For Debug

To verify that the debugging components in the digital system are correctly connected.

DFT — Design For Test

To verify that the testing signals are correctly connected to the different components in the system.

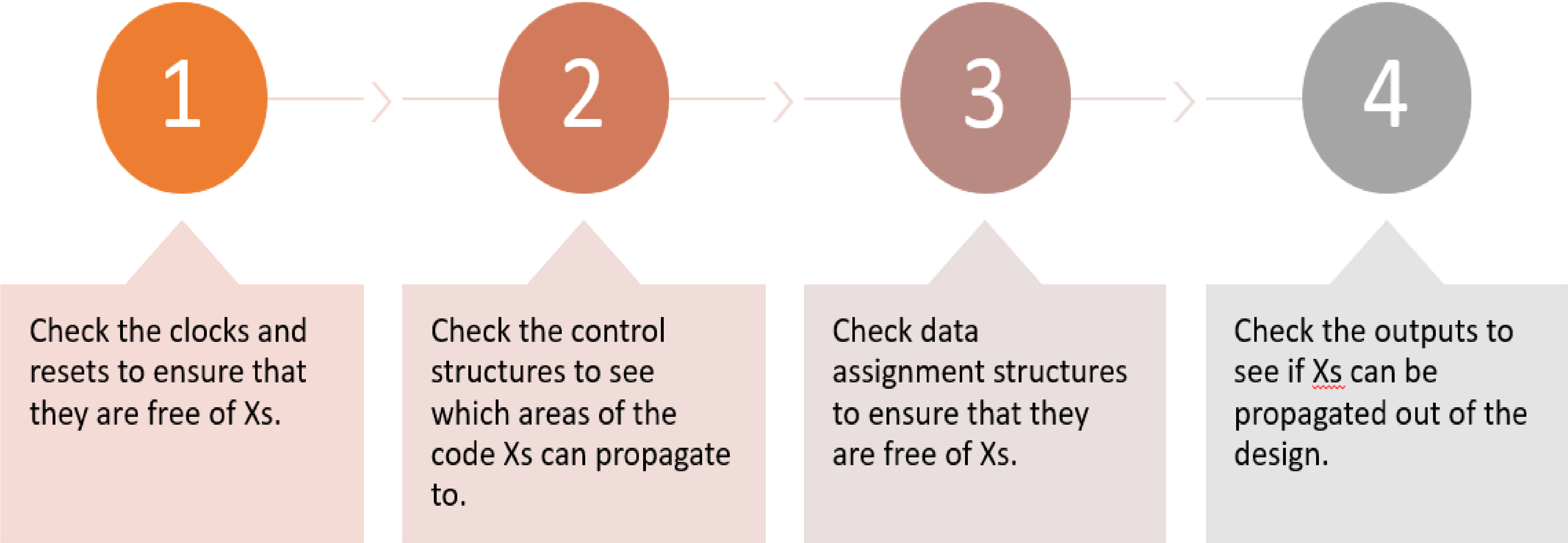
DFM — Design for Manufacturing

To verify that the connections between the different manufacturing components in the digital system are correct.

DFX — Design for Excellence

To verify that the optimization techniques used in the system are correctly connected to the different components in the system.

Uses of VC Formal X-propagation



1. Check the clocks and resets to ensure that they are free of Xs.
2. Check the control structures to see which areas of the code Xs can propagate to.
3. Check data assignment structures to ensure that they are free of Xs.
4. Check the outputs to see if Xs can be propagated out of the design.

Advantages of VC Formal X-propagation

The tool automatically generates properties on control logic, clock, and reset such that they are free from X issues. This step guarantees completeness.

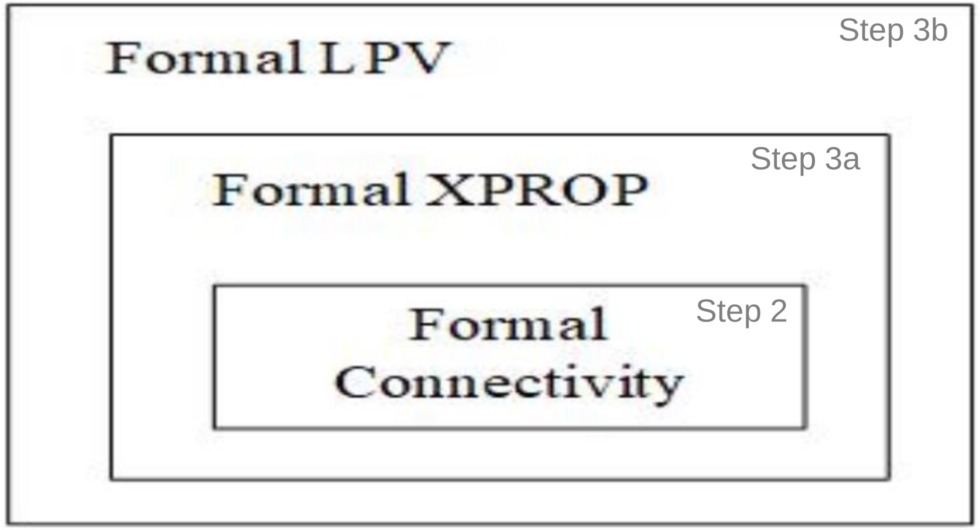
No need to create any complex testbench setup. Just need to pass the design for formal compilation and elaboration.

For the newly designed components, during the development stage itself, one can check if any multi-driver issues or floating ports are existing in RTL.

Since completeness is guaranteed, it is a good replacement for the Simulation-based XPROP approach, where all scenarios are difficult to generate.

Proposed Methodology

Hierarchical based bottom-up approach



Step 3b

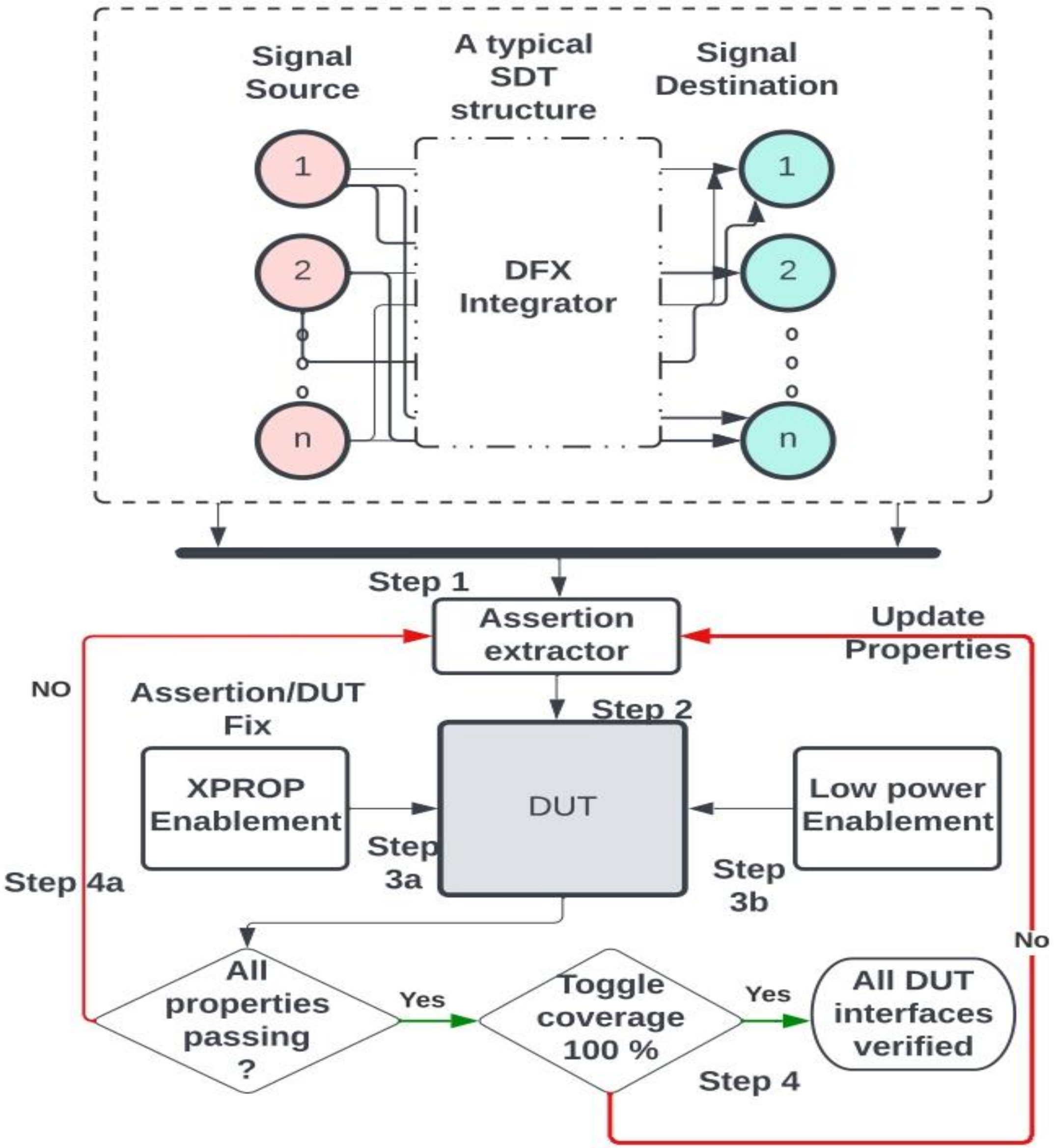
Formal LPV

Step 3a

Formal XPROP

Step 2

Formal Connectivity



Signal Source

A typical SDT structure

Signal Destination

1

2

n

DFX Integrator

1

2

n

Step 1

Assertion extractor

Update Properties

NO

Assertion/DUT Fix

Step 2

XPROP Enablement

DUT

Low power Enablement

Step 3a

Step 3b

Step 4a

No

All properties passing ?

Yes

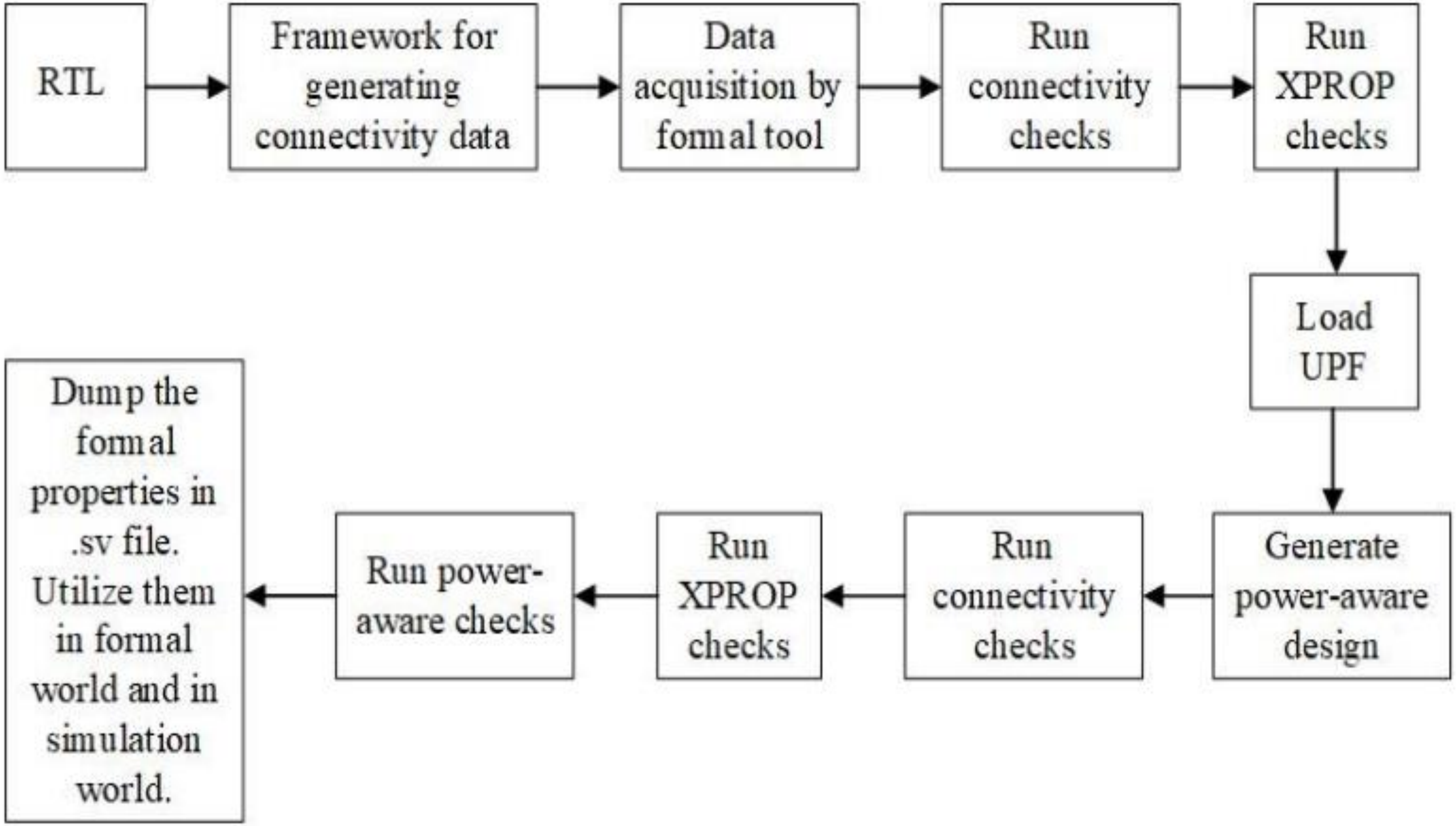
Toggle coverage 100 %

Yes

All DUT interfaces verified

Step 4

Framework for the Proposed Methodology



RTL

Framework for generating connectivity data

Data acquisition by formal tool

Run connectivity checks

Run XPROP checks

Load UPF

Generate power-aware design

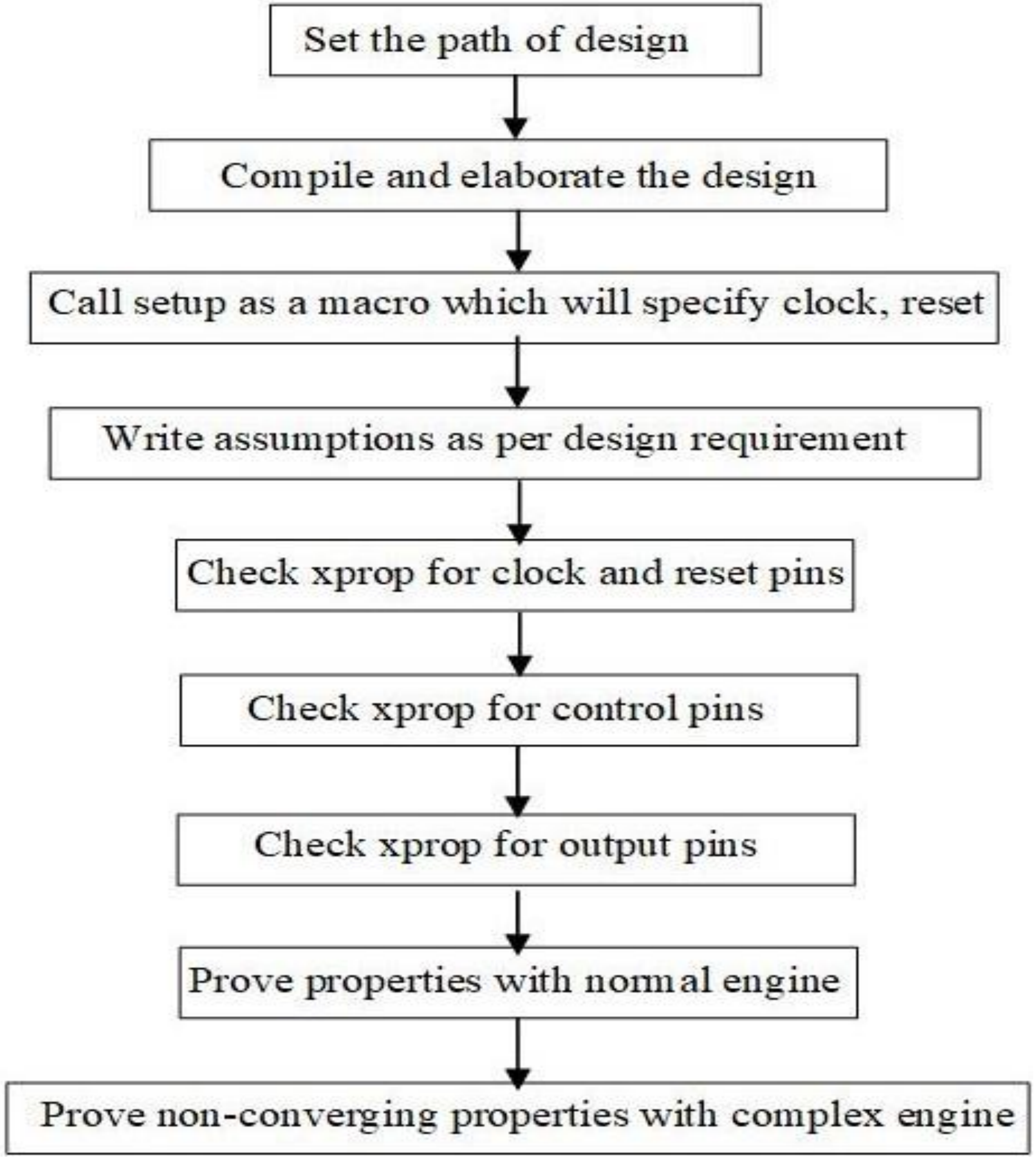
Run connectivity checks

Run XPROP checks

Run power-aware checks

Dump the formal properties in .sv file. Utilize them in formal world and in simulation world.

FV-based XPROP Automation Flow



Set the path of design

Compile and elaborate the design

Call setup as a macro which will specify clock, reset

Write assumptions as per design requirement

Check xprop for clock and reset pins

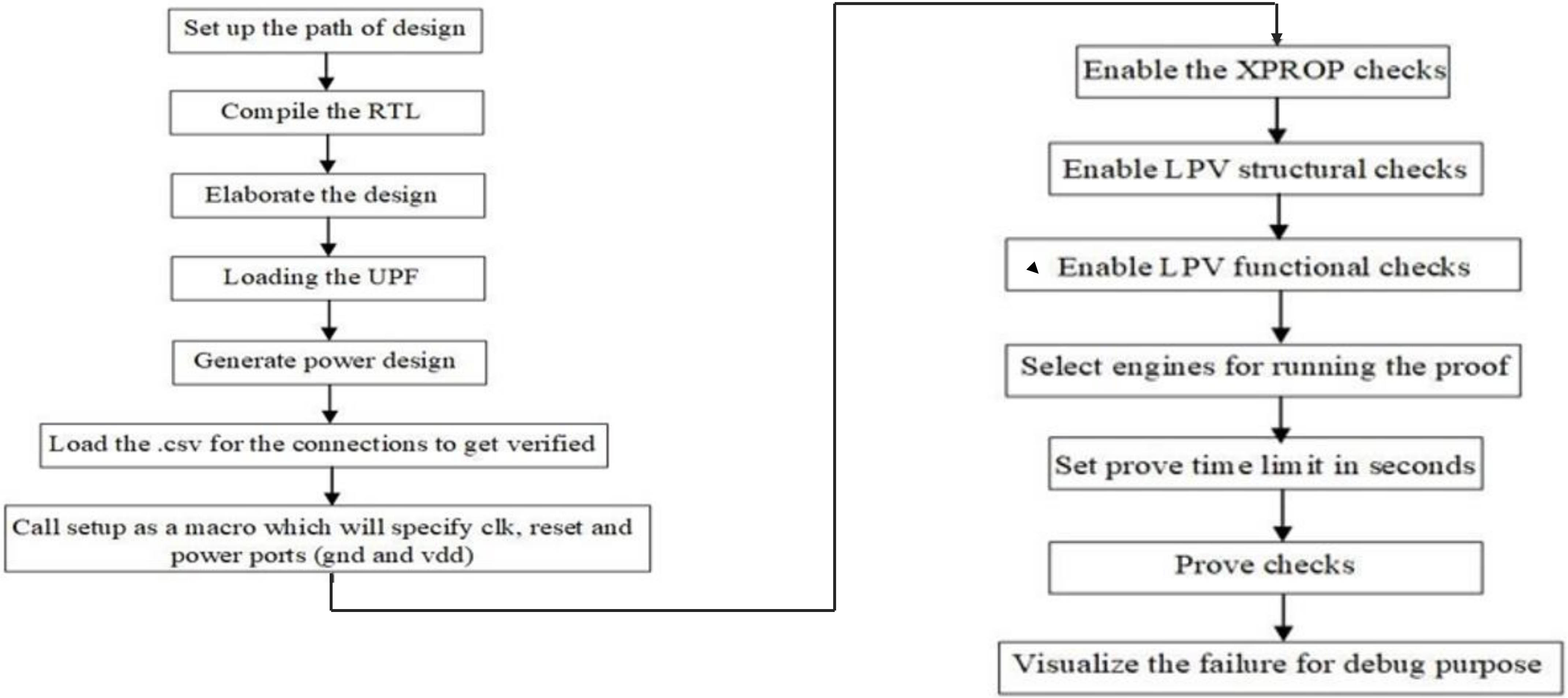
Check xprop for control pins

Check xprop for output pins

Prove properties with normal engine

Prove non-converging properties with complex engine

FV-based XPROP and LPV automation flow



Set up the path of design

Compile the RTL

Elaborate the design

Loading the UPF

Generate power design

Load the .csv for the connections to get verified

Call setup as a macro which will specify clk, reset and power ports (gnd and vdd)

Enable the XPROP checks

Enable LPV structural checks

Enable LPV functional checks

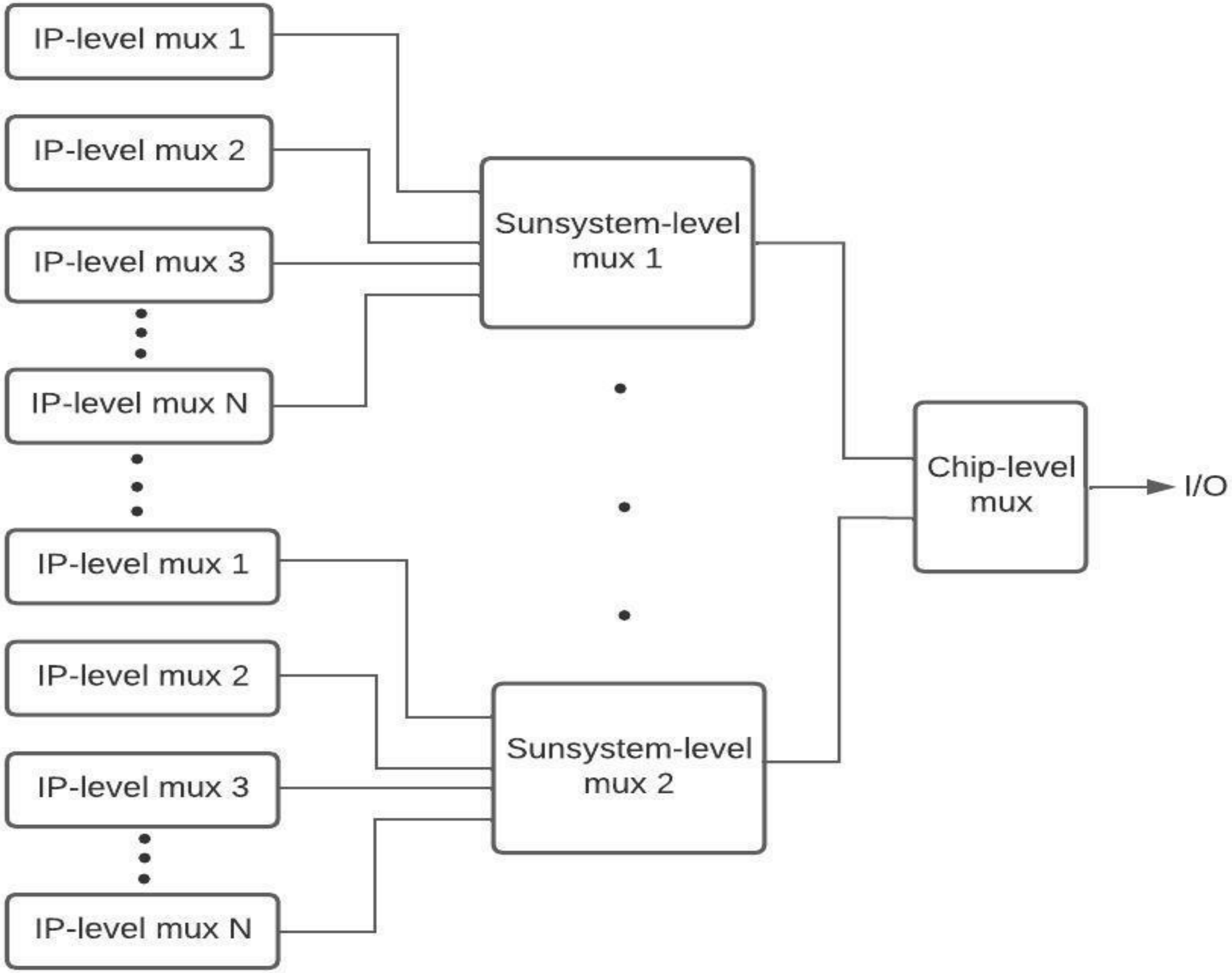
Select engines for running the proof

Set prove time limit in seconds

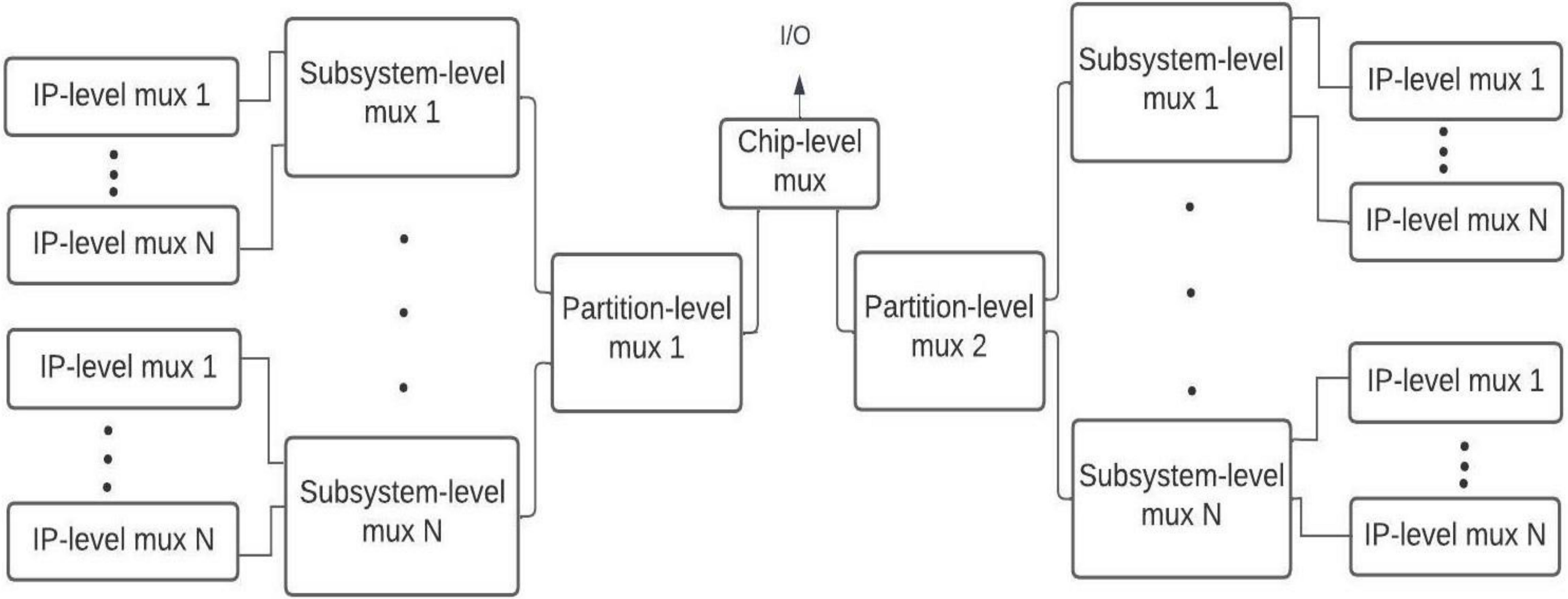
Prove checks

Visualize the failure for debug purpose

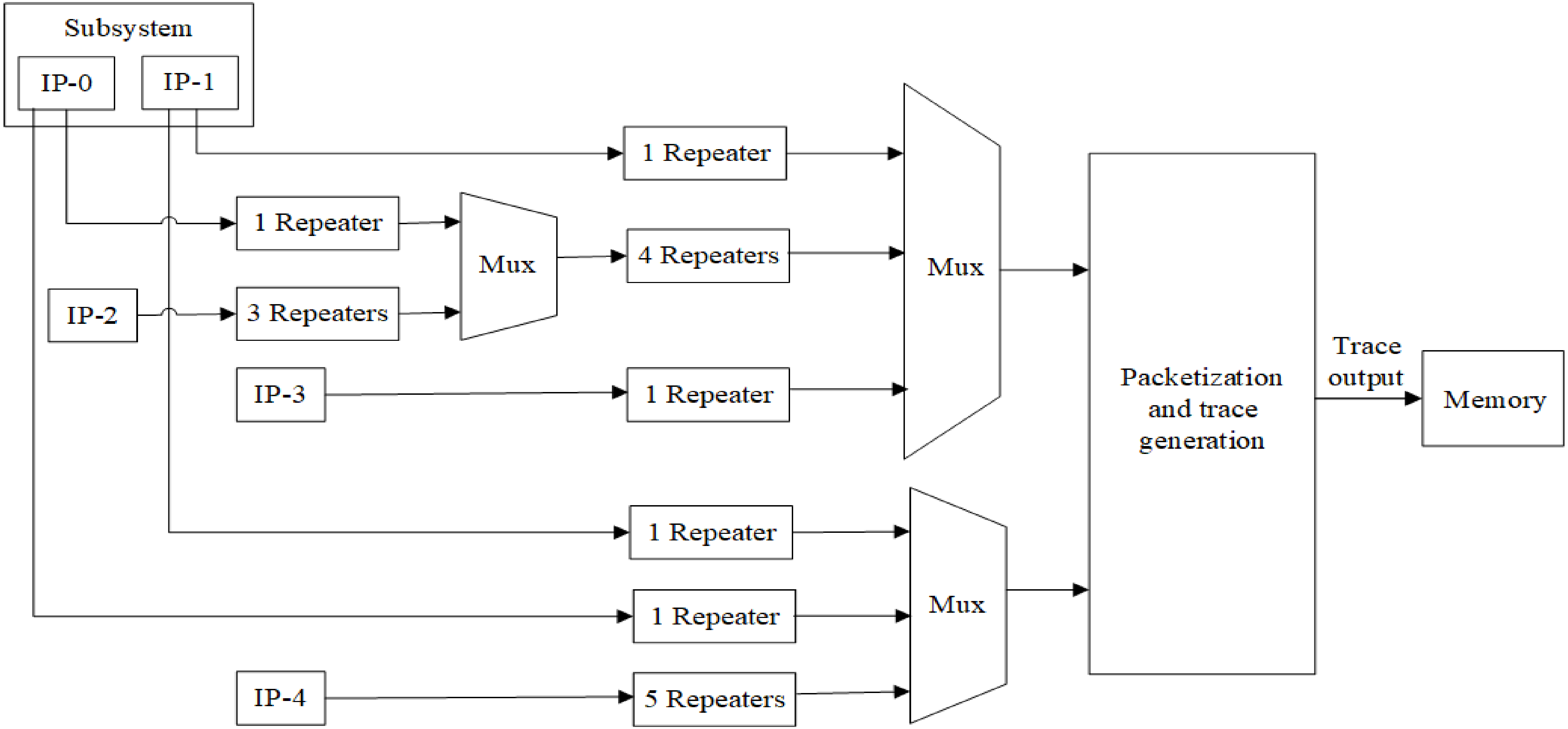
Observe-mux Tree Structure in the SoC



Partition-wise Observe-mux Tree Structure in the SoC



Observe-mux Tree Branches in the SoC along-with Repeaters for Debug Trace Generation

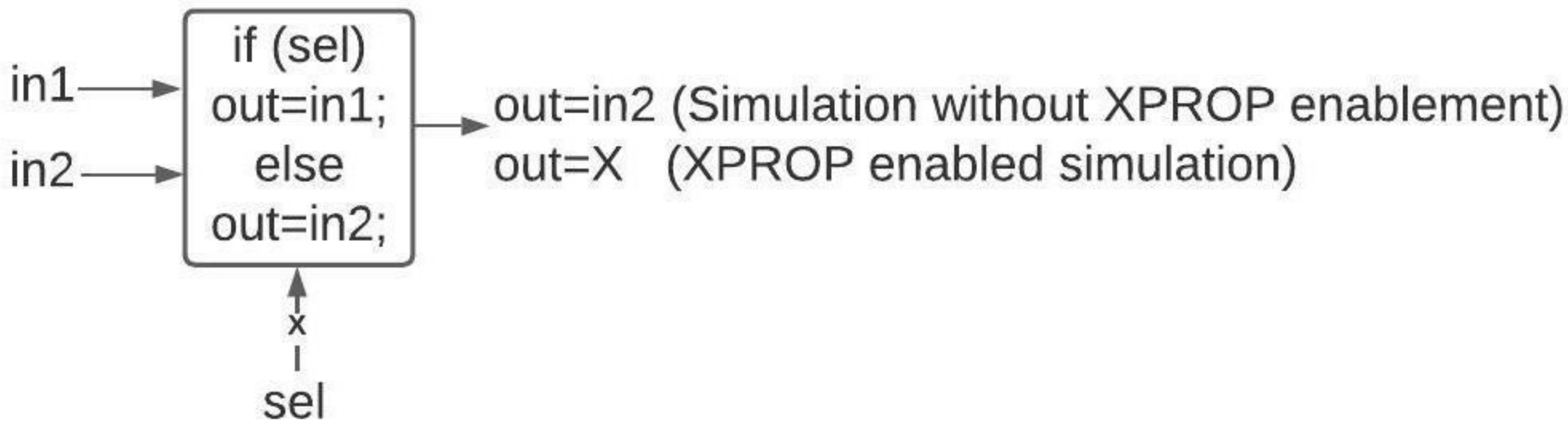


Results

X-PROP Related Bugs Detected

Output ’out’ is going ’X’ when X-PROP is enabled

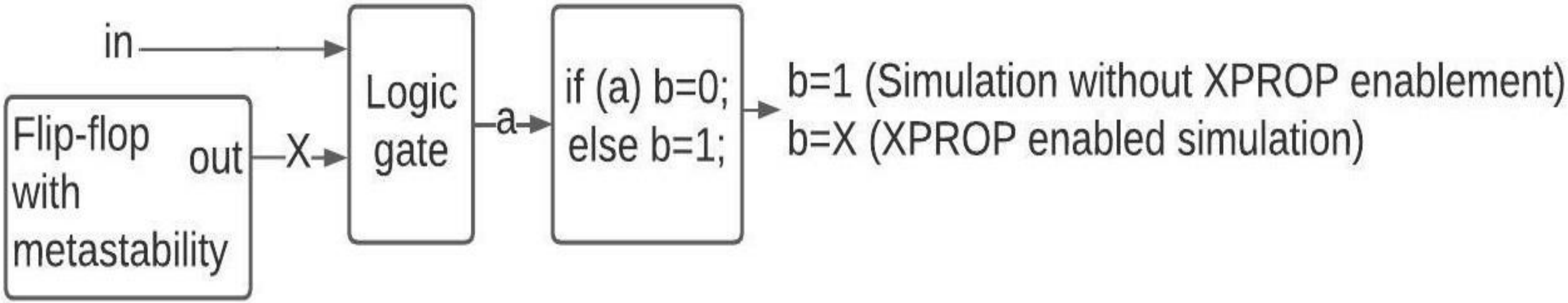
- The select line of one of the mux in the observe-mux tree was going ’X’ because of a corrupted driver.
- The output of the mux was going ’X’ with X-PROP enablement. Without X-PROP enablement, the failure was not observed because the output was ’in2’.


in1
in2
if (sel)
out=in1;
else
out=in2;
x
sel
out=in2 (Simulation without XPROP enablement)
out=X (XPROP enabled simulation)

X-PROP Related Bugs Detected

Output ’b’ going ’X’ when X-PROP is enabled

One of the flip-flops entered the meta-stability state. This led to one of the inputs of one of the logic resulting ’X’. In X-PROP-enabled formal verification, the output of logic has gone ’X’ since the input is ’X’. In the RTL simulation without X-PROP enablement, this output was ’1’.


in
Flip-flop with metastability
out
X
Logic gate
a
if (a) b=0;
else b=1;
b=1 (Simulation without XPROP enablement)
b=X (XPROP enabled simulation)

Turn-around Time Improvement with the Proposed Methodology

Category	Time for formal technique	Time for simulation technique	Gain
Connectivity	4hrs	20hrs	5X
Connectivity + XPROP	24hrs	50hrs	2X
Connectivity + XPROP + power-aware design	40hrs	84hrs	2.1X

Future Scope

Future work will deploy the proposed methodology, beginning with subsystems verification and moving on to whole SOC verification utilizing the divide-and-conquer method.

In the future, the proposed methodology will be applied to both the functional and debug paths.

The proposed methodology will also be used in server-based multi-die platforms to verify

- Die-to-tile connectivity
- Die-to-die connectivity
- Tile-to-tile connectivity

THANK YOU

Our Technology, **Your Innovation**™