# Quality and Confidence Improvement on OCP IP's using CDN OCP ABVIP

Aneet Agarwal, Mithun Ghosh
Texas Instruments

Lokesh Babu, Cadence

Session: Track I (Verification)

# Agenda

- Speaker Introduction
- Introduction to Problem
- Existing Solution
- Overview of New Solution (Formal Verification)
- About Incisive Formal Verifier (IFV)
- About CDN OCP Verification IP (VIP)
- Results
- Case Study
- Conclusion

# Speaker Introduction

- Name : Aneet Agarwal
- Company Name : Texas Instruments
- Division : WTBU- India
- Experience : About 06 years of experience in the field of design verification and silicon validation
- Role : Formal Verification Lead
- Responsibility : FV of OMAP (Application Processor) IP's

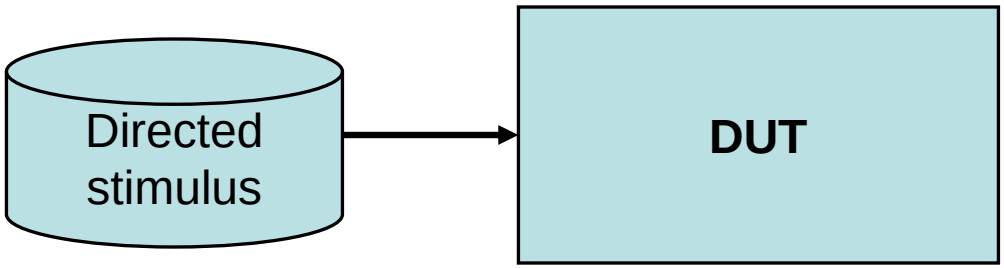
# Introduction to Problem

- Most of the IP's used in our Soc's transact via Open Core Protocol (OCP) interface as on-chip-bus interface.
- These IP's have OCP Master, Slave or both transaction ports.
- We really need to check the **OCP protocol compliance** for all OCP IP's before integrating them with others.
- Schedule to complete the verification is short.
- Confidence on exiting methodology is not high**.**

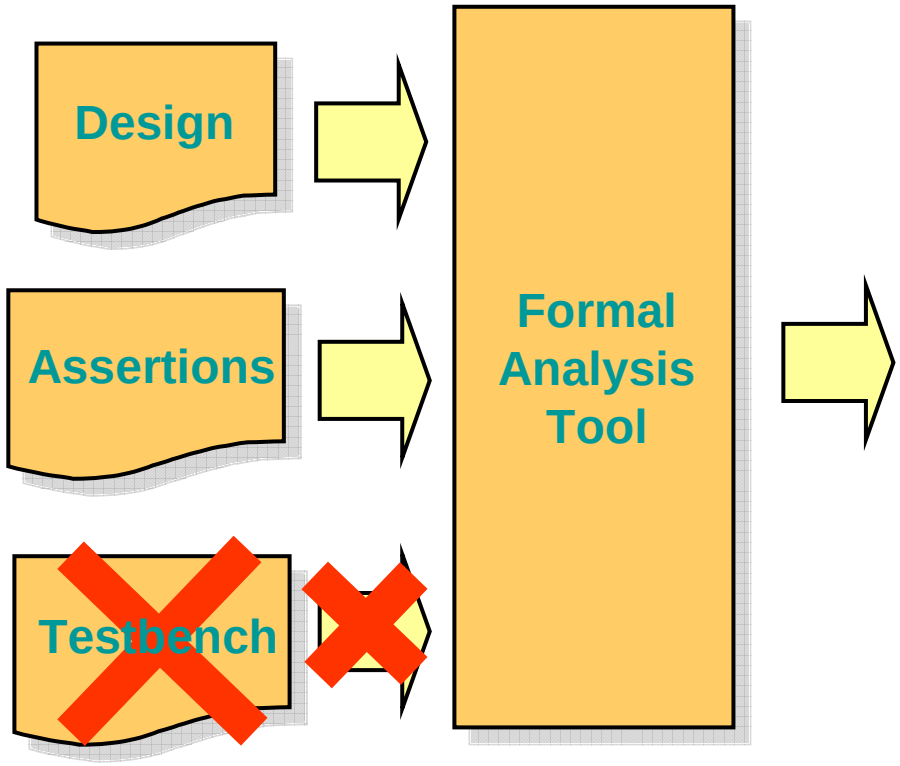
- Need a plug and play setup which can give complete protocol coverage and confidence on the IP's with in a short time.

# Existing Flow: Dynamic Verification

- Drawbacks
  - Need some effort in setting up the environment.
  - Depends on the quality of the test bench.
  - Limited controllability.
  - Debugging time is more.
  - Primary concerns are time & reusability.
  - Completeness.

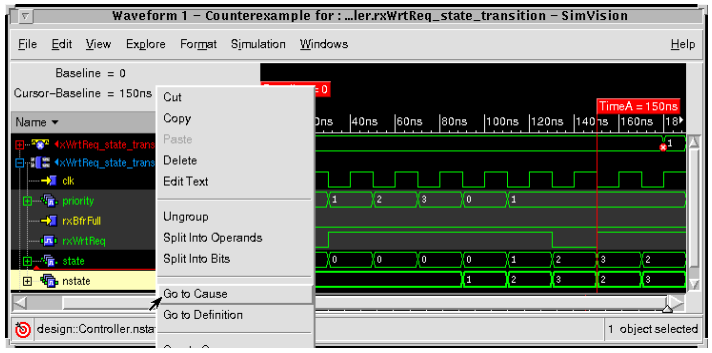


# Overview of new solution: Formal Verification (1)




Assertion holds for all possible inputs

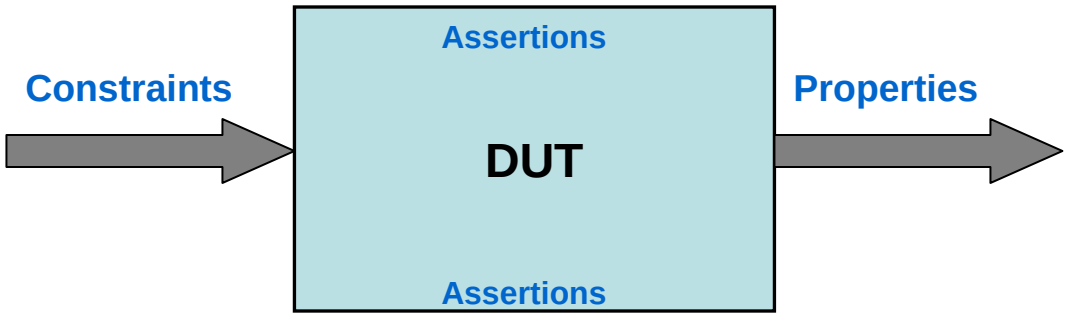
Assertion failed for at least one input



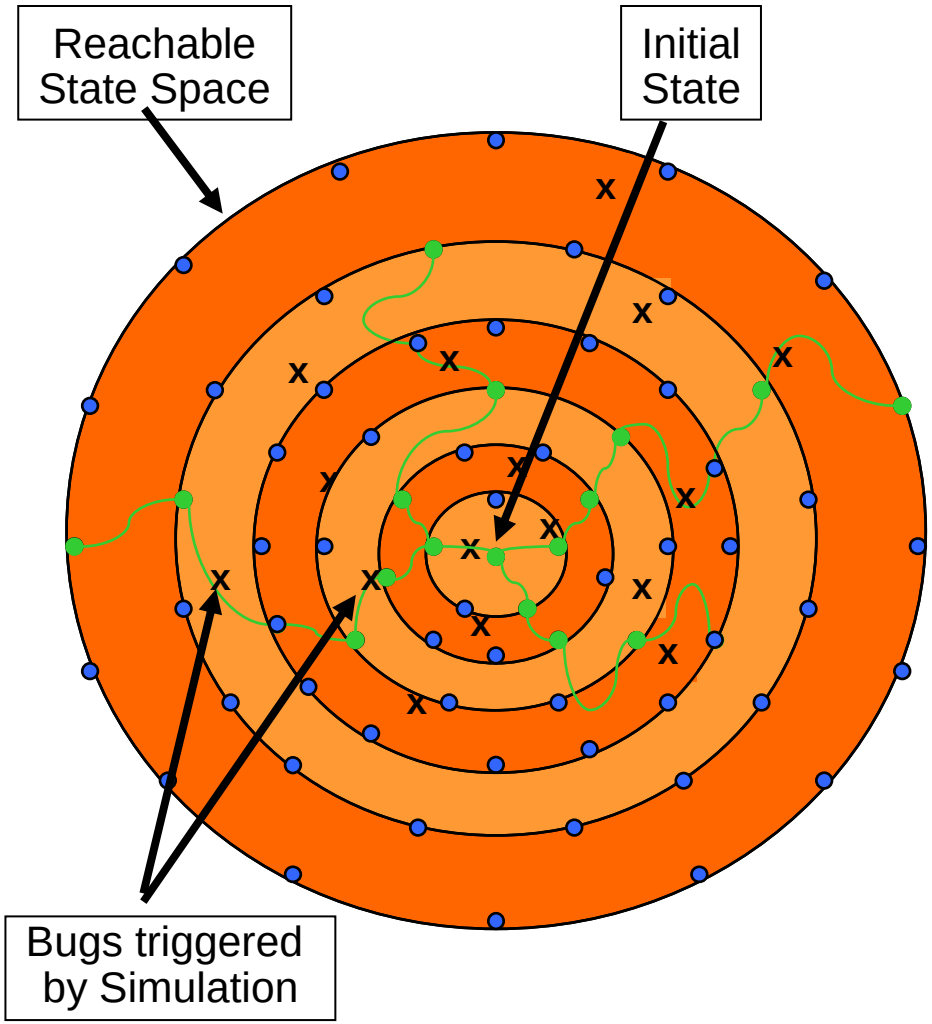

Debug environment for failed assertions

# Overview of new solution: Formal Verification (2)

- Advantages
  - No testbench required.
  - Formal reaches each and every state of design, Quality is more
  - Less debugging effort/time.
  - Same Assertions can be re-used in dynamic and emulation.
  - Gives completeness.



# Difference between Formal and Dynamic Verification



# All About Incisive Formal Verifier (IFV)

- Best-in-Class Formal Engines
  - Broad set of complementary engines
  - Automation strategy engine for ease of use
  - Advanced algorithms for experienced users
  - Sophisticated Abstraction Automation
- Broad Language Support
  - Verilog, VHDL, SystemVerilog, mixed-language
  - SVA and PSL Assertion Languages
  - OVL and IAL Assertion Libraries
- Production Proven Methodology
  - Adoptable and usable by design teams
  - Adaptable and usable by verification teams
  - Scalable and reusable flow
  - Multi-application
- Ease of Adoption
  - Complete and robust diagnosis environment
  - Usability features
  - Synergy with simulation

# Inbuilt Checks in IFV

## RTL Checks

- In built HAL
- Model checking
- DFT/structural checks
- User defined checks

## Pragma Checks

- All synthesis pragma's are supported
- 1-hot,full/parallel case .. etc
- Helps in finding RTL and NETLIST mismatches early in the design cycle

## Dead code Checks

- Reach ability analysis on every line
- Gives vector for reachable code
- **Redundant logic can be found**

## FSM Checks

- Dead lock checks.
- Reachable and transition checks.

## Bus & "X" Checks

- Multi drives and contentions on bus.
- "X" reach ability analysis

# Cadence OCP ABVIP- Key Concepts

- ABVIP
  - Assertion based verification IP.
  - Exhaustive Properties library
  - PSL + Verilog
  - Pre-verified.

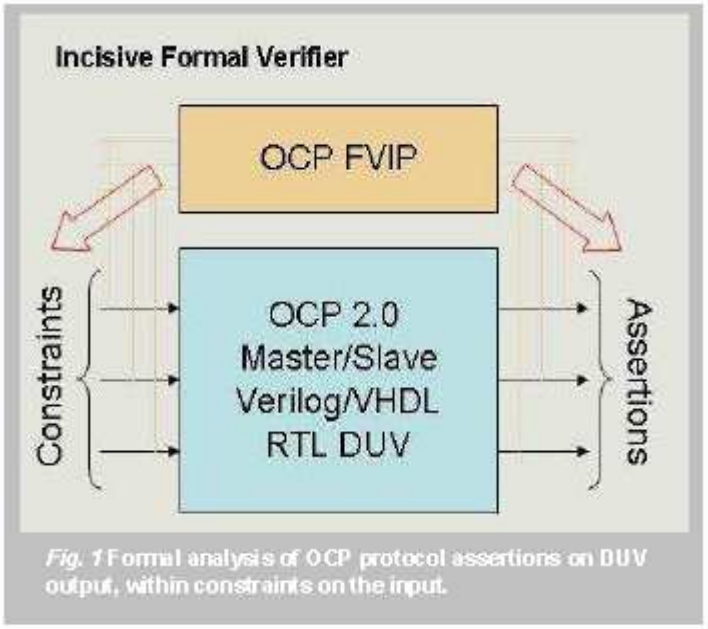
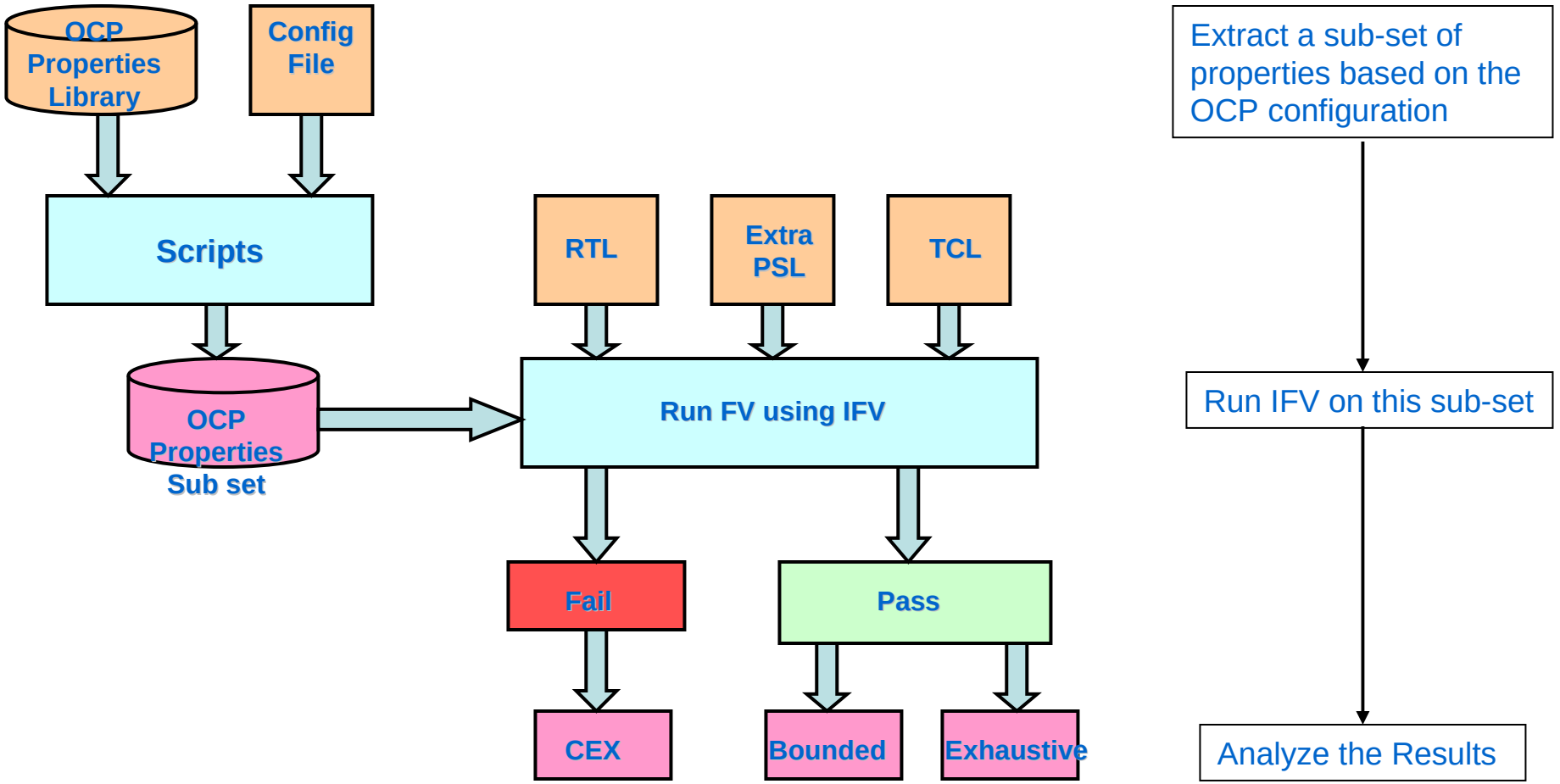



*Fig. 1* Formal analysis of OCP protocol assertions on DUV output, within constraints on the input.

- OCP ABVIP
  - Full OCP 2.0 FVWG support
  - Targeted for dynamic verification (simulation) and formal verification.
  - Fully configurable based on OCP RTL conf file.
  - ~90 protocol checks

# Cadence OCP ABVIP- Verification Flow



# OCP ABVIP Results and it's Meaning

- Fail => CEX (counter example)
  - IFV shows the shortest trace which violates the OCP property.
  - Debug & fix the RTL [use Go2Cause]
  - Add missing constraints, exclude false violations.

- Pass => bounded
  - Up to a certain depth of the state-space, IFV could not violate the property with in given time (effort).
  - Apply a higher effort, switch engine, reduce the circuit size (generics), add abstractions, …

- Pass => exhaustive
  - The property could never be violated.
  - We're done !!

# Results

| Design Name | Size (FF) | Time to complete | # Bugs found | Confidence (%) |
|---|---|---|---|---|
| RNG | ~500 | 1 week | 4 | 100 |
| DMA4 | ~100K | 3 week's | 0 | 80* |
| USB Host bridge (AHB to OCP & vice versa) | ~1500 | 1 week | 0 | 100 |

*** Some assertions are got explored at deeper depth [no Pass or Fail]**

# Case Study

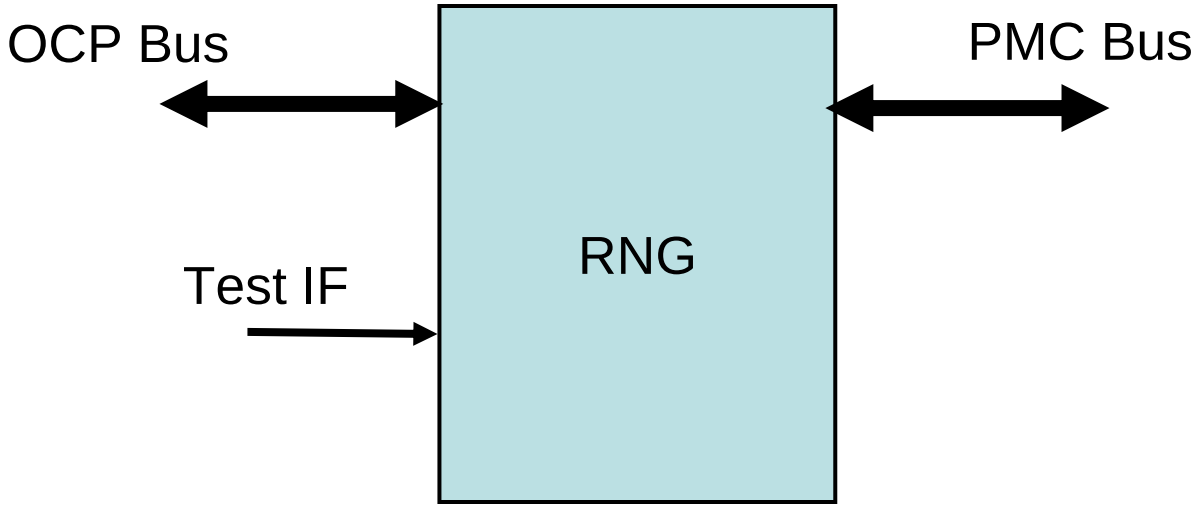
- Selected two designs:

- Random Number Generator (RNG)
- Direct Memory Access (DMA4)

# RNG block

## ➢ Description

- It is a random number generator (RNG) block.
- It has one OCP slave IF for register programming.
- OCP slave supports only IDLE/READ/WRITE commands.
- On other side it as simple power management control (PMC) interface.
- The design complexity is around 512 FF.

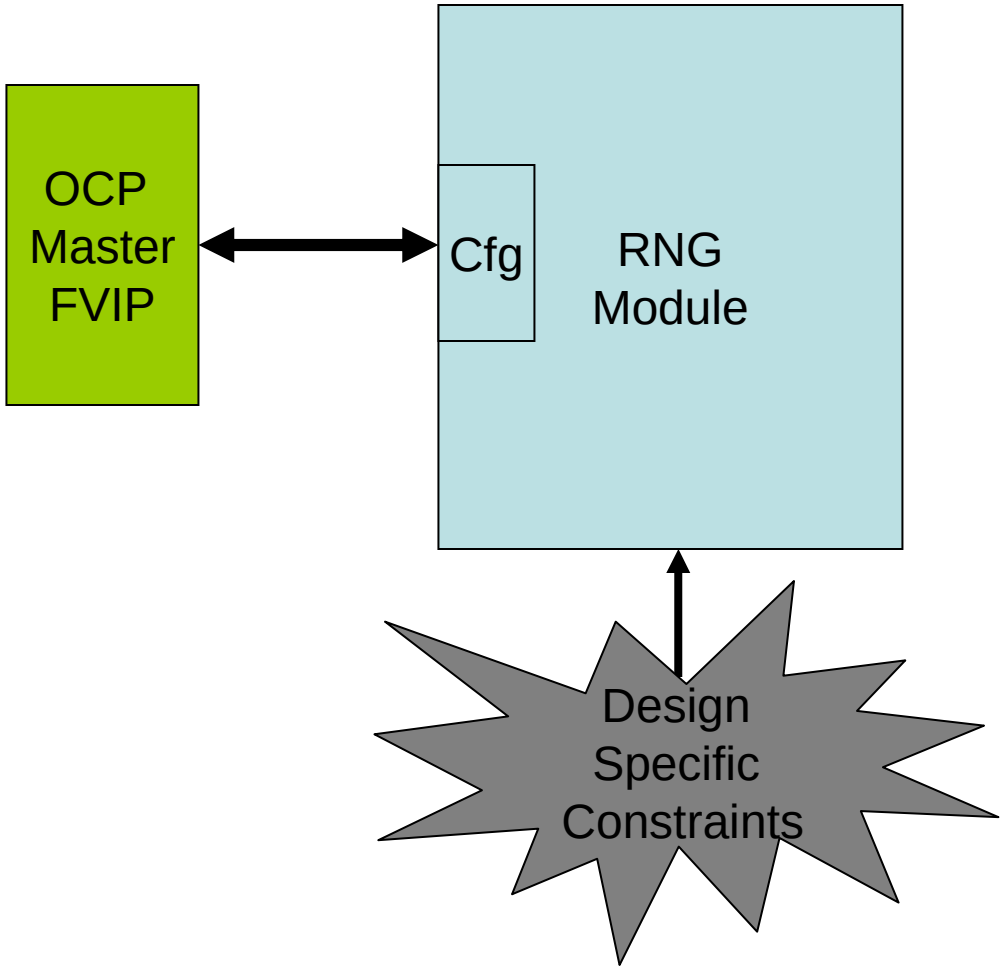


# RNG Block: Challenges & Goals

- Challenges
  - Pre-verified Block .

- Goals
  - Complete the OCP IF verification.
  - Write some design specific functional properties and prove them.
  - Write cover sequences.
  - Uncover hidden bugs.

# RNG FV Setup


OCP
Master
FVIP
Cfg
RNG
Module
Design
Specific
Constraints

# RNG Block: What we did

- Generated OCP checker files for OCP Slave DUV.
  - 8 Assertions and 6 Constraints. 23 cover properties.
- Ran Automatic checks (dead code and FSM ) and found no issues.
- Ran OCP checkers and **found two issues** in the design
- Design specific Properties (DSP's) are extraction from specification.
  - 7 Assertions.
- With DSP's we **found two issues** in the RTL.
- Written cover properties to check the valid sequences.
- Measured the design coverage using "report –cover" command.

# Issues Found : Protocol Violations

- Two cycle response observed for single MCmd request
  - **Corner case.**
  - **No body thought of this scenario until IFV shown a failure trace.**
  - **Very difficult to catch using the Simulation.**

- Null Response observed for Invalid MCMD instead of Error type response.
  - **Can be catch with Simulation.**
  - **But missed out by simulation plan.**

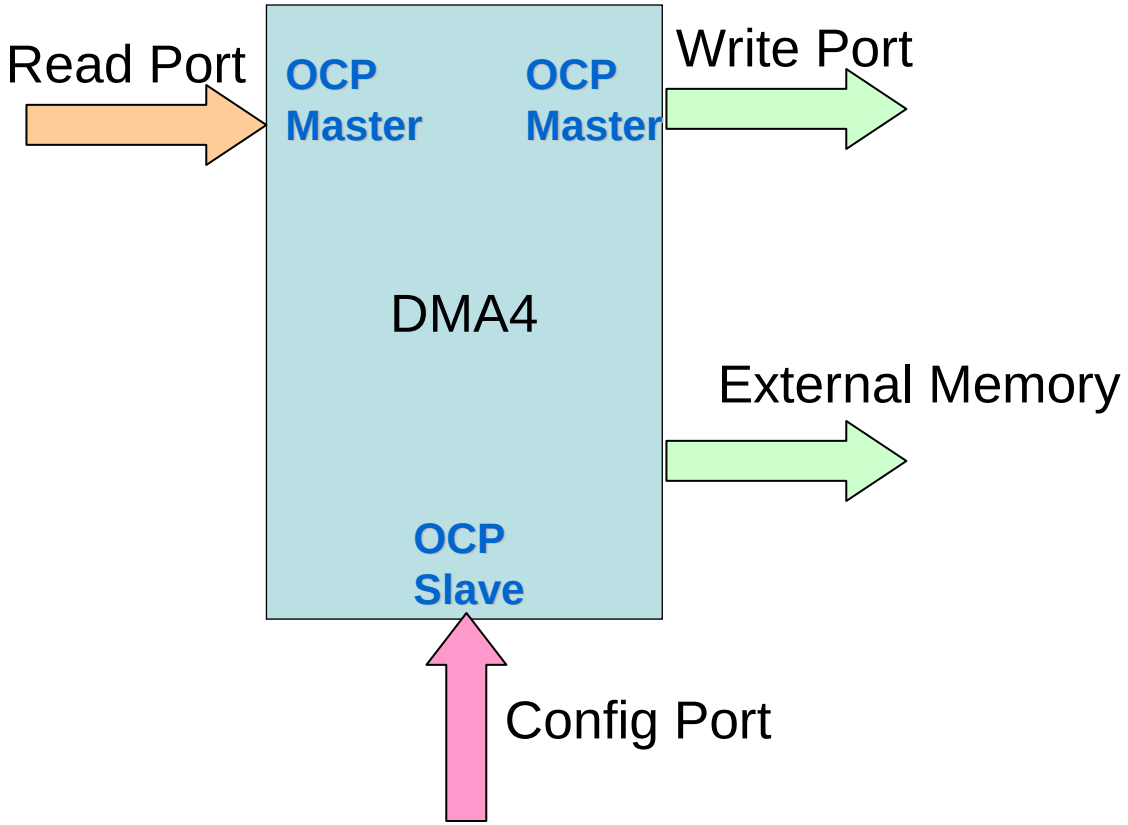
# Issues Found : Design Specific Violations

- Scmdaccept signal observed low for one more cycle after the reset.

- Error response was not coming in case of write operation to read only registers.

# DMA4 block

- Description
  - A OCP slave port used for configuration and access to status registers of DMA4
  - Two OCP master port's, one for read and another for write transactions.

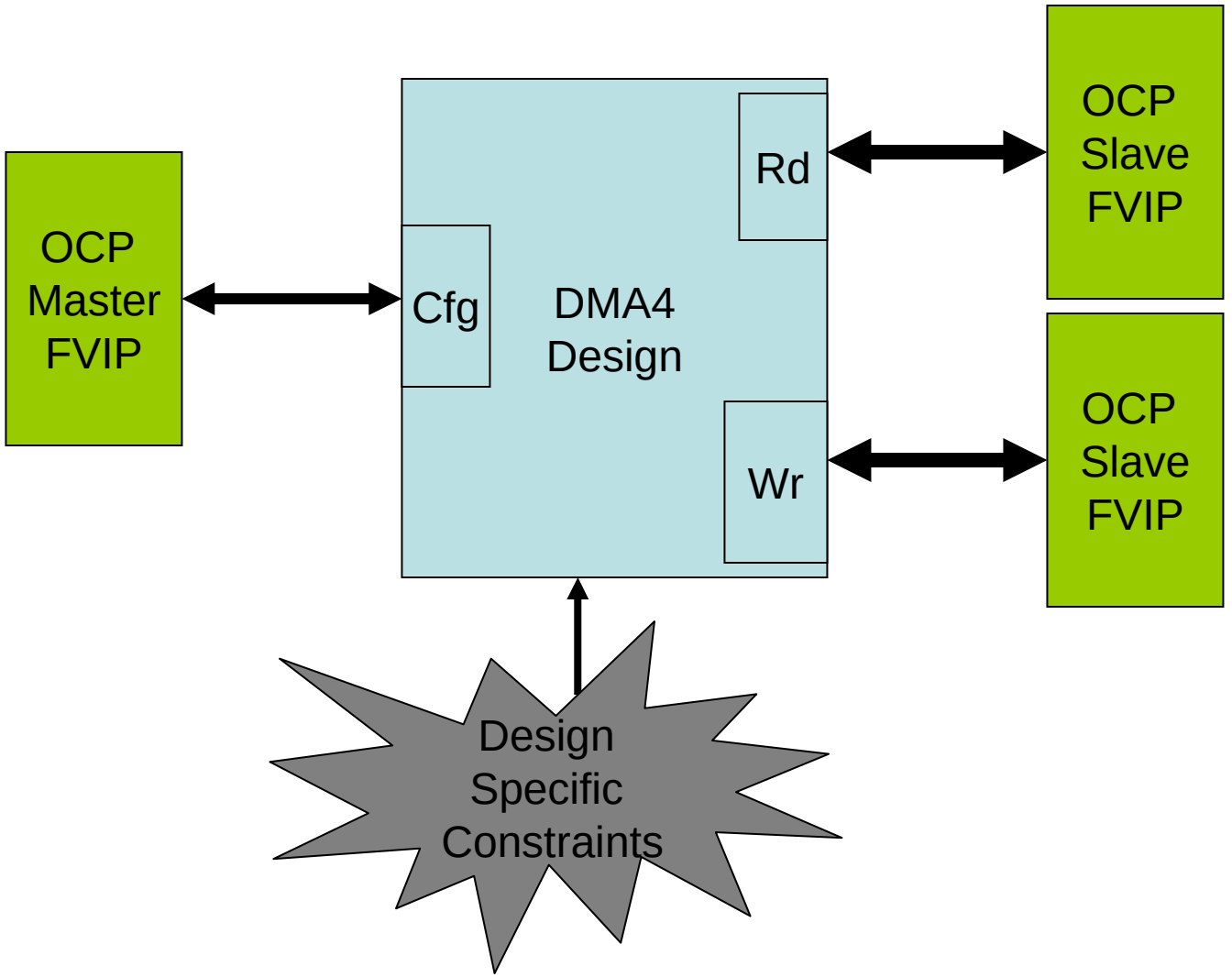


# Verification Goals & Challenges

- Goals
  - Verify the OCP interfaces for DMA4
  - Understand the Formal methodology to verify the complex modules.
  - Un-cover Hidden bugs.

- Challenges
  - Pre-verified Module.
  - On-the fly configurable External memories

# DMA4 FV Setup



# Configuration Port (OCP Slave)

- Pure formal Approach.
  - Used only OCP FVIP Assertions and Constraints.
- First ran all cover properties to check the setup correctness
  - Some cover failed (Indicates unsupported features in the design)
- Ran Assertions
  - Status: All assertions Pass (12)
- Achievement
  - 100% OCP slave protocol coverage

# Read/Write Port (OCP Master)

- Pure formal didn't work
  - Because of Complexity
  - Memories
  - On-the fly configuration
  - More no. of channels
- Divide and conquer approach couldn't applied
  - Because of no clear boundary between modules.
  - No clear understanding of internal design/signals.

# Read/write Port (OCP Master)

- So we followed semiformal approach
  - Because all configurations are not supported.
  - Take the Register configuration info from the VCD.
  - Constrained configuration port for no registers write.

- Applied black boxed approach
  - External Memory
  - Hardware Module

- Achievement
  - 80% OCP protocol coverage

# Results

- Configuration port (OCP slave)
  - 11 assertions : Pass
- Read port (OCP master)
  - 25 Assertions : Pass
  - 7 Assertions : Explored (depth :70)
- Write Port (OCP Master)
  - 24 Assertions : Pass
  - 7 Assertions : Explored (depth: 50)

# Conclusions

- Highlights
  - Formal thinks about scenarios that we may miss out.
  - Cadence ABVIP's enable a fast and exhaustive interface verification.
  - Learn Formal verification methodology and setup is easy and quick.
  - On control blocks Formal verification will give 100% results.
  - Debugging and finding cause of failure is easy with IFV.
  - Re-use of same ABVIP's and assertions in dynamic at chip level.
- Lowlights
  - Need better understanding of Formal verification to verify data path oriented designs.
  - Design or implementation details may be required to write correct assertions.

# THANK YOU !!!

# Q & A