DESIGN & VERIFICATION DVCON 2014 CONFERENCE & EXHIBITION

# Accelerated, High Quality SoC Memory Map Verification using Formal Techniques

**Cletan Sequeira** [1]
**Rajesh Kedia** [1]
**Lokesh Babu Pundreeka** [2]
**Bijitendra Mittra** [2]

[1] Texas Instruments India Private Limited
[2] Cadence Design Systems Incorporated

# Outline

- Introduction
- Prior Work
- Proposed solution
  - Use of Assertion Based VIPs
  - Memory Map verification
    - Features/Flavors
- Consolidation of features being verified
- Advantages
- Results
- Conclusion

# Introduction

- Our SoC contains a complex bus-system comprising:
  - 4 AHB masters.
  - Multiple AHB, APB and TI custom protocol based slaves.
  - Different stages of bus-fabric, bridges and decoders.
- Traditional verification flow suffer from below challenges:
  - Dependency on test case flow and chances of corner cases being missed.
  - Exhaustive verification leading to huge test development and runtime.
  - Conditions like “CPU getting stalled when sweeping through the whole memory map” could not be checked easily.
- SoC memory map verification using formal techniques was implemented:
  - To increase the exhaustiveness of verification.
  - To improve the development time.
  - To improve the runtime.

# Prior Work

- SoC level checks typically done using C or Assembly based directed test cases.
  - Exact use-case exercised.
  - Not exhaustive in nature.
  - Require significant time to develop, simulate and debug.
- Formal used for standalone systems like IPs and Bridges + directed tests at SoC.
  - Provides a high quality at unit level.
  - Does not ensure the whole system to work correctly.
- Use of emulation platforms as accelerator.
  - Happens late in the development cycle.
  - Still it may not fully guarantee all scenarios being covered.

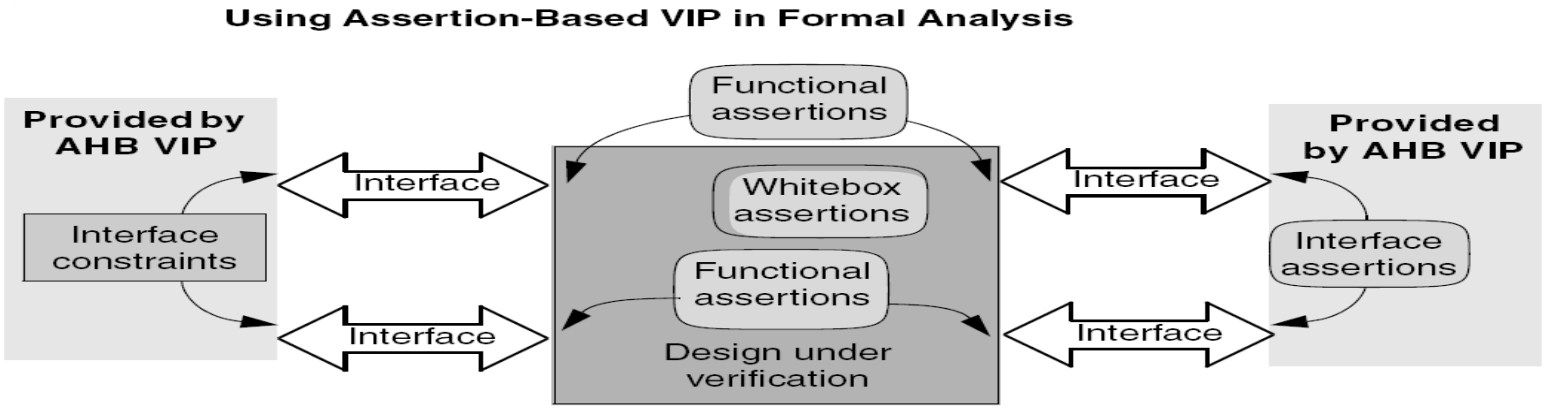
# Proposed Solution – Overview

- Hook up Assertion based Verification IPs (VIPs) to various masters and slaves:
  - AHB assertion VIPs[1] to each AHB master and slave.
  - APB assertion VIPs[1] to each APB slave.
  - TI custom protocol based VIP[2] to each custom slave.
- Properties in these VIPs are controlled precisely:
  - For masters, the master properties are made constraints and slave properties as assertions.
  - For slaves, the slave properties are made constraints and master properties as assertions.
- Develop memory map related functional properties and bind them to above VIPs
  - These will verify SoC memory map related aspects.

[1]Cadence VIPs used

[2]Coded in-house

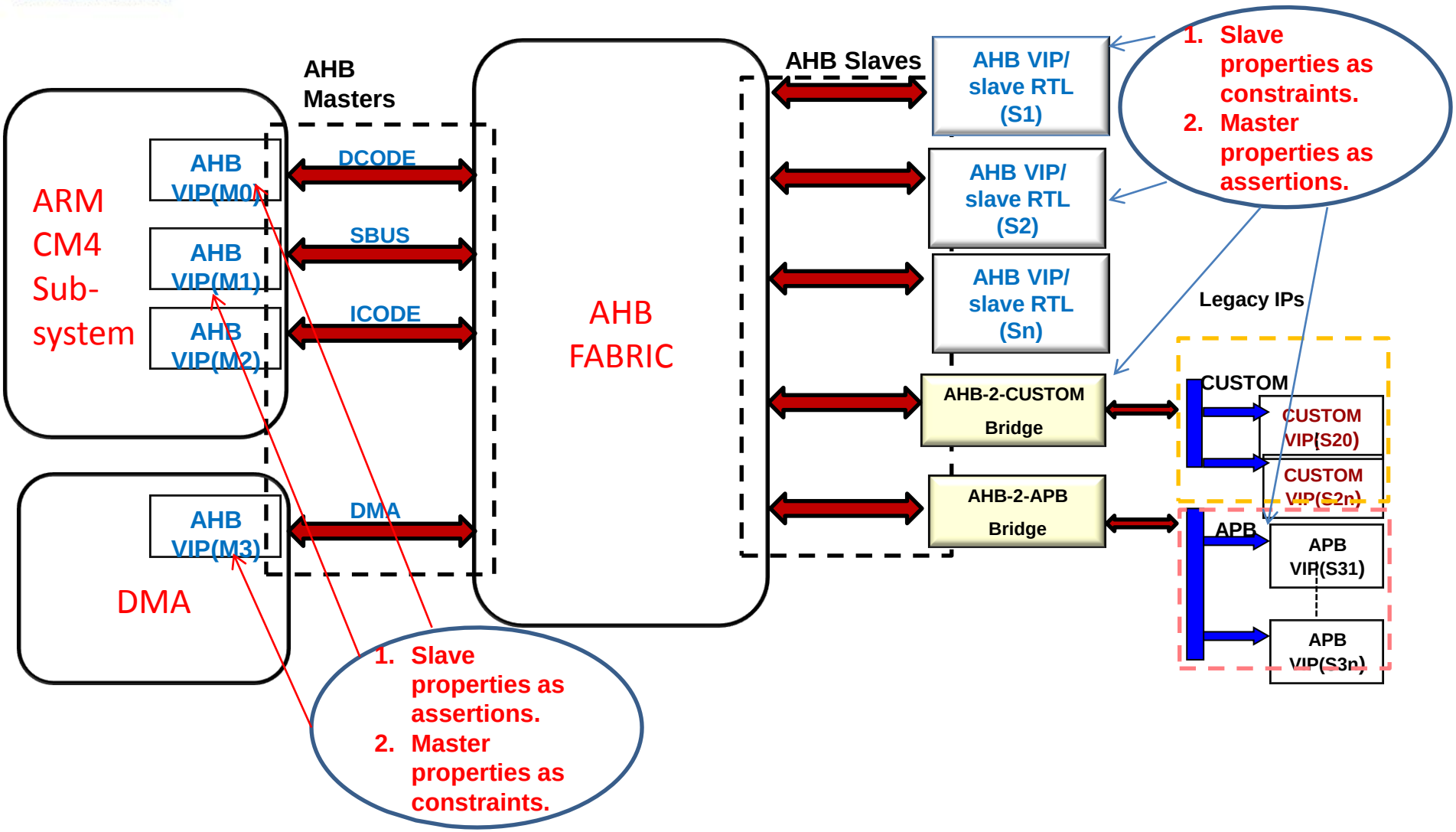
# Assertion Based VIPs




The same property file can act as checker or BFM via TCL control as shown in the example below:

- Assume property `prop1: ((!hresetn_i) -> (hready));` an AHB protocol check
- **`assert prop1`** in TCL → Property becomes an assertion and behaves as AHB slave checker. Whenever hresetn_i is '0', it will check that hready is '1' else flag an error.
- **`assume prop1`** in TCL → Property becomes a constraint and behaves as AHB slave. Whenever hresetn_i goes '0', it will force hready to '1'.

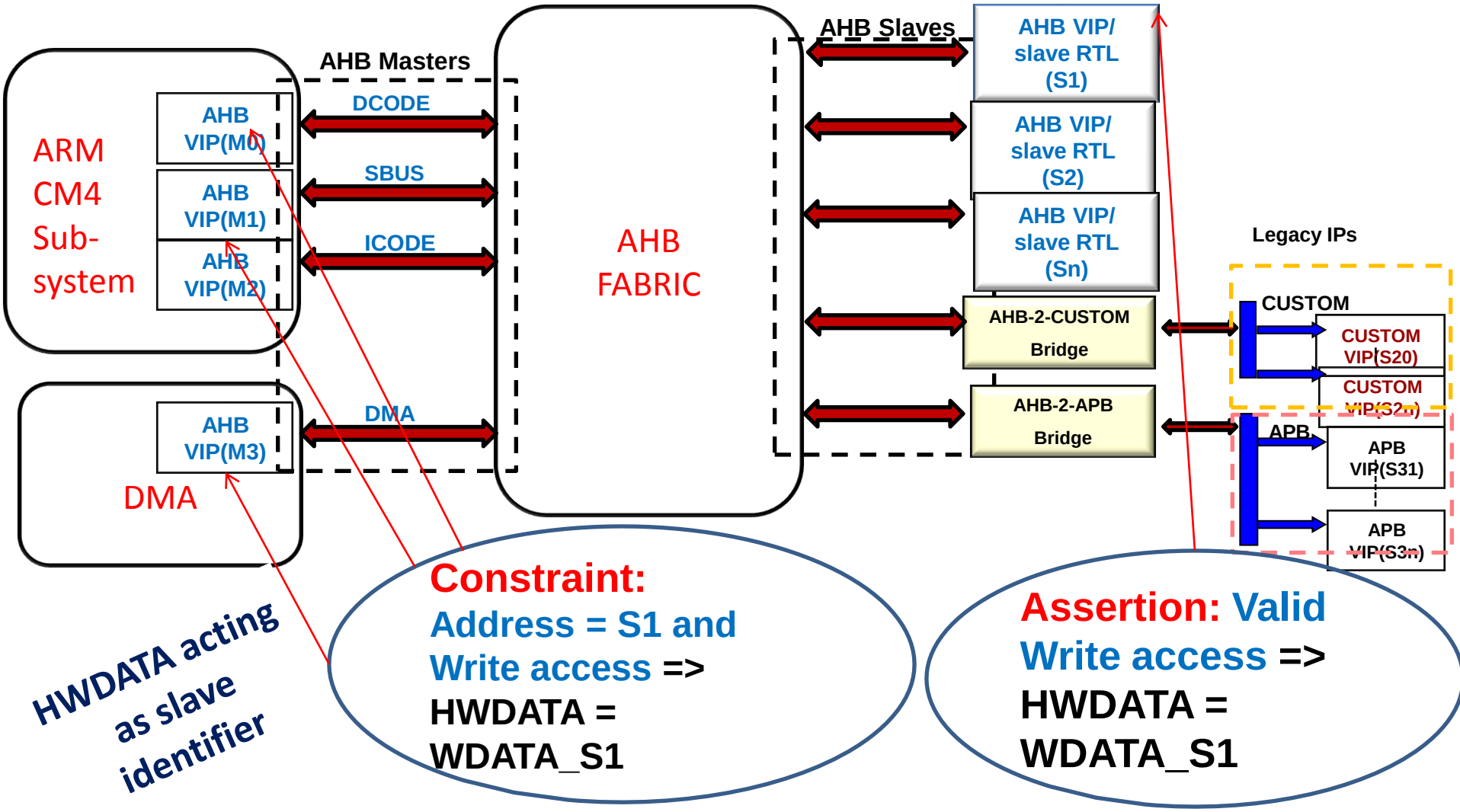
# ABVIP Hookup in our System



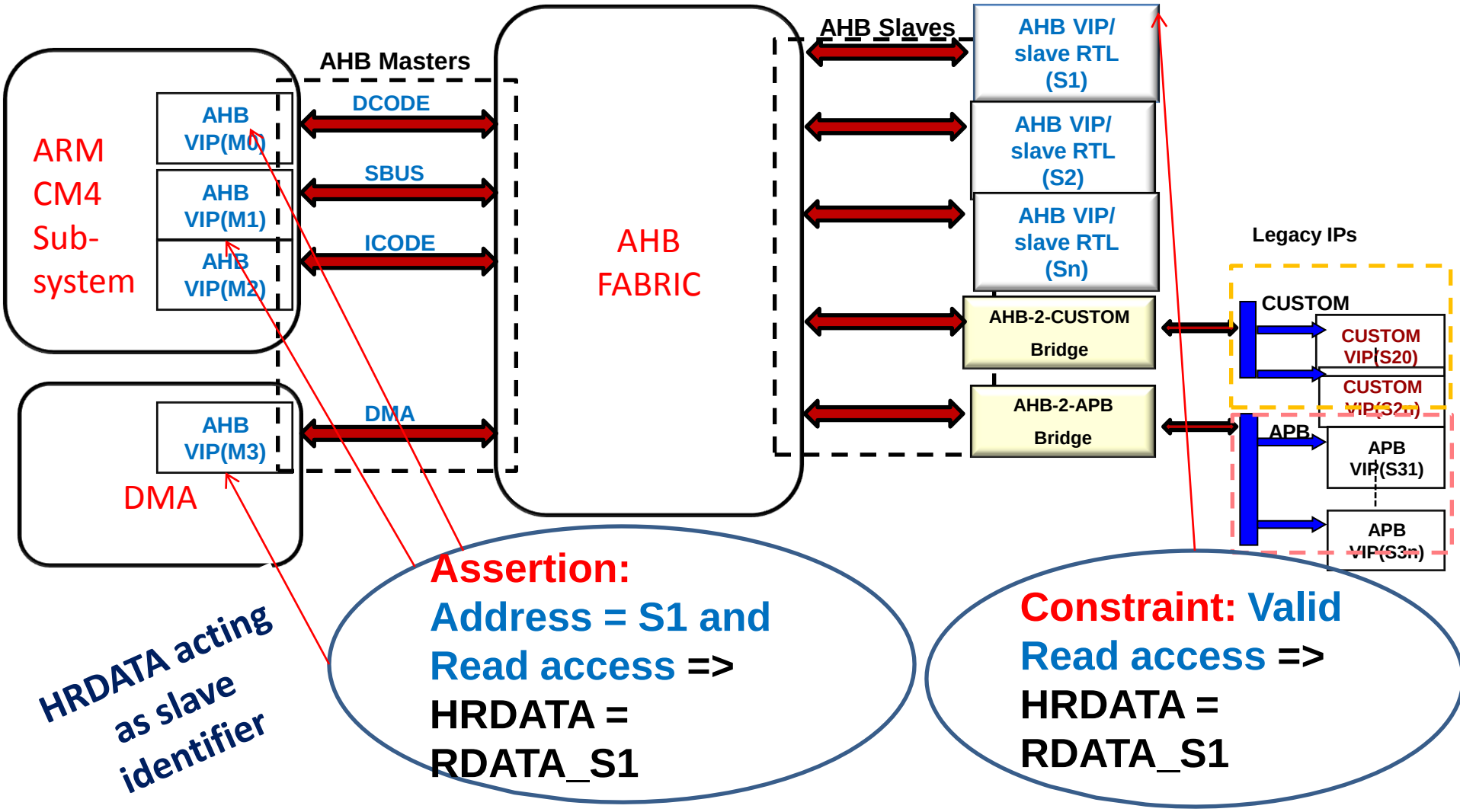
# Automation for ABVIP Hookup

| MASTER | SLAVE | START ADDRESS | END ADDRESS | SIZE | Sel_Path |
|---|---|---|---|---|---|
| M2#M0 | S31 | SA31 | EA31 | SZ31 | H1.pin1 |
| M2#M0 | S41 | SA41 | EA41 | SZ41 | H2.pin2 |
| M0#M1#M3 | S0 | SA0 | EA0 | SZ0 | H3.pin3 |
| M3#M1 | S1 | SA1 | EA1 | SZ1 | H4.pin4 |
| M2 | S1 | SA1m | EA1m | SZ1m | H4.pin4 |

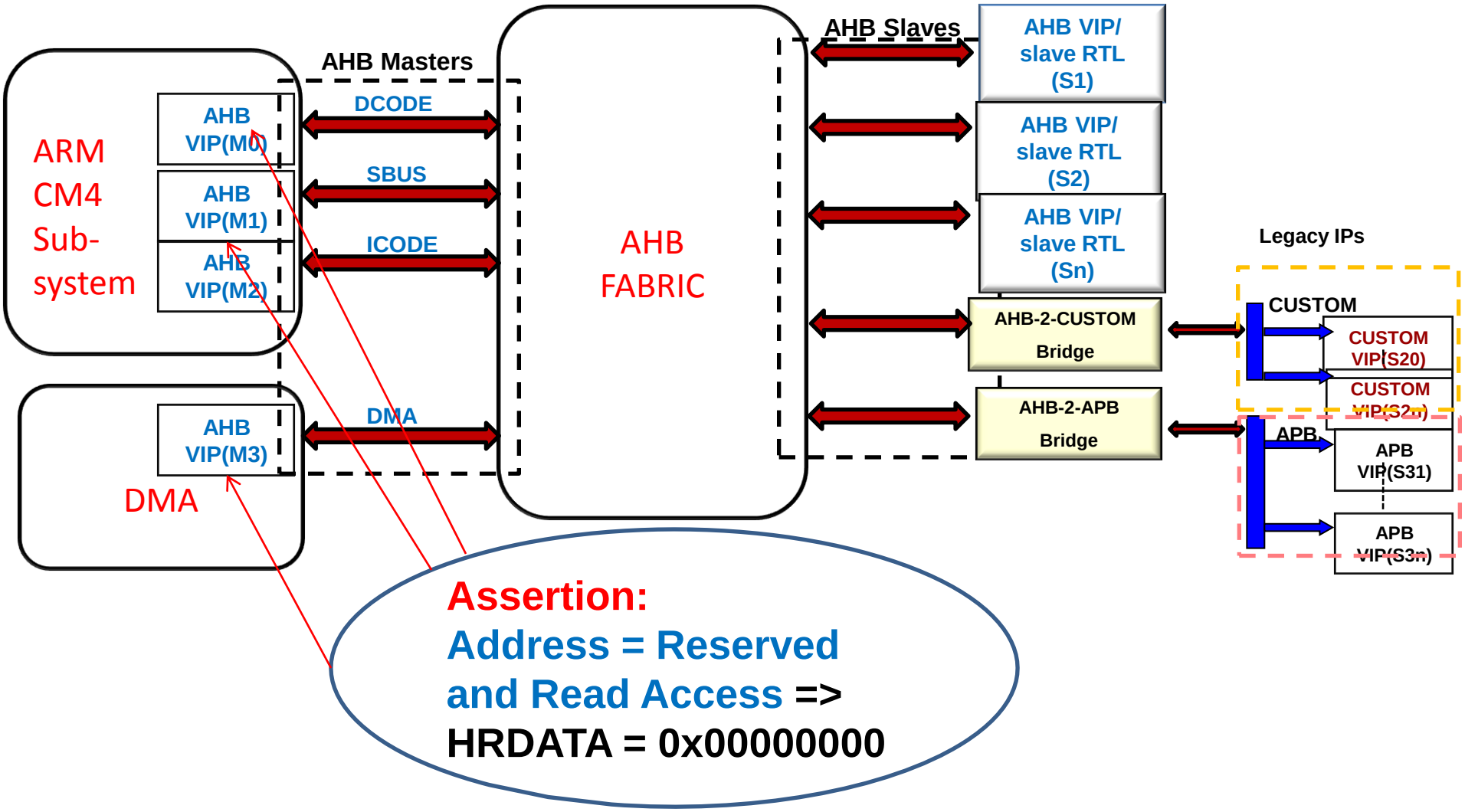
# Write Path Checks



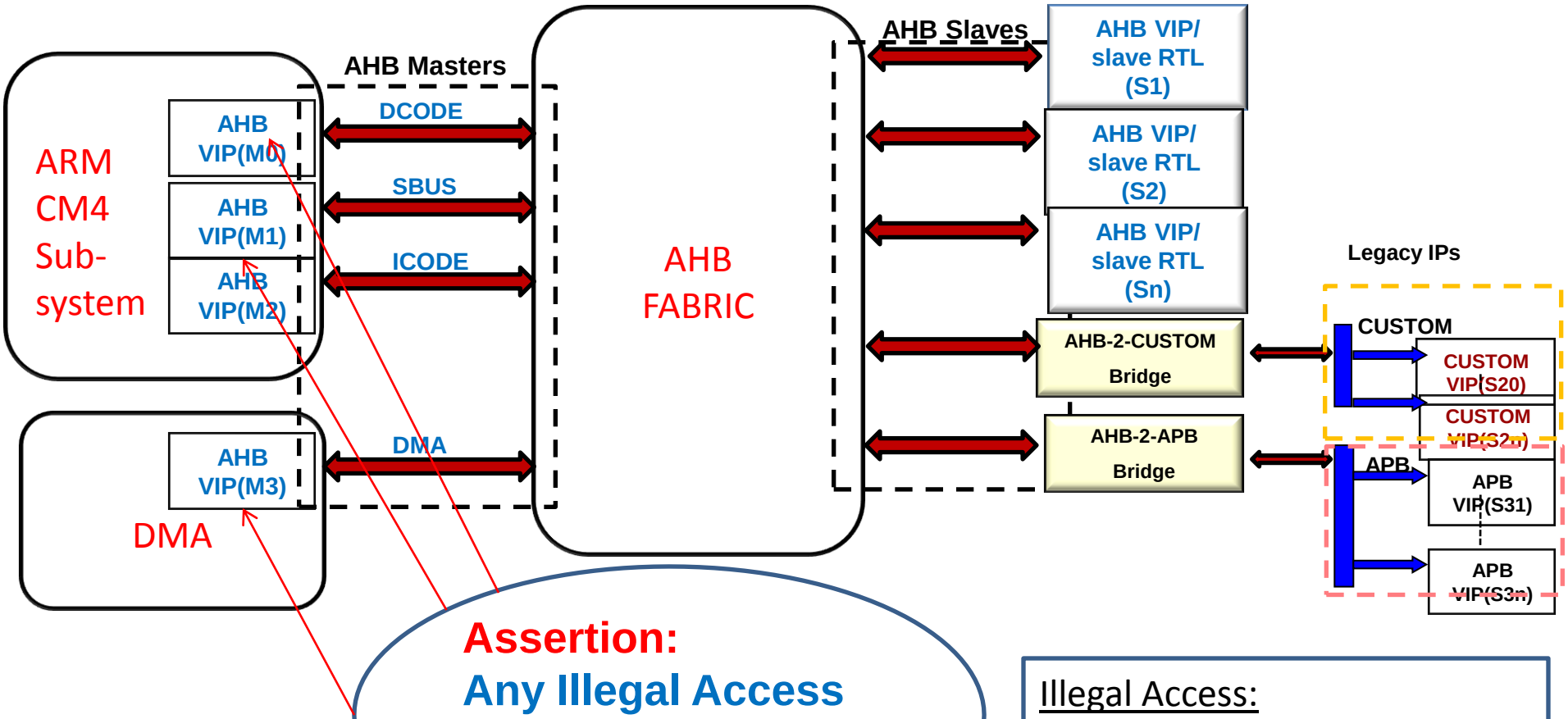
# Read Path Checks



# Reserved Space Checks



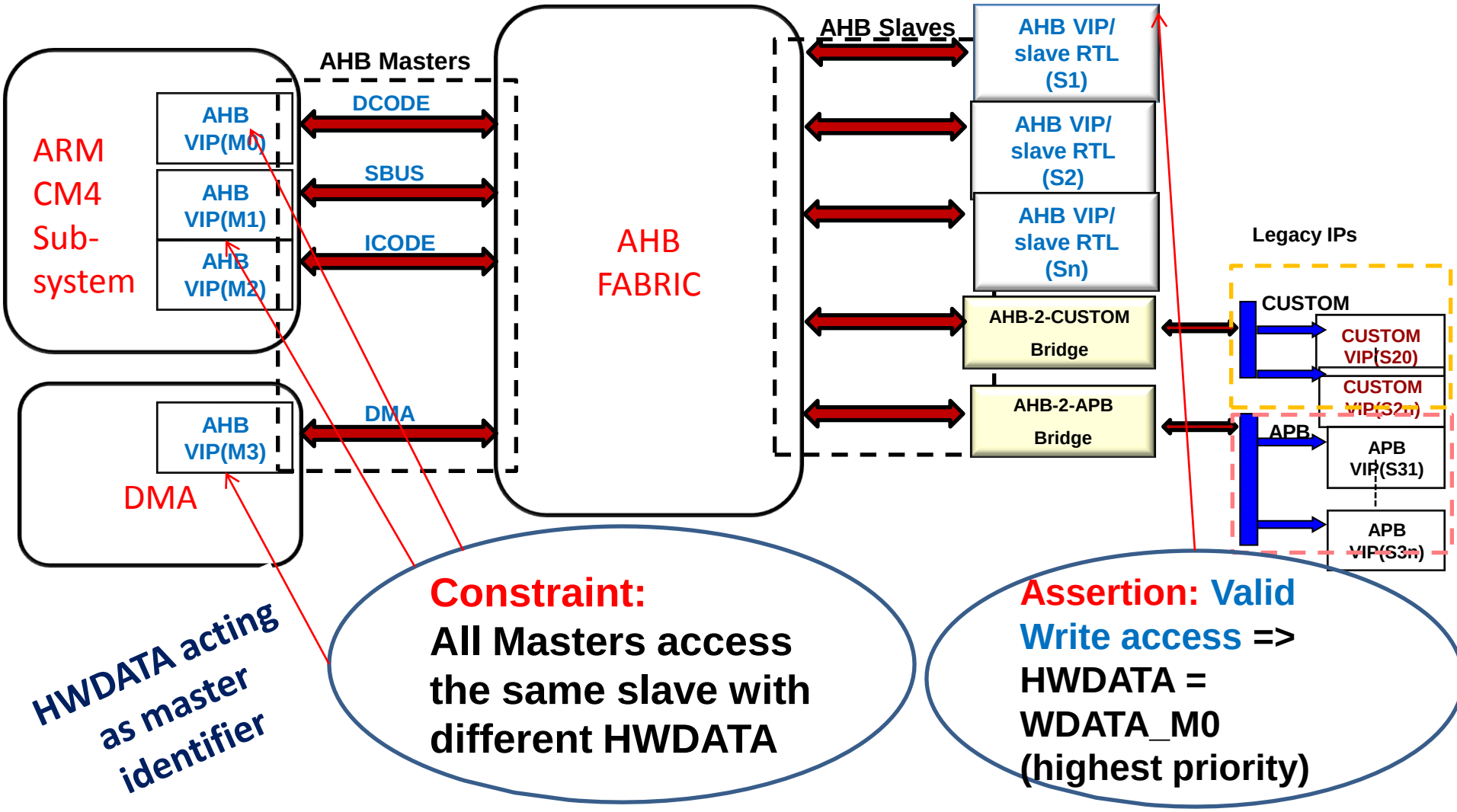
# Error Response Checks




**Assertion:**
**Any Illegal Access**
**=> HRESP = 1 in next cycle**

Illegal Access:
1. Access to invalid addresses
2. Access for higher width data for smaller width slaves

# Master Priority Checks



# Finally what gets verified

- Adherence of interconnect logic (bus fabric, decoders, bridges) to various protocols (AHB, APB, TI custom).
- Integration (Connectivity) of bus fabric with masters and slaves.
- Functionality of bus fabric, bridges and decoders.
- Accessibility and address mapping for each slave from various masters.
- Priority checks for each of the masters.
- Reserved space access behavior.
- Mirroring of Slave at different addresses for 2 different masters.
- Error response checks for all the invalid addresses.
- Error response checks for slaves not supporting 32-bit accesses.

# Advantages

- Only Assertions used for SoC memory map verification.
- Flow can work directly on the integrated RTL without changes.
- Can catch bugs early in development phase.
- Easily portable and customizable for various devices.
- Advantages inherent with FV that comes along:
  - Higher confidence on verification quality.
  - No disk-space issues because debug in IFV doesn't require waveforms to be dumped unlike in simulations.
  - Debug is quite simple – the exact (or sometimes close) cone of signals added to the waveform automatically.

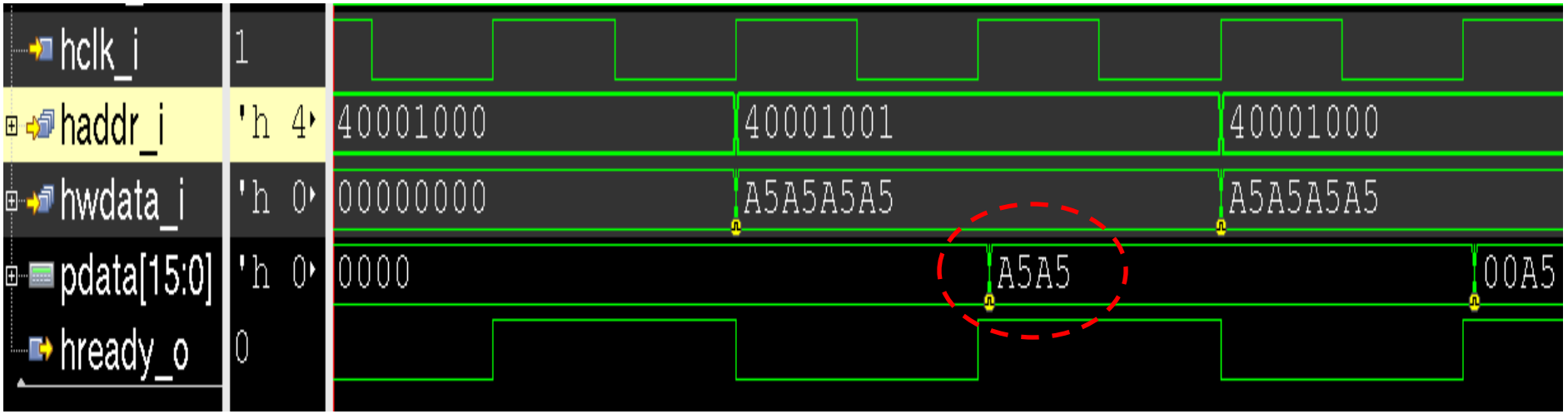
# Results

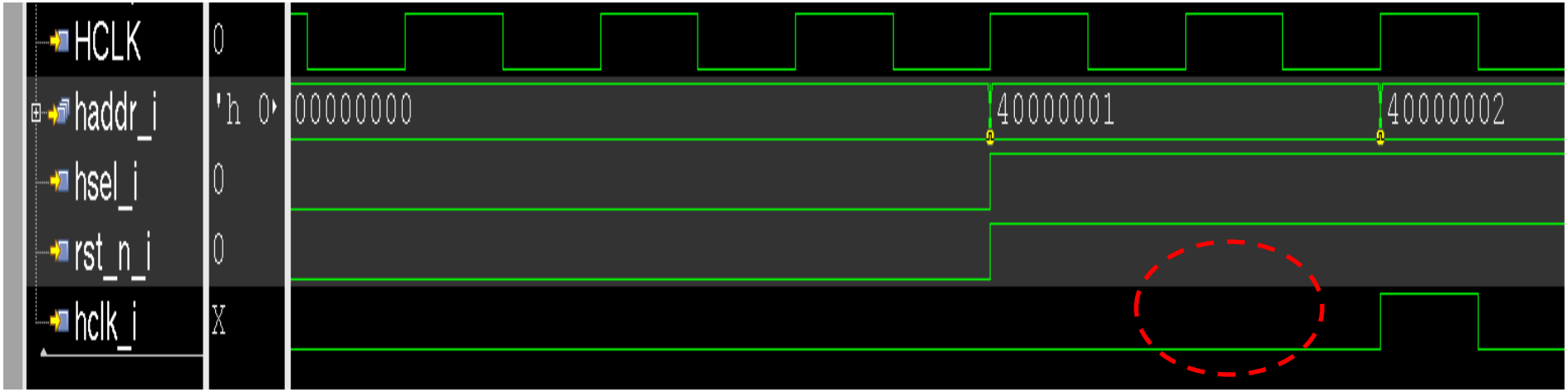
- Few corner case bugs were caught for the bridges.
- Verified access for all slaves with all possible masters very easily unlike in simulation.

| | Total Assertions | Assertions Passed | Assertions Failed | FV Run-time (hrs) |
|---|---|---|---|---|
| Protocol checks + memory map checks | 1300 | 1268 | 32 | 16 |
| Only memory map checks | 500 | 500 | 0 | 5 |

| Flow Deployment | Approximate bring-up effort |
|---|---|
| For the first device | ~ 6 weeks |
| For subsequent devices in family | ~1 week |

# Example of bugs caught



# Looking Back !

- What helped:
  - Following naming conventions for the design signals helped in automating the flow.
  - Availability of ABVIPs for AMBA™ protocols.
  - Debug features in the tool.

- Challenges faced:
  - Black Boxing unwanted modules.
  - Finding right design constraints.
  - Choosing the right solver.

# Conclusion

- FV based Memory Map verification flow has been described, which:
  - Reduces run time by a huge margin compared to simulations.
  - Provides more exhaustiveness and hence higher confidence on the verification quality.
  - Allows easy deployment in subsequent devices due to automated & configurable options in the flow.
  - Being generic, could be used with any protocol.

- Future Scope
  - Enhance the flow by using formal scoreboard for data integrity checks.
  - Replace each slave ABVIP with the corresponding RTL to gain further confidence on the whole system.

# References


- Property Specification Language Reference Manual (http://www.eda.org/vfv/docs/PSL-v1.1.pdf)
- Cadence Incisive Formal Verifier User Manual
- Cadence Assertion Based Verification IP User Guide
- AMBA™ 3 Specification Rev 1.0, http://www.arm.com

# Thanks