Automation Framework Based IP/Subsystem Integration Verification in SoC
-A Systematic Approach for Integration Quality Signoff

Aswin B, Yogeshwaran S, Karthik Rajakumar

60 DESIGN AUTOMATION CONFERENCE
FROM CHIPS TO SYSTEMS — LEARN TODAY, CREATE TOMORROW

1. Motivation & Problem Statement

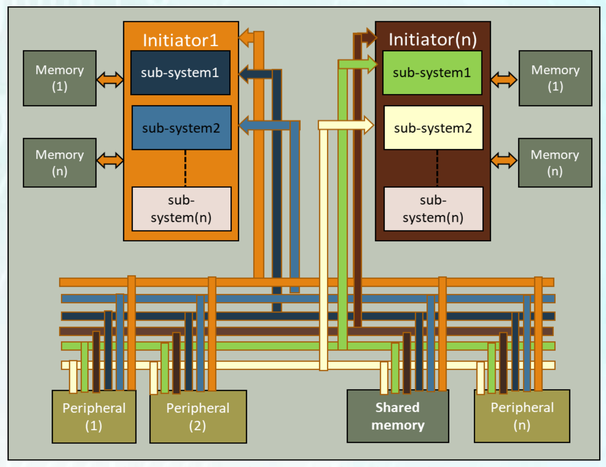
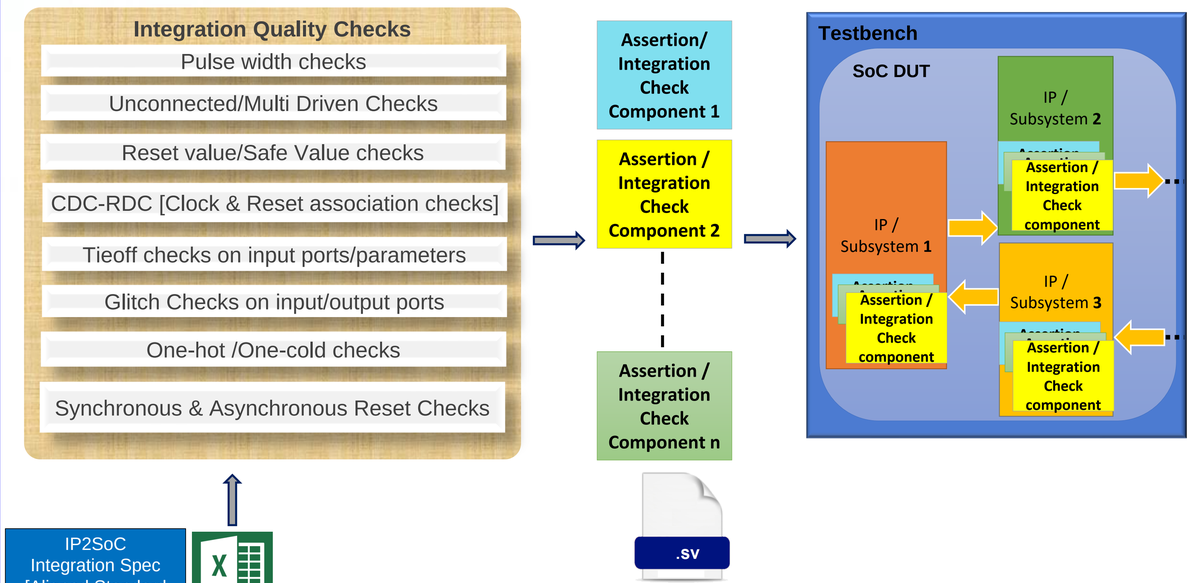


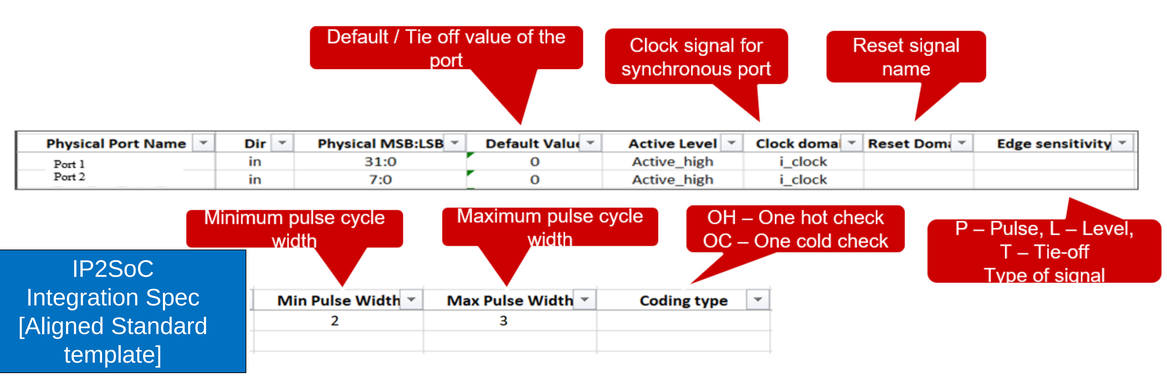
Fig.1 A typical SOC multi core design used for advanced user applications

- In this catalogue market era, SoCs are more generic to cover multiple end applications, so a typical SoC may contain 10s or 100s of sub-systems / IP instances integrated together
- Each IP / sub-system by itself can have 100s or 1000s of ports which gets connected across in the SoC
- These modules may be designed independently from various teams/companies and when integrated, there can be common issues/mismatches (IP / Integration BUGS) due to inter module assumptions/ ambiguous protocol handshakes / misses in integration to take care IP assumptions
- Ensuring integration correctness of all modules is critical to achieve early bug findings and on-time systematic verification closure
- Writing these integration checks manually for all the ports of all modules in an SOC consumes time and viable to manual error / miss / late findings in the design cycle
- Each IPDV assumptions on input ports and assertions on its output ports can be reused at SoC as RTL embedded assertions to identify these mismatches but completeness of them on all ports is not guaranteed E.g.: if assumptions/assertions are missed it can be a surprise in the SoC cycle
- We need a systematic approach to signoff "**All common integration checks x All Ports x All Modules"** with Metric Driven Verification (MDV) signoff i.e., coverage based for quality closure of integration
- In SoC DV, as multiple modules get integrated together we can use SoC level Formal Verification(FV) setup with all these assertions enabled to find assumption mismatches across modules very early and catch these bugs earlier in the design even without simulation based tests developed

2. Proposed Solution(1/3)

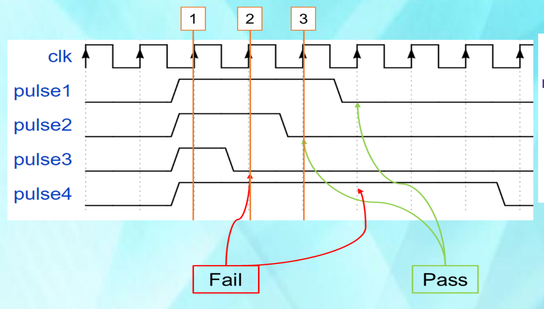
- An automation framework to generate and bind all integration checks on all ports of all IPs/ subsystems in an SOC using SystemVerilog Assertions auto-generated from IP2SoC integration handoff specification





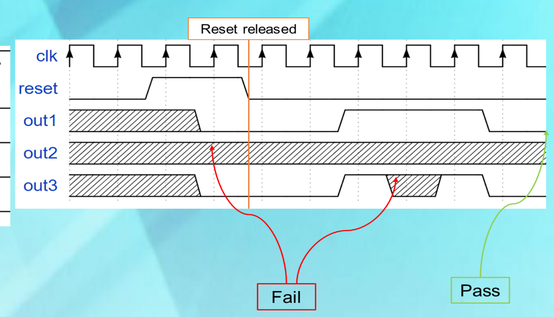
3. Proposed Solution(2/3)

Pulse width check



For Pulse expectation of 2 to 3 cycles

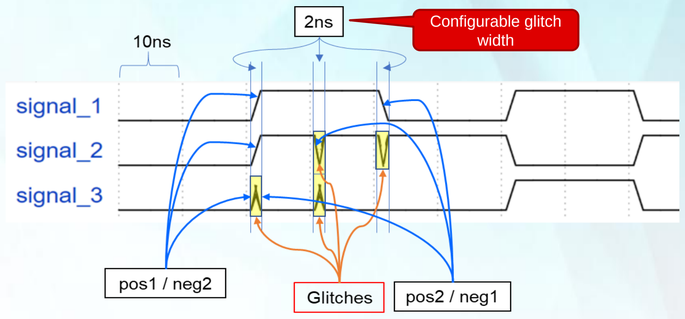
Multi-driven/Undriven Check



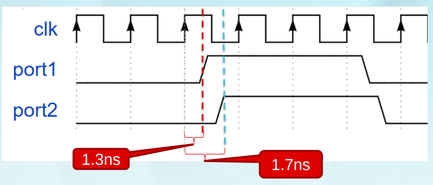
X – Multi-driven port
Z / U – Undriven port

4. Proposed Solution(3/3)

Glitch check

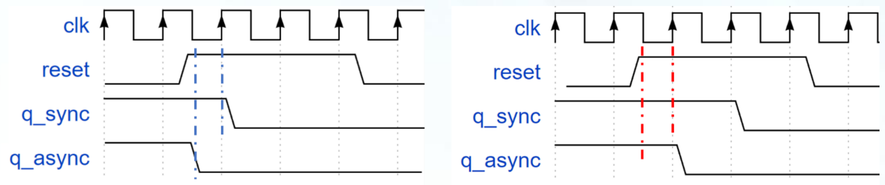


Clock association check



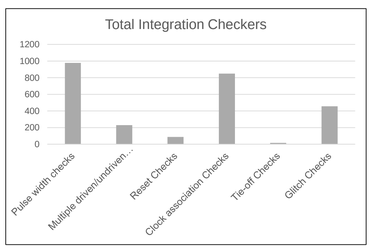
Sync or async outputs check with corresponding clock associated with configurable delta delay and setup/hold time

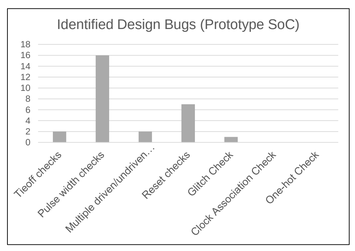
Synchronous and asynchronous reset assertion check

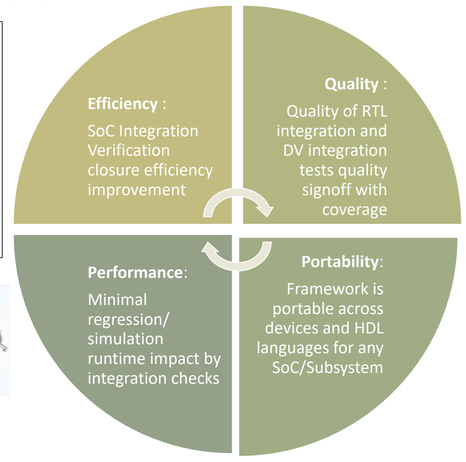


5. Evidence and Results

- Prototyped and developed assertion checks for multiple flavors of different levels of module complexities (Control peripheral, Communication peripheral, memory controller., etc)
- Modules in different HDL languages like VHDL and Verilog and SV were evaluated in the setup
- Identified 28 integration bugs in the SoC including 1 IP limitation/bug when functional verification was in ~50% completion at SoC context within short time of integration checker addition (~2 days)
- Currently in production use for new platform real time microcontroller device SoC DV in the business unit







Summary:

- Improves Integration quality and can be signed off with MDV
- Increases SoC verification efficiency
- Early bug findings and closure of all integration issues in DV initial phase

6. Conclusions

Generation Time Configurations of Integration Checks Components:
- IP2SoC Handoff specification has various options to configure pulse widths, polarity with check enable/disable logic as conditions
- If waivers needed, IP experts can leave blank if any of the checks have to be skipped for any ports with comment/reason

Compile Time Configurations of Integration Check Components:
- By default all components gets binded to all instances, but any specific check component be manually skipped for any instance if needed
- Compile time switches to disable all/ any specific type of checks across all components [E.g.: To disable glitch checker in RTL setup for all components]

Run Time Configurations of Integration Check Components:
- Any specific type of integration assertion/all integration checks can be disabled with runtime plusargs (E.g.: +disable_clock_association_chk)
- Any particular module assertion / specific assertions for a port can be turned off by plusargs

Simulation / Regression Runtime Performance Analysis:
- Since multiple assertions are getting added across all ports of all modules, we reviewed simulation profile analysis(XPROF) in an SoC prototype
- Enhanced all integration checks/monitors/assertions to be smartly turned ON by itself as required during runtime
- Minimal regression/simulation runtime impact (Average : 1.6% slower compared to no integration check regression)

Portability & Quality:
- Assertion generated can be used in both SoC formal environment and simulation environment
- Integration assertions coverage metrics can be analyzed in formal and simulation environment can be used to review integration tests/environment quality

Acknowledgements

Arif Mohammed, Saravanan Gajendran

TEXAS INSTRUMENTS