SoC Verification Plan

Gedeon Nyengele, Kalhan Koul & Rick Bahr

Sept 24, 2020

SoC Development AND Test Directions

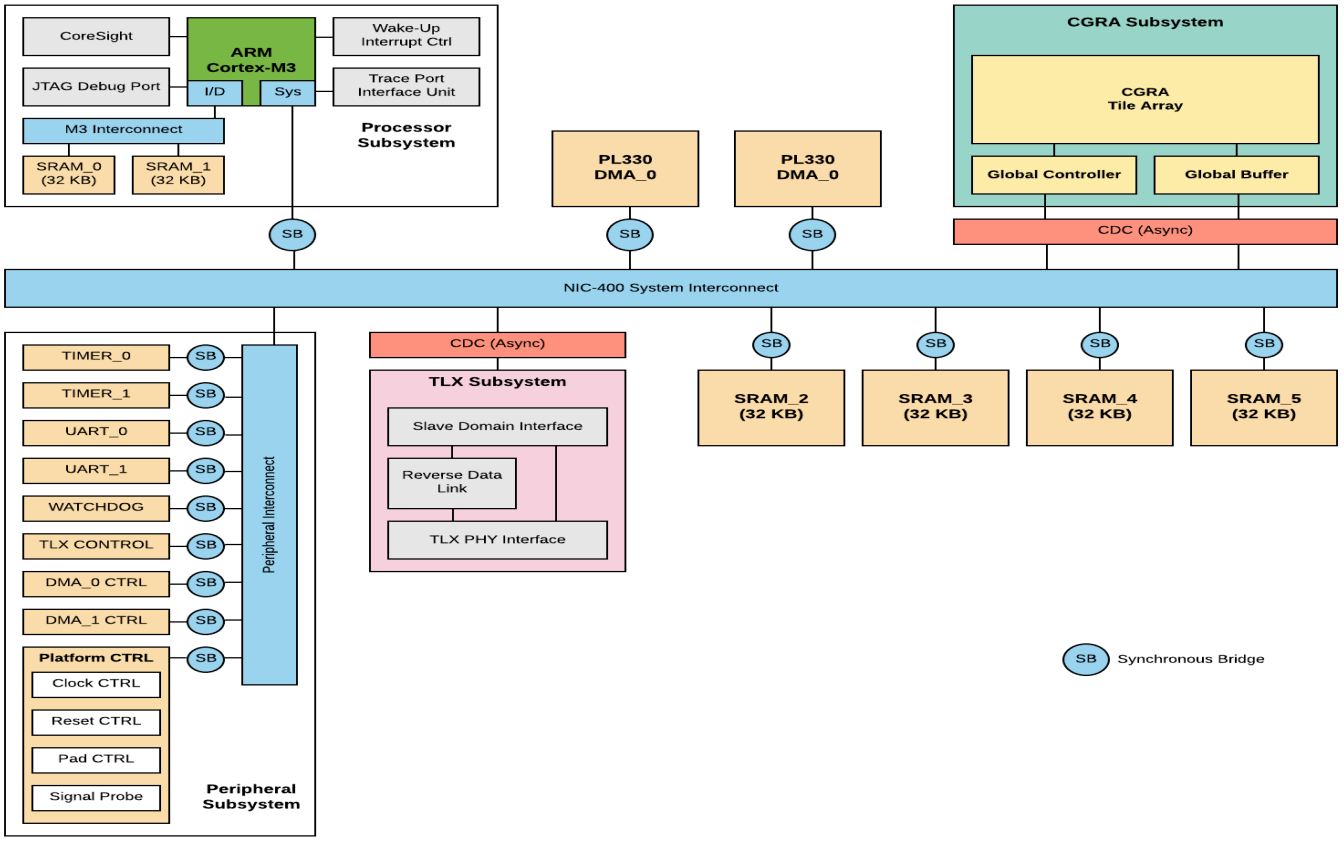
- Tops Down
  - Best way to map application(s) to a given SoC
  - DV strategy: Running applications

- Bottoms Up
  - Best way to develop an SoC for given application(s)
  - DV strategy: IP/Building Block integration testing

Generic SoC DV Principles

- Follow “correct by construction” as much as possible
  - Use known or proven IP elements/building blocks
  - Use standardized interconnect strategies for both function (application) and management (init, debug, etc)
  - Setup global agreements for “chip resources”: clocking, power, testability, debug, area
- SoC integration testing (bottoms up)
  - Confirm compliance to chip level agreements (directed, random testing and formal checks)
  - Have as many integration points as practical in the timeline (hw/hw and hw/sw)
  - Substitute BFM’s (bus functional models) when design is absent or controllability/observability needed
  - Note: Comprehensive block level coverage only for designed elements
- SoC application testing (tops down)
  - Looking for unit interactions, sw/hw compatibility & system level goal compliance
  - Interface compliance (e.g., PCIe, TLX, etc)
  - Modal operation (initialization, sleep, cal, etc)
  - Critical recipe’s (synthetic tests)
  - End to end application correctness
  - End to end application performance
  - End to end application power consumption

Amber SoC Block Diagram



Amber Verification Strategty

ARM provided RTL is assumed bugfree

- No DV coverage statistics will be needed for SoC core RTL

Bottoms up verification will focus on integration and thru simulation

- E.g., SoC core, DMA to/from CGRA, TLX PHY, Memory, etc
- Functional Models will be substituted for RTL when controllability/visibility is needed
- Comprehensive verification will be in unit testing, and on designed blocks

Top down verification will be primarily thru emulation

- Selected end to end applications will be run
- Selected recipe's to be proven
    - E.g., TLX calibration, Reset(s), Boot load, Data Transfers, CGRA initiation/completion

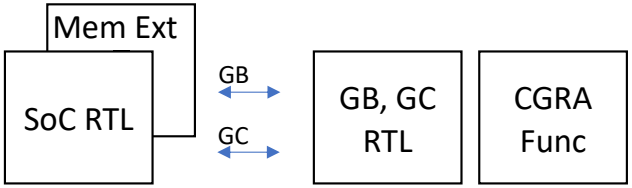
Continuous regression testing scoreboarding

- Integrated with MFLOWGEN

Amber Verification Frameworks (Simulation)

A.

B.



Amber Framework Strategy (Simulation)

Framework A (sim) is to drive coverage of GB/GC interface protocols

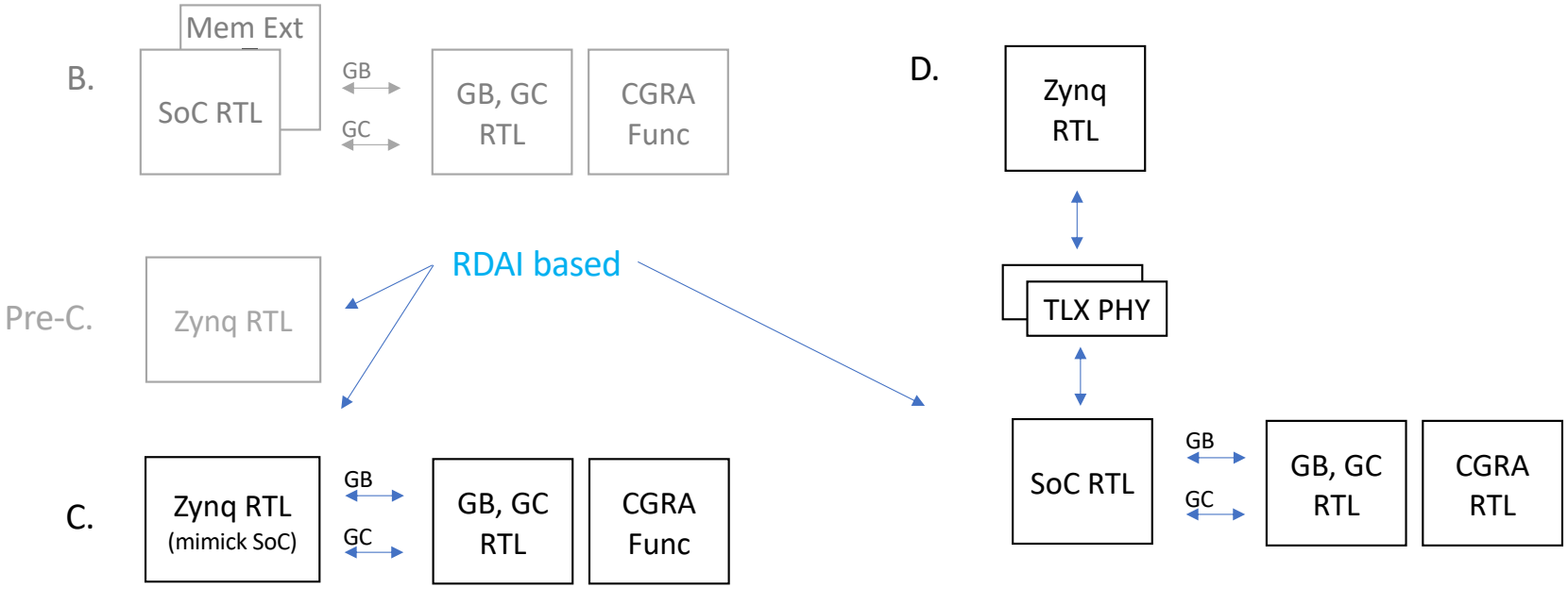
- Replay of unit testing at AXI bus level for GB/GC data transfer/control testing
- Ideally coverage based, w constrained random stimulus test if needed

Framework B (sim) is for SoC level integration testing

- Replay of ARM provided tests
- Incremental unit tests for AHA designed blocks (clocking, TLX, etc)
- Synthetic tests for application management
- M3 driven kernel-application tests (subset kernel, subset data)

Note: Exploring constrained randoms & formal verification extensions (discussion w Clark Barrett)

Amber Verification Frameworks (Emulation)



Amber Framework Strategy (Emulation)

Framework pre-C (emulation) is for RDAI development

- Demo soon w HLS

Framework C (emulation) is for application running

- Functional operation
- Selected end to end applications

Framework D (emulation) will primarily be post tapeout

- Initialization & debug recipe's
- Establishing host link recipe's
- Running selected end to end application to completion

Note: Exploring FPGA emulation also for circuit level modeling of TLX (discussion w Steve Herbst)

DV Test Plan - Two Prongs: Building Blocks and Applications

Goal (functional coverage given around 8 weeks until tapeout)

- Framework A: BFM for testing GB and GC – Owner: Taeyoung
- Framework B: Integration Testing, and Representative workloads – Owner: Kalhan
- Framework C: Emulation of SoC on Zynq – Owner: Gedeon/Charles
- Forward looking dv infrastructure
- Detailed Testing Plan

Framework A

- Starting point will be the replay of the GB/GC unit tests at the SoC, AXI bus level
- Introduction of coverage tools
- Introduction of basic constrained randoms platform

Framework B: Bottom up and Top Down Strategy

Bottoms up testing: Building block integration testing (Framework B, primarily simulation)

- Includes DMA, interrupts, boot, GB<-> CGRA, etc.
- Actual vs synthetic tests ex. DMA input image vs DMA min & max length

Tops down testing: Application testing (Frameworks B & C, primarily emulation)

- Conv3x3, cascade -> resnet layer -> resnet-50, virtualization, etc.
- Start with M3 Level (Framework B, simulation then emulation in Framework C)

- Automated regression environment setup
- Review IPO/Building Blocks for integration coverage (to develop SoC related synthetic tests)
- Review Global Buffer/Global Controller to develop synthetic tests (integration and coverage)
- Review new Garnet to Amber features (for new feature testing, e.g. virtualization support)
- APIs for M3 Code (Framework B)
  - Current state – series of register writes on GC/GB
  - Abstracting this allows us to write applications faster
- Speedup CPU image load (Infrastructure)
- Resolve bitstream versioning/acquiring
- Review clock generation/testing

Framework C, D End to End Testing

- Framework C replaces SoC with Zynq A53 + Zynq R cores + (GM/GC) + CGRA Func
  - We're just emulating the CGRA and using the Zynq FPGA as SoC RTL. The goal here is rapid host + GarnetSoC app flow development
- Framework D is where we emulate the entire Garnet SoC on the Zynq FPGA

Amber DV Plan Timeline

- Now — Working, synthesizable and placeable Garnet SoC RTL for Amber
- 9/29 — Emulation demo for RDAI on FPGA (Framework pre-C)
- 10/4 — Any planned SoC RTL changes (new features for virtualization or debug?) defined, test plan defined
- 10/18 — Any new SoC RTL integrated into Amber flows
- 11/2 — Verification Plan completed (bugs closed)
- 11/2? — Emulation demo for RDAI on subset CGRA w SoC (Framework C)
- 11/16 — TSMC Amber submission
- 12/1 — TSMC Amber shuttle

- To be Investigated
  - Inclusion of coverage tools
  - Introduction of Formal Verification
  - FPGA emulation of TLX link

- Post Tapeout
  - D emulation framework for applications
  - Constrained randoms bus transactors
  - Direct mapping flow from Halide to Framework D