Verification Reuse

Ilia Gendelev – Cisco

Ory Tal - Cadence

Verification Reuse

- Verification IPs (e.g. eVCs)
- VIP Implementation
- Block to Full Chip Reuse
- Project to Project Reuse
- Legacy Code Reuse

Verification IPs (e.g. eVCs)

- Usage of standards (e.g. eRM)
- Common protocols – acquire from external vendors
- Proprietary protocols - Internal implementation
  - Development by dedicated verification group (No project will invest in VIP implementation)
  - Same look and feel (and coding style) between VIPs
- Non protocol VIP (e.g. scoreboards, registers)
- Integrating a VIP to an env takes time

VIP Implementation

- Reuse between VIPs (usage of same templates)
- Verification of VIP (separate from the DUT/project)
  - VIP checks verification
  - Scoreboard between BFM and monitor
  - Usage of coverage as criteria for verification completion
- Build so that it will support current and future topologies (e.g. both regular AHB bus and Multi-Layer envs)
- Development can be split into phases to allow usage before completion (e.g. BFM + monitor -> checks -> coverage)

Block to Full Chip Reuse

- Requires strict methodology (sVM ?)
- Reuse of environments from the block level to the chip level almost as is
- Passing information by means of ports (channels)
- Keep modular structure – separate generator and driver from the rest of the environment
- Maintain common reset and init flow in all blocks

Project to project Reuse

- Requires common methodology in both projects
  - Hard to come into agreement about that
  - Lack of code understanding causes changes not in line with the methodology
  - Solution – reuse of code pieces by means of copy paste
    - Data structures
    - Complex functions
- Reuse phase – part of the development process
  - At the end of the project see which parts can be reused also to other projects and if worth, invest in their reuse

Legacy Code Reuse

- Often driven by management
- Gives an illusion of a contribution to faster development
- Gives excuses to a developer
  - "I don't fully understand the code and cannot be sure it will work after my changes"
  - "I don't know why they implemented it this way, they probably had their reasons"
- Bad code can cause a lot of frustration among engineers, bring old bugs to a new project