Functional Verification of Microprocessors
Verification: Why needed?

• Effectively finding bugs from the early design stage
  – Microprocessor designs usually don’t rely on ASIC approach. Finding bugs at later stages makes it difficult to fix them.

• Achieve the confidence level for the tape out
  – Also, cover reasonable coverage goals for each block and at the full chip at each integration milestone

• Bug-free @ chip arrival
  – Practical goal: no show-stoppers @ chip testing upon arrival

• Infrastructure to reproduce post-silicon issues and debug them efficiently
  – Come up with robust and flexible test regression suite including static/random
Big Question

• To cover the combination of potential behaviors of all the gates inside the microprocessors nowadays will require almost infinite space

• So, how much functional testing do you need to finish before taping out the design…?
When Can You Tape Out?

- There is no golden rule. Have seen extreme cases.
  - If time-to-market is too critical, then bare minimum testing can be done taking the risk
- However, it will be very risky to tape out microprocessor design with completing the moderate level of testing.
  - Debugging issues with silicon is indeed painful. Non-trivial to figure out what’s happening inside the actual silicon
  - Mixed problems may exist on silicon such as process related and electrical problems. This will slow down the progress since they need to be filtered out first to deal with the pure functional problems
How to Build Verification Team

• Choices
  – Engineers do design and verification helped by dedicated infrastructure engineers
  – Dedicated verification team separated from the design team

• May consist of several sub-groups depending on the scale of the project
  – Block level vs. Full-chip level
  – Core level vs. SoC level
  – DFT
Design/Verification Task Partition

- Design Team (in terms of functional validation)
  - Come up with the detailed microarchitecture specification (and/or PRM) and the corresponding review
  - Joint review of the test plan
  - RTL coding and code linting
  - Strict version control such as checking in the design files always after passing the certain pre-defined test suite (partial or whole suite)
  - LEC
  - In-line assertions
  - Various property checks (depending on methodology)
  - Analyze code coverage meters (line/condition fsm...)
Design/Verification Task Partition

• Verification Team
  – Joint review of microarchitecture specification (and/or PRM)
  – Writing the test plan and hold the corresponding review
  – Build test environment with stimulus/diags, monitors, checkers, assertions, etc
  – Perform various test regressions such as static and random regressions
  – Strict version control such as checking in the test environment files always after passing the certain predefined test suite (partial or whole)
  – Write functional coverage objects and achieve the predefined goal for functional coverage
Verification Engineers

• Typical Requirements
  – HDL
  – C/C++ languages
  – Verification languages
  – Scripting skills
  – Assembly languages
  – Experiences with simulation tools
  – Understanding of RTL design
  – Documentation and communication skills
  – (Optional) Co-simulation experience
Block Test Environment

• Stand-alone test is to speed up the block level testing
  – Is this really needed? Indeed, it is up to your discretion
    • Pros: Stress more effectively. Potential benefits for reproduction of post silicon failures
    • Cons: Expensive. More costs in terms of time are involved to synchronize with other test environments when many files are commonly shared
  – If you choose to do it, then how much will you need to do?
• Core testing is based on assembly language based diags which check architecture states against simulator results. Core can also break down into multiple stand-alone testing
• When SoC level testing is involved, hierarchical stand-alone testing environments can be built

October 2006
Full Chip Testing

• Usually, “huge” infrastructure needs to build to model the system. This means that it is extremely difficult to maintain the environment

• Need to be done in multiple phases
  – Design won’t be ready soon
  – Building and testing diags/benches will also take time to implement
  – Need to sync up with project milestones

• Need to focus on setting up strategy how to uncover issues that block level testing can not cover, and execute the plan aggressively
Typical Verification Stages

• Test plan and review
• Test environment
• Directed (or static) testing
  – Black box testing
  – White box testing. Completion of this stage means the block level test regression set is ready
• Random regression
• Functional coverage
Random Testing

- Strategy review
- Come up with random templates
- Come up with knobs to randomize
- Randomly tweak knobs based on the weights chosen by the user
- Better if combined with coverage measures to monitor the progress
Typical Validation Categories

- Interface rule check
- CSR access check
- Main functionality testing
- Error testing
- Reset/interrupt related check
- DFT related testing
- Gate regression
- Performance related testing
Hardware Emulation

• Benefits
  – CPS improvement
  – Higher confidence level by achieving a lot of system level testing ahead

• Issues
  – Very expensive
  – Sometimes, even a big one cannot fit the whole design
Infrastructures

- Various validation tools for simulation, waveform analysis, assertion, coverage, etc
- Version control tool
- Release mechanism
- Bug tracking tool
- Regression computing resources and corresponding management
- Automatic regressions (mini, daily, weekly)
- Web-based test reporting
Review for Completeness

- Test plan execution status
- Bug rate/type/discovery mechanism
- MTBF is a good measure to decide when to chill or freeze the design
- Failure depth (i.e., corner case?)
- Code coverage
- Functional coverage
Challenges to Overcome

• Designs tend to be growing bigger and bigger
  – The CPS will become a major issue for the full chip level testing

• The more you integrate, the more you need to test
  – More blocks to test and longer period of testing which can be the bottleneck of the project
  – More resources, more expertise (which ends up with multi-site operation), and more management