Maximizing Formal ROI through Accelerated IP Verification Sign-off

Scott Peverelle, Hao Chen, Kamakshi Sarat Vallabhapurapu, Rosanna Yee, Hee Chul Kim, Johann Te, Jacob Hotz
Introduction

• The value of formal verification in ASICs has been recognized across the industry and is increasing in usage

• Our team concurs and is looking to increase our breadth of formal usage
Problem Statement

• Formal verification can be very powerful
• But we have limited ROI due to relatively low adoption rate
• Low adoption rate typically driven by perception that formal is limited in what it can handle
  • Large designs (# of gates and flops)
  • High sequential depth
  • Lack of engineer expertise to deal with complexity
• For formal to reach its full potential, we need to address these issues
  • Complexity handling techniques (e.g. abstractions)
  • Good partitioning and planning
  • Formal reuse
A Hybrid IP Verification Strategy

• Old verification strategy centered on layers of simulation reuse environments (unit, cluster, IP)

• New strategy: push as much to formal as practical
  • Primarily at unit level
  • Formal at cluster level possible as well

• Leverage simulation and formal’s respective strengths
Why this Hybrid Strategy?

• We want to shift left wherever possible
• Formal can typically start finding bugs earlier than simulation
• Handles unit level designs better (usually) where verification typically starts earliest
• Simulation (or emulation) better handles the very deep sequential cases that often need to be verified at IP level
Our Project Life Cycle
Formal Shift Left vs. Dynamic Verification

- Hybrid approach using FV on unit level finds bugs faster
- DV alone may not find all bugs
- Bugs it does find are usually found later -> greater schedule impact
Where is Dynamic Verification Still Useful?

• Reminder that we are proposing a hybrid methodology, not FV only
• FV is great but still has limitations even with advanced techniques
• Focus DV on areas FV struggles with
  • Big clusters or IP level (large gate count)
  • Long sequences to be verified
• Examples:
  • PCIe or media PHY training and linkup
  • Bandwidth measurements
  • IP or SoC power state transitions
  • FW and ASIC co-simulation (could also be done in emulation)
Comprehensive Signoff Methodology

With Microarchitects
- Verification Planning

With Designers
- FV Initial Setup
- RTL Coding + FV Bringup
- FV Exploration
- FV Signoff
FV Planning in Hybrid Context

• Important at planning stage to carefully pick where FV will be applied versus other techniques to cater to strengths
• Define unit level and cluster level verification environments
• Think about potential for FV reuse
  • Reuse in other FV environments
  • Reuse of some FV code in DV
Hybrid Planning Example
Parallel RTL Coding and FV Bringup

• As mentioned earlier, we propose doing FV bringup at the same time RTL is still being coded
• Inspired by test-driven development concept
• Aggressive shift left with corresponding schedule benefits
• Immediate feedback to designers helps reduce amount of re-coding when an issue is found
• Designers are co-owners of FV bringup tasks and environment and work closely with FV experts
Unit Level Exploration and Signoff

• Formal signoff criteria for FPV
  • 100% functional coverage hit
  • No failing checkers
  • All assertions are fully proven or bounded proven past relevant coverage sequential depth

• Additional formal apps are leveraged as appropriate for signoff
  • SEC for dynamic clock gating equivalency
  • CSR for blocks with registers
  • XPROP for all blocks
  • Connectivity on cluster and chip level
  • CDC app for blocks with CDC crossings; includes FPV with metastability injection
Scope of Formal Apps
Formal Reuse

• Reuse can occur in another formal environment, or in a simulation environment
• Use of FVC modes are key to enabling this in our methodology
• Benefits:
  • Validation of unit level assumptions
  • Reducing duplication of modeling and coverage code
Unit Level FV Architecture

FVC Mode = Active

End2end constraint
- E2E assume

Checking Model
- E2E assert

Coverage model
- E2E cover

Legend
- FVC component
- Assert prop
- Assume prop
- Cover prop
FVC Reuse Example

- **FVC Reuse Example**
  - **End2end constraint**: E2E assume
  - **Checking Model**: E2E assert
  - **Coverage model**: E2E cover

  **Legend**:
  - FVC component
  - Assert prop
  - Assume prop
  - Cover prop
Architectural FPV

- Another optional technique that can be applied once FVCs have essential checking and modeling coding ready
- Does not rely on RTL being available
- Can catch cluster level issues very early (shift left)
- Constraints will be verified by the formal reuse mechanism
FVC Configuration for Architectural FPV
Results – Shift Left

• FV on block level (and arch FPV) catches issues much earlier
Results – Shift Left Unit Example

• Results from one of our unit FV environments from the target IP
• A deadlock issue was identified the same day RTL for it was coded
  • Test environment was already available -> just run the checkers with the new RTL
  • Quickly verify if the fix works
• Spec issues were identified quickly as well
  • Incorrect calculation of an address in spec documents
  • Led to a violation of a spec requirement’s associated checker on return type for an access to this space
  • We have found issues like this with DV in past projects, but only much later
Results – Quality

• We are also able to locate ‘super-bugs’ with this flow that are otherwise very difficult to find

• Architectural FPV helps with this, and has highest ROI when done early

• On our cluster environment, we were able to prove absence of deadlock for certain cases despite RTL having large flop count and sequential depth
Deadlock Proof Example

- Arch FPV was done on a cluster level
- Cluster contained many unit models communicating overall several protocols
- End-to-end checkers were used to prove absence of deadlocks
- DV approach would not have been exhaustive
Results – Quality

• Flow can also catch difficult bugs in interactions with 3rd party IP
• Simulation often finds these very late if at all
• Issues like these have been a problem on past projects
• On latest project, found some such examples in formal before IM1 milestone
• Only need the 3rd party RTL and some protocol checkers and modeling code to catch issue
Other Results – Team Growth

• Lack of formal expertise is a major barrier to greater formal adoption
• Our approach provided many opportunities for team members to try formal for the first time and get comfortable with it
• Majority of verification engineers on IP were doing FV at some point
• For many, first time on a real project
• Also resulted in learnings regarding formal reuse, architectural FPV
Future Work

• Create more common abstraction models, not just FBMs for reuse across unit level models

• Continue to build team capability, have more engineers who are capable of more advanced FV work
  • Arch FPV
  • Advanced complexity reductions
  • Formal signoff using a range of apps

• Refine our planning
  • FV versus DV division partitioning
  • Maximize reuse potential
  • Get Arch FPV started earlier for best ROI
Conclusion

• Hybrid methodology greatly expanded scope of formal usage with substantial quality and schedule benefits
• We can leverage FV and DV each where they are strongest without verification gaps
• Many more team members gained experience with formal with expected benefits to future projects
Questions
Contact Information

scott.peverelle@intel.com