A Mutually-Exclusive Deployment of Formal and Simulation Techniques Using Proof-Core Analysis

Keerthikumara Devarajegowda, Jeroen Vliegen
Infineon Technologies AG
Goran Petrovity, Kawe Fotouhi
Cadence design Systems
Agenda

• Problematic
• Proposed Flow
• Application on a Mixed-Signal Design
  – Verification Split
  – Formal Verification Flow
  – Simulation Reductions
• COV DB Merging
• Conclusion
Agenda

• **Problematic**
• Proposed Flow
• Application on a Mixed-Signal Design
  – Verification Split
  – Formal Verification Flow
  – Simulation Reductions
• COV DB Merging
• Conclusion
Usage of Formal & Simulation on the same Design

- **Simulation only**
  - Usage: 80-100%
  - Solid UVM methodology
  - Scales from IP up to SoC
  - ...
  - Usage: 10-20%
  - Hunt for corner cases
  - Prove SoC connectivity
  - ...
  - Usage: 5-10%
  - @ IP boundary
  - Automated with APPS
  - ...

- **FV on top of SIMU**

- **FV only**
Given an IP with:
4 basic and 1 complex feature

Case: formal on top of simulation
• all features are verified with simulation
• C1 is checked in addition with formal

We have:
- increased efforts
+ increased quality
Formal Mutually-Exclusive to Simulation

Case: formal mutually exclusive to simulation
• Features F1, F2, F3 are simulated
• Features F4, C1 are formally proven

We have:
+ reduced simulation efforts
+ increased quality
Obstacles for going Mutually Exclusive

- Making a smart decision on the split requires expertise
- Features may traverse module boundaries
- APP automation might be missing
- ...
- Assessing the simu & formal results to conclude whether we are „done“ might not be trivial
- A uniform and profound completeness metrics analysis spanning across formal and simulation is often in-existent
- ...

© Accellera Systems Initiative
Agenda

• Problematic
• **Proposed Flow**
  • Application on a Mixed-Signal Design
    – Verification Split
    – Formal Verification Flow
    – Simulation Reductions
• COV DB Merging
• Conclusion
The flow on the left was created within the scope of a master thesis as a Proof-of-Concept in order to combine formal and simulation on the same design.

So far it has not been integrated in the Infineon design flow yet.
Agenda

• Problematic
• Proposed Flow
• **Application on a Mixed-Signal Design**
  – Verification Split
  – Formal Verification Flow
  – Simulation Reductions
• COV DB Merging
• Conclusion
DUT & VE

- DUT: ADAS multi voltage safety system supply
- SIMU: UVM SystemVerilog, NCSIM/Xcelium
- Property language: SVA
- Formal Tool: JasperGold
Choosing Formal Friendly Blocks

- **Concurrent blocks**
- **Sequential blocks with low sequential depth**
- **Control blocks**
- **Data Transfer blocks**
- **Data Transform blocks with less intensive arithmetic operations**

**Type of Design-block**
- Sequential blocks with high sequential depth
- Data Transform blocks with intensive arithmetic operations

**Type of Operation**
- Data Transform blocks with less intensive arithmetic operations
- Size of Design-block (# flip-flops)

Consider Capabilities of formal engines, Bounded-proofs & Abstraction techniques!

**Flowchart**:
- define FV $\Rightarrow$ SIMU split
- create vPlan
- Formal Prove
- Simulation
- ProofCore Analysis
- COV collection
- merge & analyze COV DB
- iterate until 100%
Chosen Blocks for Formal

- SPI interface
- Register files
- SECDED logic
- PROT logic
- DEVCTRL logic
- (potentially more)
Verification Plan

- Feature tree
- SIMU perspective
  - Covergroups
  - Checks
- FV perspective
  - Assertions
- Assume-Guarantee
  - left out (simplicity)

FV Perspective

SIMU Perspective

- F3: SMPS
  - check_m
  - covergroup_n

- F4: SPI interface
  - assert_x
  - assert_y

- F5: SECDED
  - assert_p
  - assert_q

- ADAS supply

- define FV ⇨ SIMU split
- create vPlan
- Formal Prove
- Simulation
- ProofCore Analysis
- COV collection
- merge & analyze COV DB
- iterate until 100%
Agenda

• Problematic
• Proposed Flow
• **Application on a Mixed-Signal Design**
  – Verification Split
  – **Formal Verification Flow**
  – Simulation Reductions
• COV DB Merging
• Conclusion
- We used JasperGold for the formal prove
- The register files were proven with the Control and Status Register App
- All other properties were handwritten
- IP-XACT automatically extracted (IFX automated flow)
- 62 regs -> ~3.7K props
- Several access policies used, volatile accesses, ...
Handwritten properties

- Written 140 props (by hand)
- Proven in ~16 minutes
Completeness

- Review vPlan
- Review Properties

Functional

Structural

- Reachability coverage (deadcode)
- Bounded proof coverage
- ProofCore coverage

DUT

CSR APP

Manual Properties

COV App

© Accellera Systems Initiative
Given:

- Property $P$
- $\text{COI} = P \ I_1 \ I_n$

Proof Process:

- Abstract the COI $\rightarrow A_x$
- Prove $(P)$ on the abstracted COI
  - if $P$ fails $\rightarrow$ CEX
    - Prove $(P)$ on full COI
    - if CEX $\rightarrow$ true negative
  - else $\rightarrow$ extend abstraction
- if $P$ holds
  - $\text{ProofCore} = P \ A_m$
ProofCore

**Given:**
- Proofcore \( P \) \( \text{Am} \)

**Structural Metrics:**
- RTL code coverage
  - Today
    - Block
    - Statement
    - Expression
  - Coming
    - Toggle
Agenda

• Problematic
• Proposed Flow
• Application on a Mixed-Signal Design
  – Verification Split
  – Formal Verification Flow
  – Simulation Reductions
• COV DB Merging
• Conclusion
Simulation Reductions

Regmodel updating uses:
- UVM reg predictor: `update()`, `mirror()`
- SV ref model: `predict()`
Simulation Reductions

First remove:
- the UVM reg predictor: update(), mirror()
- the SV ref model: predict()

Next change the regmodel updating to:
- Hook the reg model directly to the HDL regs using add_hdl_path()
- Use $nc_mirror to detect HDL changes
- Upon a change use peek() to update the reg model
Simulation Reductions

- add SVA assertions to check
  - RTL volatiles (status, flags, ..)
  - versus SV ref model (former predict())
- SVA is bi-directional!
  - [RTL volatile] implies [SV ref]
  - [SV ref] implies [RTL volatile]

++ no more SPI reads needed to check status!
++ SVA is more precise! (quality ++)
# Simulation Reductions

<table>
<thead>
<tr>
<th>ENV</th>
<th>SPI</th>
<th>SECDED</th>
<th>DEVCTRL</th>
<th>PROTMGR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COV</td>
<td>COV</td>
<td>COV</td>
<td>COV</td>
</tr>
<tr>
<td></td>
<td>GEN</td>
<td>GEN</td>
<td>GEN</td>
<td>GEN</td>
</tr>
<tr>
<td></td>
<td>REF/CHK</td>
<td>REF/CHK</td>
<td>REF/CHK</td>
<td>REF/CHK</td>
</tr>
</tbody>
</table>

- reduce (remove)
- reduce (remove)
- reduce (remove)
Agenda

- Problematic
- Proposed Flow
- Application on a Mixed-Signal Design
  - Verification Split
  - Formal Verification Flow
  - Simulation Reductions
- **COV DB Merging**
- Conclusion
COV DB Merging

- Define FV \(\Leftrightarrow\) SIMU split
- Create vPlan
- Formal Prove
- Simulation
- ProofCore Analysis
- COV collection
- Merge & analyze COV DB
- Iterate until 100%
## Confirm SIMU ↔ FV split

### Simulation Coverage Checks

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
<th>Checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>multiEngineMDV</td>
<td>52.03%</td>
<td>59.3%</td>
</tr>
<tr>
<td>1 Interfaces</td>
<td>56%</td>
<td>56%</td>
</tr>
<tr>
<td>1.1 UART</td>
<td>68.97%</td>
<td>58.97%</td>
</tr>
<tr>
<td>1.2 AHB</td>
<td>47.83%</td>
<td>47.83%</td>
</tr>
<tr>
<td>2 Memory Transactions</td>
<td>100%</td>
<td>50%</td>
</tr>
<tr>
<td>3 Assumption Validation</td>
<td>88.89%</td>
<td>88.89%</td>
</tr>
<tr>
<td>4 Registers</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1 Jasper CSR (automatic vPlan)</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.1 Rx_Buffer_1</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.2 IER</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.3 IR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.4 LCR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.5 LSR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.6 MSR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>5 Code Coverage</td>
<td>52.94%</td>
<td>r/a</td>
</tr>
<tr>
<td>5.1 Jasper UNR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>5.2 uart</td>
<td>61.29%</td>
<td>r/a</td>
</tr>
<tr>
<td>5.3 npb_subsystem</td>
<td>51.9%</td>
<td>r/a</td>
</tr>
<tr>
<td>6 Integration Testing</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>6.1 Jasper CONN</td>
<td>r/a</td>
<td>r/a</td>
</tr>
</tbody>
</table>

### Formal Coverage Checks

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
<th>Checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>multiEngineMDV</td>
<td>76.66%</td>
<td>81.15%</td>
</tr>
<tr>
<td>1 Interfaces</td>
<td>76%</td>
<td>41.89%</td>
</tr>
<tr>
<td>1.1 UART</td>
<td>96.55%</td>
<td>41.29%</td>
</tr>
<tr>
<td>1.2 AHB</td>
<td>63.64%</td>
<td>13.04%</td>
</tr>
<tr>
<td>2 Memory Transactions</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>3 Assumption Validation</td>
<td>88.89%</td>
<td>r/a</td>
</tr>
<tr>
<td>4 Registers</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1 Jasper CSR (automatic vPlan)</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.1 Rx_Buffer_1</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.2 IER</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.3 IR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.4 LCR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.5 LSR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>4.1.6 MSR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>5 Code Coverage</td>
<td>73.12%</td>
<td>r/a</td>
</tr>
<tr>
<td>5.1 Jasper UNR</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>5.2 uart</td>
<td>83.29%</td>
<td>r/a</td>
</tr>
<tr>
<td>5.3 npb_subsystem</td>
<td>62.26%</td>
<td>r/a</td>
</tr>
<tr>
<td>6 Integration Testing</td>
<td>r/a</td>
<td>r/a</td>
</tr>
<tr>
<td>6.1 Jasper CONN</td>
<td>r/a</td>
<td>r/a</td>
</tr>
</tbody>
</table>

### Flow Diagram

1. **define FV ↔ SIMU split**
2. **create vPlan**
3. **Formal Prove**
4. **Simulation**
5. **ProofCore Analysis**
6. **COV collection**
7. **merge & analyze COV DB**
8. **iterate until 100%**
Agenda

• Problematic
• Proposed Flow
• Application on a Mixed-Signal Design
  – Verification Split
  – Formal Verification Flow
  – Simulation Reductions
• COV DB Merging
• Conclusion
Conclusion

• bridges the gap between FV and SIMU
• enables a structural analysis
• enables a confirmation of FV ⇔ SIMU split

• we apply an Optimal Mixture of FV & SIMU
• over time, we get a Quality Increase (formal proof)
• over time, we get an Efficiency Increase (simu reductions)
Questions

Thank you for your attention!