Compatible Qualification Metrics for Formal Property Checking

Holger Busch
Senior Staff Engineer Verification
Infineon Technologies
Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin‘s Coverage Feature
- Certitude
  - General set-up
  - Coupling with Onespin
- Experience and Comparison
- Conclusions
Why Qualify Formal Property Sets?

- "Formal properties are exhaustively checked!"
- 100% coverage?

Yes:
- All input combinations implicitly checked by formal provers

No:
- Property assumptions constrain inputs
  (better than constrained randomized simulation: not just seed)
- Property commitments cannot check all outputs
- Single property cannot cover all input-/output behaviour

Properties are developed according to partitioned DUT-function

Task: Guarantee completeness of partitioning
Goals

- **Quality control**
  - Assessment of formal property sets

- **Formal verification management**
  - Progress indication
  - Sign-Off Criteria

- **Handling of mixed verification tool landscape**
  - Directed & constraint driven randomized simulation
  - Formal property checking

- **ISO26262 compliance of automotive µC products**
  - Traceability
  - Documentation of design and verification process
Qualification Approaches for Formal

- **Manual**
  - Review of formal properties

- **Formal completeness checks**
  - Onespin’s gap-free verification methodology
    - Strongest criterion, not related to simulation coverage metrics

- **Formal witness generation**
  - Simulation coverage for witness trace: line, branch
    - Quality of witness?

- **Design mutation**
  - Onespin’s built-in coverage feature Quantify
  - Link to test-bench qualification tool Certitude
Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin’s Coverage Feature
- Certitude
  - General set-up
  - Coupling with Onespin
- Experience and Comparison
- Conclusions
Onespin 360°™ MV

- Bounded model-checker
  - Various proof engines
- Property languages:
  - ITL (Interval Language), SVA, PSL
- Linting
- Consistency checker
  - Dead-code, Stuck@signals, ...
- Property debugger
- Coverage
  - Formal completeness checker
  - Line & branch coverage: “Quantify“
Onespin’s Quantify Feature

- **Pre-analyses**
  - Formal-proof-based identification of dead, constrained, redundant code regions
  - Code reachability by given property set

- **Observation coverage:**
  - Formal proof that code location (assignment) checked
  - Code location observed when proof fails

- **User-guidance easy**
  - Push-button, focusing to code regions possible

- **Result presentation**
  - XML -> UCDB-compatible
  - HTML-visualisation
Onespin’s Quantify Feature
Onespin’s Quantify Feature
Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin’s Coverage Feature
- Certitude
  - General set-up
  - Coupling with Onespin
- Experience and Comparison
- Conclusions
Certitude: Fault Instrumentation

- Fault-instrumentation of HDL-sources
- Check fault detection by test-cases
Certitude: Qualification Phases

- Modelling phase: RTL-code instrumentation by Certitude
  - Different fault models injected into RTL code
  - Top-level entity with additional input vector for individual activation

- Activation phase: Each test-case run once:
  - Activation: test-case stimulus activates fault condition
  - Propagation: fault visible at observation points (DUT interface)

- Detection phase: Analyses for pairs of \{fault test-case\}:
  - Detection: fail of test-case instead of pass
  - Fault-sets: $F_{\text{injected}} \supseteq F_{\text{activated}} \supseteq F_{\text{propagated}} \supseteq F_{\text{detected}}$
  - Iterative detection controlled by Certitude:

- Statistical Approach by Certitude:
  - Metrics computation for statistical samples

- Application to Formal Properties
  - Iterative invocation of property checker for \{fault / formal-property\} pairs instead of simulator for \{fault / test-case\} pairs
Certitude Qualification Flow

Certitude

RTL*

Test-cases/Props

Simulator/Formal Property Checker

Scripts:
- Elaborate RTL*
- Generate fault assumption
- Check
- Write result

Qualification Results

Selection: {fault, test-case/prop}

Result: fail/pass, check-time
Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin‘s Coverage Feature
- Certitude
  - General set-up
  - Coupling with Onespin
- Experience and Comparison
- Conclusions
Iterative procedure:

- Let Certitude select:
  - Property $P$ from set of qualification properties
  - Fault $c$ from current set of non-detected faults
- Add fault assumption to regular property

Regular Property $P$:

\[
\forall c. \ f(c) = 0
\]
\[
\text{ass}(P) \models \text{com}(P)
\]

Property $P$ with enabling of fault $c$:

\[
\text{1hot}(f) \wedge f(c) = 1, \text{ass}(P) \models \text{com}(P)
\]

- Check fault-$c$-enabled Property $P$ in property checker
- Return proof result + run-time to Certitude
  - Fail: fault $c$ detected by Property $P$
- Repeat until Certitude is finished:
  - All faults detected
  - All $\{\text{fault} / \text{property}\}$-pairs exercised for non-detected faults
Overview

■ Motivation
■ Goals
■ Qualification Approaches
■ Onespin‘s Coverage Feature
■ Certitude
  – General set-up
  – Coupling with Onespin
■ Experience and Comparison
■ Conclusions
Experience and Comparison

<table>
<thead>
<tr>
<th>Quantify</th>
<th>Certitude &lt;-&gt; Onespin Qualification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added value: Coverage results useful for productive verification projects</td>
<td></td>
</tr>
<tr>
<td>Design size: Modules with several 10 k locs manageable</td>
<td></td>
</tr>
<tr>
<td>Usage: easy</td>
<td>Usage: less easy at the beginning: Set-up for Certitude and FPC required</td>
</tr>
<tr>
<td>Fault injection: elaborated model</td>
<td>Fault injection: RTL design</td>
</tr>
<tr>
<td>User control: Code regions (focus, skip, exclude) Property set</td>
<td>User control: rich configurability Code inclusion, fault types, density, properties; instance- or module-based</td>
</tr>
<tr>
<td>Restartability: yes (longer setup time)</td>
<td>Restartability: yes (short set-up time)</td>
</tr>
<tr>
<td>Compatibility with simulation: mergeable</td>
<td>Compatibility with simulation: 1:1</td>
</tr>
<tr>
<td>Maturity: Product feature, potentials for improvements</td>
<td>Maturity: Certitude available for many years; recently: scripting for IFX-internal usage for performance optimization</td>
</tr>
<tr>
<td>Licences: regular Onespin prover licenses</td>
<td>Licences: Onespin prover licenses + Certitude license</td>
</tr>
</tbody>
</table>
Overview

- Motivation
- Goals
- Qualification Approaches
- Onespin‘s Coverage Feature
- Certitude
  - General set-up
  - Coupling with Onespin
- Comparison
- Conclusions
Conclusions

- Two simulation-compatible FPC qualification methods for productive usage
- Both handle big IPs and property sets
  - Usage strongly recommended!
- Two different paradigms:
  - Integrated in property checking environment: Quantify
    - Efficient, but closed and vendor-specific
    - Metrics similar to simulation coverage
  - Coupling of separate tools: Certitude-Onespin/Simulators…
    - Open for customization
    - Exactly same metrics for FPC and simulation
  - Potential synergies
Questions ?