Automated Safety Verification for Automotive Microcontrollers

Holger Busch
Infineon Technologies
Contents

• Motivation
• ISO26262
• Hardware Safety Measures
• Formal Safety Verification
• Formal Fault Injection
• Safety-Modifications
• Results
• Conclusions
Motivation

• Growing number of safety applications for μC:
  – Restraint Systems (e.g. Airbag)
  – Electric Power Steering
  – Electro Hydraulic Power Steering
  – Chassis Domain Control
  – ABS/VSC
  – Vehicle Stability Control (ESP)
  – Suspension Control
  – Advanced Driver Assistant Systems
ISO 26262 Standard

- Mass-produced passenger cars (<3.5 t)
- ASIL classification of automotive μC parts
  - Safety concept, with ASIL decomposition
- HW and SW safety measures
- ISO 26262 certification
  - V-model for development process
  - Diagnostic coverage analysis
  - Documentation, requirement traceability
- Safety verification on top of functional verification

Higher product complexity, area and power consumption, higher development and production costs

No choice: ISO-compliance mandatory!
HW Safety Measures

- Error detection/correction codes
- Double/Triple Redundancy
HW Safety Measures (1)

• Alarm architecture

Safety Management Unit

Test Controller

Safety Flip-Flop

Alarm Reduction
• Phases after fault occurrence:
  – Diagnostic test interval until detection
  – Fault reaction time until safe state reached
Formal Safety Verification (1)

- Verify module-part of safety mechanism:
  - Self-test feature
  - Alarm generation

![Diagram showing safety verification process](image-url)
Formal Safety Verification (2)

- Basic formal property scheme:
  - fault occurrence,
  - additional side conditions
  \[ \implies \text{expected reaction} \]
  - Side conditions:
    - No reset; environment constraints,…
  - Expected reaction:
    - Error flag set, alarm pulse sent
    - Protocol violation at module interface
  - Formal property succinct, yet powerful
    - Covers all scenarios
    - If disproven, corner case is shown

Any Behavior Covered

Witness Trace

3/2/2016 Holger Busch, Infineon Technologies
no test \implies \text{no (false) alarm (1)}

\[ \forall t_1: t \leq t_1 \leq t + T_{st} \quad t(t_1) = 0 \implies \text{alarm}(t + T_{st} + T_{dti}) = 0 \]
Formal Safety Verification (4)

\[ \text{test enabled} \Rightarrow \text{alarm (2)} \]

\[ \text{te}(t) = 1 \land \forall t_2: t \leq t_2 \leq t + T_{st} + T_{di} \Rightarrow \exists t_3: t \leq t_3 \leq t + T_{st} + T_{di} \Rightarrow \text{alarm}(t_3) = 1 \]
Formal Safety Verification (5)

no test-alarm (3) ←

Unreachable without fault injection

true ⇒ talarm(t) = 0 (3)
Formal Fault Injection

• Onespin's cut-option for compilation:
  – List of signals to be cut in model
  – Fan-in-part: real behavior according to design (output)
  – Fan-out-pat: behavior arbitrarily constrainable (input)

• Fault models set by formal assumption in property

- Connected (No fault)
- Stuck@0
- Inverted
- Bridged
Formal Fault Injection (1)

\[
\forall t_1: t_1 < t \, be(0)(t_1) \land talarm(t_1) = 0 \land be(1)(t) \\
\Rightarrow \exists t_2: t \leq t_2 \leq t + T_{dti} \, alarm(t_2) = 1 \land \exists t_3: t \leq t_3 \leq t + T_{dti} \, talarm(t_3) = 1
\]
Formal Fault Injection: Flow

- Automatic signal filtering for injection
- Automatic macro generation
- Pre-defined generic properties
Safety Modifications

• Due to changed safety requirements and analyses
  – New safety application requires inclusion of more registers
  – New SW-mechanism allows removal of HW-safeguarding
  – Product derivative for customer with non-safety application
  – Safety measure found to be missing, or superfluous

• Need to ensure:
  – Mission function not affected

• Verification approaches:
  – Rerun regression with complete coverage (-> Certitude, Quantify)
  – Equivalence check focussed on input-output behaviour of modified sub-components
Safety Modifications (1)

- Equivalence checking flow:
  - Generation of wrapper with both design versions
  - Generation of equivalence properties
Safety Modifications (2)

• Property-based equivalence checking:

\[
ipx_o(t) = ipx'_o(t) \land q(t) = q'(t) \implies ipx_o(t+1) = ipx'_o(t+1) \land q(t+1) = q'(t+1) \tag{8b}
\]

\[
reset(t) = 1 \implies ipx_o(t) = ipx'_o(t) \land q(t) = q'(t) \tag{8a}
\]

All corresponding outputs
Safety Modifications (3)

- Injection of correctable errors:

\[
\forall n: n \leq n_c \forall t_1: t_1 < t \quad be(0)(t_1) \land talarm(t_1) = 0 \land be(n)(t) \\
\Rightarrow \forall t_4: t_4 \leq t_4 + T_{dti} \quad ipx_o(t_4) = ipx'_o(t_4) \land q(t_4) = q'(t_4) \\
\exists t_2: t_2 \leq t_2 + T_{dti} \quad alarm'(t_2) = 1 \land \exists t_3: t_3 \leq t_3 + T_{dti} \quad talarm'(t_3) = 1
\]

(9)
## Results - 2436 Fault Locations

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
<th>MemoryPeak (MB)</th>
<th>Proof Time (hh:mm:ss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sff_0_be_alarm_rst</td>
<td>hold</td>
<td>2479.07</td>
<td>00:01:22</td>
</tr>
<tr>
<td>sff_0_be_alarm_step</td>
<td>hold</td>
<td>2250.46</td>
<td>00:02:33</td>
</tr>
<tr>
<td>sff_0_be_no_alarm_rst</td>
<td>hold</td>
<td>2552.25</td>
<td>00:13:44</td>
</tr>
<tr>
<td>sff_0_be_no_alarm_step</td>
<td>hold</td>
<td>14199.30</td>
<td>02:24:11</td>
</tr>
<tr>
<td>sff_0_be_no_talarm_rst</td>
<td>hold</td>
<td>1990.68</td>
<td>00:21:01</td>
</tr>
<tr>
<td>sff_0_be_no_talarm_step</td>
<td>hold</td>
<td>2635.84</td>
<td>00:01:57</td>
</tr>
<tr>
<td>sff_0_be_ok_rst</td>
<td>hold</td>
<td>1597.71</td>
<td>00:00:04</td>
</tr>
<tr>
<td>sff_0_be_ok_step</td>
<td>hold</td>
<td>8280.96</td>
<td>00:00:26</td>
</tr>
<tr>
<td>sff_1_be_alarm_rst</td>
<td>hold</td>
<td>1698.65</td>
<td>00:00:19</td>
</tr>
<tr>
<td>sff_1_be_alarm_step</td>
<td>hold</td>
<td>7812.13</td>
<td>00:01:45</td>
</tr>
<tr>
<td>sff_1_be_talarm_rst</td>
<td>hold</td>
<td>1685.32</td>
<td>00:00:17</td>
</tr>
<tr>
<td>sff_1_be_talarm_step</td>
<td>hold</td>
<td>8544.13</td>
<td>00:01:17</td>
</tr>
<tr>
<td>sff_any_be_alarm_rst</td>
<td>hold</td>
<td>1682.46</td>
<td>00:00:07</td>
</tr>
<tr>
<td>sff_any_be_alarm_step</td>
<td>hold</td>
<td>7598.20</td>
<td>00:04:11</td>
</tr>
<tr>
<td>sff_any_be_talarm_rst</td>
<td>hold</td>
<td>1528.82</td>
<td>00:00:44</td>
</tr>
<tr>
<td>sff_any_be_talarm_step</td>
<td>hold</td>
<td>7546.81</td>
<td>00:01:27</td>
</tr>
</tbody>
</table>
Results (1)

- Limited set of re-usable properties sufficient
- Astronomical # fault combinations exhaustively checked in reasonable time: $2^{**2436}$ in 4 min
- More expensive to prove that no false alarm
- Partitioned fault sets so far not needed
- Typical bugs found:
  - Not all SFF-bits included in alarm reduction
  - Clock domain discrepancies
  - Incorrect SFF-integration
Conclusions

• Exhaustive formal safety verification extremely fast
  – 100% diagnostic coverage proven
• High proof automation by generic pre-defined properties and design-specific generated macros
• Equivalence checks of safety modifications efficient
  – Focused on modified sub-components
  – Complete debugging support of property checker
  – Applicable if no complete regression suite available
• Automatic safety verification functions tailorable to specific HW safety mechanisms