Automatic Firmware Verification for Automotive Applications

Dr. Torsten Andre, Daniel Valtiner
Infineon Technologies Austria AG
Motivation

SW Verification

Safety

Development Time

Quality
Development Time

Partition

Firmware

Hardware
Parallel Development

- **Parallel** development

- **Influencing** development
  - Confirm feasibility of partitioning
  - React quickly if unfeasible
Quality of Verification

- Functional requirements
  - Behavior
  - Non-behavior
- Non-functional requirements
  - Resource constraints (e.g. timing, memory size)

- Fault tolerant time interval (FTTI)
- Timing Limitations
  - Processing delay
  - Acceptable jitter
  - Refresh rates
void safety()
{
    limit_check();
    __asm__('ADD R1, #1');
}

void limit_check()
{
    int16_t value = 1;
    ...
}
ISO 26262

ISO 26262-6:2011 9.4.6
“The test environment for software **unit testing** shall correspond as closely as possible to the target environment. […]”

ISO 26262-6:2011 10.4.8
“The test environment for software **integration testing** shall correspond as closely as possible to the target environment. […]”
Implications in a Nutshell

Pro’s Host Verification
• Many tools to choose from
• Verify C code

Pro’s Target Verification
• Enable parallel development
• Non-functional requirement verification
• Supports verification of native assembler statements
• ISO 26262
• Verify machine code (do not depend on compiler)
Evaluation Criteria

• Comprehensive verification
• Easy to use
• Low overhead
• Fast execution
• Support regression
• FW Unit verification

• FW Unit Integration verification
• Independent of hardware development
• High level of reuse
• Reliability tooling
• Simple result interpretation
Hardware Abstraction

- **Bitmap**: All registers accessible by HW and FW
Verification Tool Architecture

Inputs
- Bitmap description
- ROM image
- Config
- Test spec

HW Abstraction
- Bitmap
- Persistent Memory
- Interrupt Handler
- Timer Events

Test Bench
- ISS
- Logger
- Profiler

Outputs
- Profiler report
- Verification report

Auto generated

© Accellera Systems Initiative 2017
Inputs

• **Bitmap Description**
  – Register specification
  – Auto-generated from central specification

• **ROM image**
  – Binary FW image
  – Compiled from high level language

• **Config**
  – Project specific adaptation
  – Specified once per project
Test Specification

• Input stimuli
• Expected values
• Test definition in Microsoft Excel
  – Well-known tool
  – Reliable libraries available to enable parsing by framework
  – Reference implementation using Excel formulas
  – CSV import from other tools for reference implementation (e.g. Matlab)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Inputs</td>
<td></td>
<td>Expected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Output</td>
</tr>
<tr>
<td>2</td>
<td>algo_value</td>
<td>Lower limit</td>
<td>Upper limit</td>
<td>Limit error</td>
</tr>
<tr>
<td>3</td>
<td>-15</td>
<td>-25</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>-35</td>
<td>-25</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Verification Framework Architecture

Inputs
- Bitmap description
- ROM image
- Auto generated
- Config
- Test spec

HW Abstraction
- Bitmap
- Persistent Memory
- Interrupt Handler
- Timer Events

Outputs
- Profiler report
- Verification report

Test Bench
- ISS
- Logger
- Profiler
- Verify
- Tester
HW Abstraction

• **Bitmap**
  – Automatically generated from bitmap specification

• **Persistent memory**
  – Models specific behavior to retrieve data, e.g. EEPROM paging

• **Interrupt/Event handler**
  – Register behavior based on conditions/time
Test Bench

• **Instruction Set Simulator (ISS)**
  – Allows execution of native FW image
  – Cycle accurate

• **Tester**
  – User defined
  – Maps input stimuli to FW units
**Instruction Set Simulator (ISS)**

- FW enriched with triggers/commands

```c
void safety()
{
    VERIFICATION_CMD(SAFETY_START);
    limit_check();
    regbist();
}

void limit_check()
{
    VERIFICATION_CMD(LIMIT_CHK_START);
    int16_t a = GET(EEP1_LOWER_LIMIT);
    ...
}
```

**Tester**
- Callback
  - Start/end of units
  - Memory access
  - Beacon

**Update ISS state**
- Update bitmap

© Accellera Systems Initiative
Verification Tool Architecture

Inputs
- Bitmap description
- ROM image
- Config
- Test spec

HW Abstraction
- Bitmap
- Persistent Memory
- Interrupt Handler
- Timer Events

Outputs
- Profiler report
- Verification report

Test Bench
- ISS
- Logger
- Profiler
- Verify

Auto generated

© Accellera Systems Initiative
Outputs

- **Profiler Report**
  - Statement coverage
  - Branch coverage
- **HTML Verification Report**
  - Test summary (skipped, pass, fail)
  - Profiling information (# opcodes, execution time)
  - Access order to persistent memory
  - Bitmap access (when, how often, min/max interval)
  - Memory access violations
Application Example

- Application example in the paper
- Two units (algo, safety)
Evaluation Criteria

- Comprehensive verification
  - Easy to use
  - Low overhead
  - Fast execution
  - Support regression
  - FW Unit verification

- FW Unit Integration verification
  - Independent of hardware development
  - High level of reuse
  - Reliability tooling
  - Simple result interpretation
Conclusions

• Verification tool for target architecture
• Safe and high quality verification
• Fast and parallel development