Virtual Hardware-in-the-Loop and Fault Injection using Virtual Prototypes

Marc Serughetti
Synopsys
Presenter: Marc Serughetti

- 18 years experience in embedded software & systems technologies
  - Simulation, development tools, OS/RTOS, systems

- Companies
  - Synopsys/CoWare
  - Wind River/Integrated Systems
  - ABB, Alenia

- Drives product business development in semiconductor and vertical market system companies (Wireless, Consumer, Automotive, etc.)
Renesas Technology & Solution Portfolio
Agenda

- Why Virtual Prototyping in Automotive?
- Automotive Use Cases
  - Software Development
  - Virtual Hardware-in-the-Loop
  - Fault Testing
- Synopsys and Renesas
- Summary
Virtual Prototyping in Automotive

- Software is introducing a new set of challenges
- Growing software content and system complexity needs to be balanced with the cost of testing
- Have you considered:
  - How to deploy more **and** better testing?
  - How to test **corner** cases?
  - How to support **safety standards**?
  - How to maintain or reduce current development **costs**?

“Jaguar announces the recall of 17,600 units only on the UK due to software glitches on the cruise control system“, IEEE Spectrum, October 2011

After thorough unit testing you can expect 1 bug per 1000 lines of code in the final release

- 1 line ~5 bytes, so 1 bug per ~5KB

<table>
<thead>
<tr>
<th>Application</th>
<th>Microcontroller Type</th>
<th>Code Size</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steering Angle Sensor</td>
<td>8 Bit</td>
<td>32KB</td>
<td>7 Bugs</td>
</tr>
<tr>
<td>Low-end Sensor Cluster</td>
<td>16 Bit</td>
<td>128KB</td>
<td>26 Bugs</td>
</tr>
<tr>
<td>Airbag Controller</td>
<td>16/32 Bit</td>
<td>256KB</td>
<td>52 Bugs</td>
</tr>
<tr>
<td>EPS Controller</td>
<td>16/32 Bit</td>
<td>512KB</td>
<td>104 Bugs</td>
</tr>
<tr>
<td>Central Chassis Controller</td>
<td>32 Bit</td>
<td>1.5MB</td>
<td>308 Bugs</td>
</tr>
</tbody>
</table>
What are Virtual Prototypes?

Fast, fully functional software model of systems under development executing unmodified production code

Early Availability

Easier Deployment
What are VDKs?

Software Development Kit that uses a **Virtual Prototype as a target**. Combined with popular debuggers, VDKs provide higher debug/analysis efficiency.

- **Visibility**
- **Controllability**
- **Fast**
- **Non Intrusive**
- **Deterministic**
- **Scriptable**

Better Software Developer Productivity
Virtual Prototypes in the V Cycle

- Fast simulation models of hardware representing MCU, ECU or network of ECU
- Deliver better visibility, control and determinism compared to standard hardware based development environment
- Simplifies software development and system integration, testing and verification
- Scale throughout the development V Cycle
Automotive Use

- Software development: MCAL/Complex Drivers, multicore and AUTOSAR
- Virtual Hardware-in-the-Loop for system integration and test
- Fault injection and test coverage
Complex Driver/MCAL Development & Multicore Software Development

1. Start 12 months before HW is available
2. Faster Debug w/ Higher Visibility and Control
3. Non Intrusive Visibility and Control
4. Deterministic Repeatable Results

AUTOSAR OS AWARENESS
Complex Driver/MCAL Development & Multicore Software Development

1. Start 12 months before HW is available
2. Faster Debug w/ Higher Visibility and Control
3. Non Intrusive Visibility and Control
4. Deterministic Repeatable Results
5. Combine HW and SW Performance Analysis
Complex Driver/MCAL Development & Multicore Software Development

1. Start 12 months before HW is available
2. Faster Debug w/ Higher Visibility and Control
3. Non Intrusive Visibility and Control
4. Deterministic Repeatable Results
5. Combine HW and SW Performance Analysis
6. Expands Capabilities of Existing SW Tools
"In-the-Loop" technologies

**MIL**
- Plant Model
- Control Model

**SIL**
- Plant Model
- Control SW
- PC Target (.exe)

**PIL**
- Plant Model
- Control SW
- Evaluation Board

**HIL**
- Plant Model
- Control SW
- ECU

**Project Timeline**

**HIL Current Limitations:**
- Limited access due to limited number of HIL systems (cost and access)
- Limited visibility and controllability of HW & SW
- Hard to deploy in regression
- Complex to set up
Virtual Hardware in the Loop Flows

Control Design

Control Algorithm

Plant Model

Code Generation

Co-Simulation or Code Generation

Virtual Plant Model

Embedded Software

Virtual HW

Embedded SW

RTE

Drivers

Basic SW

MCU, ECU

Network of ECU

User Interface

SW Debugger

Regression

Integration and Test

Interactive

AUTOSAR aware analysis & scripting

CAN analysis, Calibration
Integrating Simulink: Co-Simulation Flow

Connector Blocks in VP are parameterized with name of counterpart in Simulink

*Signal* denotes a connection that transports values (bool/int/float)
Integrating Simulink: Export Flow

**Simulink Coder**

**Target Language Compiler (TLC)**
- Detects Connector Blocks and generates SystemC wrapper (need to implement templates)

**Exported SPM (C-Code)**

**Export PlantModel.h/cpp**

**Cosim Stub model**

**Import PlantModel**

**Virtual Prototype (VP)**

**Simulink Plant Model (SPM)**

**Signal**

**I/O**

**Core**

**Mem**
Model Libraries & Integration Views

Export flow

Virtualizer

Simulink Model & Library
Virtual HIL Bridges the Gap

- Start early: before ECU HW is available
- Reduce costs: easier deployment and set up costs
- Improve productivity: Non intrusive AUTOSAR aware SW debug and analysis
- Improve quality: more testing coverage more often (scripting for automated regression execution)

Accelerates integration and test
Improve quality through parallel regressions
## Fault Injection Current Techniques

<table>
<thead>
<tr>
<th>Faults injection points</th>
<th>Hardware-based Fault-Injection w/ contact</th>
<th>Hardware-based Fault-Injection wo/ contact</th>
<th>Software-based Fault-Injection</th>
<th>Simulation-based Fault-Injection (RTL /Gate Level)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Limited set of injection points, mainly IO pin level</td>
<td>Internal (soft errors: radiation, EMI)</td>
<td>Only locations accessible by SW (memory, registers)</td>
<td>Full access to HW blocks</td>
</tr>
<tr>
<td>Able to model permanent faults</td>
<td>Yes</td>
<td>No</td>
<td>No (unless explicit HW support)</td>
<td>Yes</td>
</tr>
<tr>
<td>Intrusiveness on the experiment</td>
<td>None</td>
<td>None</td>
<td>High</td>
<td>None</td>
</tr>
<tr>
<td>Observability</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Controllability</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Repeatability</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Experiment speed</td>
<td>Real time</td>
<td>Real time</td>
<td>Real time</td>
<td>Very slow (limits the actual cases where can be applied)</td>
</tr>
</tbody>
</table>
## Fault Injection using Virtual Prototypes

<table>
<thead>
<tr>
<th>Faults injection points</th>
<th>Full access to internal and external HW elements (that have been modeled), as well as SW</th>
<th>Modify the state of the complete system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Able to model permanent faults</td>
<td>Yes</td>
<td>Faults reside in the simulation framework / do not go into release code</td>
</tr>
<tr>
<td>Intrusiveness on the experiment</td>
<td>None</td>
<td>All HW and SW events recorded and correlated</td>
</tr>
<tr>
<td>Observability</td>
<td>High</td>
<td>Triggered by SW, HW or time events</td>
</tr>
<tr>
<td>Controllability</td>
<td>High</td>
<td>Completely deterministic</td>
</tr>
<tr>
<td>Repeatability</td>
<td>High</td>
<td>Run fast complete SW stacks</td>
</tr>
<tr>
<td>Experiment speed</td>
<td>1/10 real time</td>
<td></td>
</tr>
</tbody>
</table>

More efficient fault testing & code coverage to accelerate certification & reduce associated costs
Error Example

Error: data abort due to memory corruption

- ECC detect 2-bit error on memory location
- Cause code execution exception
- Jump to error routine
Fault Injection Scenario Description

- Core runs the SW application for some time (10ms)

- When the next interrupt arrives the core access to data memory (SRAM)

- Accessed memory location has corrupted value due to radiation, ECC module detects a 2-bit error in the store value and flags a error back

- The core enters into an exception, exception trigger recovery SW routine
Scenario Scripting and Outcome

```tcl
# Data abort scenario
source ./scripts/fault_injection/fault_injection_library.tcl
namespace eval fi {
    proc trigger_on_ISR {} {
        SRAM_1 ECC_error_on_next_access
    }
    proc trigger_on-timeout {} {
        trigger once [hardware_event rh850_0/p_extint_b] fi::trigger_on_ISR
    }
    trigger once [time_event 10 ms] fi::trigger_on_timeout
```
Synopsys & Renesas
Synopsys and Renesas Center of Excellence

- Synopsys in automotive: virtual prototyping technologies, global worldwide support and services, long term vision and financial viability
- Synopsys supports a broad set of processor models from Renesas including
  - SH2A
  - SH4A
  - V850
- Synopsys recently announced a joint partnership with Renesas for the RH850 Microcontroller Family

Synopsys Collaborates with Renesas to Advance Software Development Solutions for Automotive Applications

Joint Engineering Team Will Develop Virtual Prototypes to Speed Software Debug and System Testing for Renesas' RH850 Microcontroller-Based Designs
Benefits for Renesas Customers

- Commercial availability of RH850 MCU VDKs and reference designs for specific application area (powertrain, chassis, body)

- Build with Synopsys powerful virtual prototyping tools
  - Modeling
  - Debug, analysis and scripting
  - Integration

- Leverage Synopsys deployment expertise, services and support

- Multi-year & one stop shop!
RH850 Microcontroller Family Focus

Synopsys & Renesas CoE

G3M/G3K core + DMAC
OSTM, WDTA
TSG2-nxt, TAUA,
TAUJ, ENCA, RTCA
SCI, SCIF, CSIH,
SSI, I2C, RQSPI,
RCAN, USB Host
HS, USB Func HS,
SDHI, MMC HS,
Ether MAC x2ch
w/IEEE 1588, Endat
2.2

G3M based
Powertrain VDK

G3M based
Chassis VDK

G3M/G3K based
Body VDK

Peripherals & Core
Reference Platform

Synopsys & Customer Engagement

Derivative G3M
Powertrain
Customer Specific
VDK

Derivative G3M
Chassis
Customer Specific
VDK

Derivative G3M/K
Body
Customer Specific
VDK

Customer Platform
Summary: Getting Started with Virtual Prototyping

Automotive SW Challenges

Growing software content requires **more and better testing** along the V Cycle

Virtual Prototyping

Virtual Prototyping provides a **powerful solution** for SW development, system integration, test and verification

Renesas & Synopsys

Renesas and Synopsys delivers the **models, tools & integrations** for multicore SW development, virtual HIL and fault injection

Deliver Safer Products to Market Faster and Better
Questions?
Please Provide Your Feedback

- Please utilize the ‘Guidebook’ application to leave feedback

- Or, ask for the paper feedback form