Safety and Security Aware Pre-Silicon Hardware / Software Co-Development

Nikola Velinov, nvelinov@ghs.com, Green Hills Software
Frank Schirrmeister, franks@cadence.com, Cadence Design Systems
Agenda

– Challenges
– Safety and Security
– Shift Left
– Benefits
– Questions
Verification Challenge: System & Chip + Software

Source: IBS 2018

VERIFICATION & SOFTWARE

$2^N$
# Automotive Design Chain & Software Challenges

<table>
<thead>
<tr>
<th>Domain</th>
<th>HW Scope</th>
<th>View</th>
<th>SW Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle</td>
<td><img src="image1" alt="Vehicle HW Scope" /></td>
<td><img src="image2" alt="Vehicle View" /></td>
<td><img src="image3" alt="Vehicle SW Scope" /></td>
</tr>
<tr>
<td>Subsystem</td>
<td><img src="image4" alt="Subsystem HW Scope" /></td>
<td><img src="image5" alt="Subsystem View" /></td>
<td><img src="image6" alt="Subsystem SW Scope" /></td>
</tr>
<tr>
<td>ECU</td>
<td><img src="image7" alt="ECU HW Scope" /></td>
<td><img src="image8" alt="ECU View" /></td>
<td><img src="image9" alt="ECU SW Scope" /></td>
</tr>
<tr>
<td>MCU</td>
<td><img src="image10" alt="MCU HW Scope" /></td>
<td><img src="image11" alt="MCU View" /></td>
<td><img src="image12" alt="MCU SW Scope" /></td>
</tr>
</tbody>
</table>

**Vehicle SW**
- 1 Billion lines of code by 2010
- > 2000 functions
- Distributed on 75 ECUs (average)
- Growing inter-dependencies

**Vehicle SW**

![Vehicle SW Diagram](image13)

![Semiconductor Focus](image14)
Software Rework has Major Impact on System Cost

- **exponential growth in SW size and complexity**
- **20.5%, 300-1000x**
- **0%, 9%, 80x**

**70% Requirements & system interaction errors**
- **70%, 3.5% 1x**
- **10%, 50.5% 20x**

**80% late error discovery**
- **20%, 16% 5x**

Major cost savings through rework avoidance by early discovery and correction
A $10K architecture phase correction saves $3M

Rework & certification is 70% of SW cost, and software is 70% of system cost. Thus, 49% of system cost can be attributed to software rework and cert.

Delivery Delays Not Known Until Late into Project Schedule

Where faults are introduced
Where faults are found

The estimated nominal cost for fault removal

Approved for Public Release. Distribution is unlimited.
... and so does Hardware Rework!

... and requires a “Shift Left”

Relative cost of a bug

![Bar chart showing the cost of bugs at different stages: Arch (1), Design (3), Block test (10), System test (30), Customer (100).]

Bugs found late are costly
- Hard to debug
- Limited options to fix
- Increasing mask costs
SAFETY AND SECURITY
Significance of Security in Safety

These Chinese hackers tricked Tesla’s Autopilot into suddenly switching lanes

Published Wed, Apr 3 2019 - 15:17 AM EDT • Updated Wed, Apr 3 2019 - 12:22 PM EDT

Hackers Remotely Kill a Jeep on the Highway—With Me in It

Security

Newb admits he ran Satori botnet that turned thousands of hacked devices into a 100Gbps+ DDoS-for-hire cannon

One moron down, two to go

https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
The Pattern in the Case Studies

• Mechanical functions are replaced by software
  – Lane assist & distance control
  – AD functions

• Traditionally closed systems are becoming connected
  – A pacemaker can communicate with the world
  – A car’s steering and throttle can be influenced via sensors
  – The Internet of Things is all about connectivity

• Safety and security become more entangled
  – A safe system must also be secure
Safety vs. Security

Safety
- The system behaves according to the requirements in all cases
- Protects humans from machines
- A safe system must be secure
- Techniques to achieve safety
  - Requirements traceability
  - Code coverage
  - Comprehensive testing
  - Redundancy

Security
- The system behaves according to the requirements in all cases and does nothing else
- Protects machines from humans
- Security does not imply safety
- Techniques to achieve security
  - Security policy
  - Penetration testing
  - Covert channel analysis
  - Practically all safety techniques
How Does a System Look Like?

- Abstraction and functionality moves from the bottom to the top
- Dependencies move from the top to the bottom
- You can make a “safe system” with “unsafe hardware”
- Can you make a secure system without secure hardware?
  - Who would you trust?
Building a Chain of Trust

Software Abstraction

Chain of Trust

T(0)=device power on

Time

Bootloader

Middleware (FS, USB, TCP/IP, etc)

Operating system

App

App

FOTA

© Accellera Systems Initiative
Building Safe and Secure Software

• Separation is key for both

• Separation in the time domain
  – Paramount for safety – real time behavior
  – Required for security – prevent DoS

• Separation in the memory domain
  – Significant enabler for complex systems
  – Significant cost redactor

• Maintaining flexibility in the system could become difficult
  – The right choice of technology is important
Planning for Safety and Security in Software

• Partitioning system resources
  – Separation allows greater flexibility

• Certifying mixed criticality components
  – SEooC can be leveraged to reduce total cost
  – Safety decomposition
  – Separation is essential

• Security policy enforcement
  – Separation allows you to be cost efficient
  – Can also help for a clean and simple design
The Right Tools for the Right Job

• Safety standards mandate tool qualifications
  – Could be replaced by testing in some scenarios
• Safety standards mandate tools for the whole lifecycle
  – Requirements definition and traceability
  – Code coverage
  – Reports
• Security is more challenging
  – Penetration testing requires sophisticated frameworks
  – Different code analysis tools are available
  – Run-Time agents & utilities for additional security
Hardware Considerations for Safety and Security

Functional Safety Analysis links to the traditional design/verification and implementation flow:
- To include safety mechanisms and meet the HW metrics/ASIL
- Safety metrics, PPA, verification time, automation are all to be considered
Benefiting From Cooperation

• Hardware can greatly reduce the cost of safety
  – Reduce the overall resource demand
  – Replace software safety mechanisms
  – Provide a secure trusted platform

• Use safe hardware as early as possible!
SHIFT LEFT IN VERIFICATION
Shift Left in the V Diagram – Automotive Example

Vehicle requirements
System Design
Subsystem Design
ECU specification

ECU Development 9 to 24 months

Vehicle validation
System verification
Subsystem integration
ECU test

Virtual integration

Vehicle requirements
System Design
Subsystem Design
ECU specification

ECU test

ECU development

Vehicle validation
System verification
Subsystem integration

accellera
SYSTEMS INITIATIVE

DVCON
CONGRESS AND EXHIBITION
EUROPE

2019
Embedded Software Challenges ...

... addressed by Virtual and Hybrid Platforms!

- Register interface out of synch with drivers
- Missing Registers or incorrect register definitions
- Incorrect routing logic or incorrect memory map
- Software Memory Accesses to unmapped memory
- Missing Interrupt events
- Verification of SW drivers in the context of an OS boot
- Embedded SW Programming Errors and bottlenecks
- First time OS Bring up
Integration HW and SW Early

Applications (Basic to Complex)
Middleware (Graphics, Audio)
OS and Drivers (Linux, Android)
Bare-metal SW
SoC in System
System on Chip
Sub-System
IP

Virtual Platforms
Shift Left
SW development on chip

Time for critical bugs in system environment to be removed

Bug rate
RTL becomes stable
Only small gate level changes and ECOs
Production
Post-silicon validation

Spec
RTL-Design and IP Integration and Verification
IP Qualification
Netlist to GDSII
Fab
Post Si
### What Do Users Care About?

<table>
<thead>
<tr>
<th></th>
<th>Time of Availability</th>
<th>Speed</th>
<th>Accuracy</th>
<th>Capacity</th>
<th>Development Cost</th>
<th>Replication Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Earlier is better</td>
<td>Faster is better</td>
<td>More is better</td>
<td>More is better</td>
<td>Less is better</td>
<td>Less is better</td>
</tr>
</tbody>
</table>

**Software Debug**  
**Hardware Debug**  
**Execution Control**  
**System Connections**  
**Bring-Up Time**  
**Value Links** (Power, Performance)
Wouldn’t It Be Nice If “One Tool Would Rule Them All”? … like that German Fable?

• Eierlegende Wollmilchsau
• "egg-laying wool-milk-sow“

• Perfect farm animal uniting several qualities:
  – chickens (laying eggs)
  – sheep (producing wool)
  – cows (giving out milk) and
  – pigs (can be turned into bacon)

• Produces all the daily necessities and is tasty to boot, it is an animal that only has good sides to it …

System Verification Characteristics

- **SDK OS Sim**
  - Highest speed
  - Earliest in the flow
  - Ignore hardware

- **Virtual Platform**
  - Almost at speed
  - Less accurate (or slower)
  - Before RTL
  - Great to debug (but less detail)
  - Easy replication

- **RTL Simulation**
  - KHz range
  - Accurate
  - Excellent hardware debug
  - Little software execution

- **Acceleration Emulation**
  - MHz range
  - RTL accurate
  - After RTL is available
  - Good to debug with full detail
  - Expensive to replicate

- **FPGA Prototype**
  - 10s of MHz
  - RTL accurate
  - After stable RTL is available
  - OK to debug
  - More expensive than software to replicate

- **Prototyping Board**
  - Real-time speed
  - Fully accurate
  - Post silicon
  - Difficult to debug
  - Sometimes hard to replicate
Bare Metal Compute Options

Performance

Arm Server  
X86

Xcelium™ Simulation

NEw 2018

FPGA

Custom Processor

Palladium® Z1 Emulation

Protium™ X1 & S1 Prototyping

Debug Flexibility & Compile Time
Hardware/Software Co-Verification during SoC Design

- Applications (Basic to Complex)
- Middleware (Graphics, Audio)
- OS and Drivers (Linux, Android)
- Bare-metal SW

Virtual System Platform: System-C
Functional Simulation: RTL
HW/SW Emulation: RTL
FPGA Prototyping: RTL

Software based hardware tests

1st Silicon
Board

System on Chip

Architecture Exploration & Spec Definition Phase

Frontend Design & Functional Verification

Place & Route, Tape Out

Chip Production
Silicon Bringup

SoC Development
Fab
Post Si

12 month
4 month
3 month
6 month
Virtual Platforms to the Rescue

Embedded Software Development

- **Instruction Accurate** software model of hardware system, at the transaction-level
- Full programmers view of design
- Runs unmodified target code
- Runs very fast (sometimes faster than real-time)
- Available 6-12 months before silicon or boards, depending on model availability
- Enables early integration of hardware and software, improves quality
- Could provide insight into performance bottlenecks, architectural analysis
- Includes SW stack
- Easy to distribute to many users

Speed, Controllability, Observability, Repeatability
Hardware/Software Integration

Software Debug: C Code

Hardware Debug: RTL

System Debug: Software

System Debug: Hardware

Image sources: Cadence, Arm, Green Hills
Virtual Platform Debug
SystemC/TLM Aware Debug

- SystemC Threads and Methods in sidebar
- Source Code View
- Call Stack
- Debug and Simulation Console
- SystemC Module Hierarchy
- Device Registers with Bit Fields
- SystemC/ C++/C Variable Watch Window
- Activation of SystemC processes

Device Registers with Bit Fields

Activation of SystemC processes
## Logviewer

### Log Viewer Screen

<table>
<thead>
<tr>
<th>Theme</th>
<th>Sim Time (ns)</th>
<th>Time</th>
<th>Target/Code</th>
<th>Message</th>
<th>Initiator/Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:21:57</td>
<td>example_subsystem.dma.tsocket</td>
<td>WRITE to ADDR=0x001014 DATA=0x00000000, 4 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:21:57</td>
<td>example_subsystem.dma.tsocket</td>
<td>WRITE to ADDR=0x001018 DATA=0x00000000, 2 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:21:57</td>
<td>example_subsystem.dma.tsocket</td>
<td>WRITE to ADDR=0x00101c DATA=0x00000050, 4 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:21:57</td>
<td>example_subsystem.dma.tsocket</td>
<td>WRITE to ADDR=0x001020 DATA=0x000500, 2 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:21:57</td>
<td>example_subsystem.dma.tsocket</td>
<td>WRITE to ADDR=0x001014 DATA=0x00000000, 4 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:21:57</td>
<td>example_subsystem.dma.tsocket</td>
<td>WRITE to ADDR=0x001018 DATA=0x00000000, 2 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:22:03</td>
<td>example_subsystem.dma.tsocket</td>
<td>WRITE to ADDR=0x001000 DATA=0x00000001, 4 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>PRO</td>
<td>2016118</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>Process activated</td>
<td>example_subsystem.dma.dma_process</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>READ from ADDR=0x00000000 DATA=0x1234567890abcdef, 64 bytes in 0 s (blocking)</td>
<td>example_subsystem.dma.isocket_1 [Last]</td>
</tr>
<tr>
<td>TLM</td>
<td>2016118</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>Process activated</td>
<td>example_subsystem.dma.dma_process</td>
</tr>
<tr>
<td>PRO</td>
<td>2016118</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>Process activated</td>
<td>example_subsystem.dma.dma_process</td>
</tr>
<tr>
<td>TLM</td>
<td>2016123</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>Process activated</td>
<td>example_subsystem.dma.isocket_2 [Last]</td>
</tr>
<tr>
<td>PRO</td>
<td>2016123</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>Process activated</td>
<td>example_subsystem.dma.dma_process</td>
</tr>
<tr>
<td>TLM</td>
<td>2016123</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>READ from ADDR=0x001000 DATA=0x00000000, 4 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>TLM</td>
<td>2016123</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>WRITE to ADDR=0x00000000 DATA=0x00000000, 4 bytes in 0 s (blocking)</td>
<td>example_subsystem.core.core.pvbus_m</td>
</tr>
<tr>
<td>PRO</td>
<td>2016123</td>
<td>07:22:03</td>
<td>example_subsystem.ddram.socket</td>
<td>Process activated</td>
<td>example_subsystem.dma.dma_process</td>
</tr>
</tbody>
</table>

### Log Viewer Interface

- **File**
- **Theme**
- **Simulation**
- **Windows**
- **Help**

**Theme Selections**

- DMI
- PRO
- SCE
- SCI
- SCW
- TLM
- VIO
Green Hills – Cadence Integration Points for Verification

Hardware Debug: RTL

Software Debug: C Code

Cadence
Green Hills
Example Integration:

MULTI® Advanced Linux® Debugging Xcelium™ Parallel Logic Simulation

Green Hills Software’s MULTI IDE
- C/C++ Debugger
- Fix bugs faster
- Find bugs automatically
- Make sense of complex systems
- Prevent new problems
- Spend more time developing

MULTI & Virtual Platform Demo
- MULTI Debugger for Linux
- Xcelium-Based Arm® SoC Virtual Platform
- MULTI connection to Virtual System Platform

MULTI Connects via Green Hills Probe to Palladium® & Protium™

Advanced Shift Left Software Development

Green Hills MULTI Debugger connected to Arm Versatile Express Platform modeled as Virtual Platform on VSP on Xcelium®
Virtualization with Emulation Enables SW Shift-

Using virtual platforms and hybrids to accelerate SW development and HW/SW validation.

All Virtual
Pre-RTL SW Development

Tests and Benchmarks
OS SoC Drivers

SoC Virtual Platform

IP Hybrid
Pre-SoC IP / Driver Validation & Optimization
RTL – Palladium or Xcelium

Tests and Benchmarks
OS IP Driver

CPU Virtual Platform

SOC Hybrid
Pre-Tapeout HW/SW Validation

Tests and Benchmarks
OS SoC Drivers

CPU Virtual Platform

Emulation, FPGA Proto
Pre-Tapeout Fully Accurate HW/SW Validation

Tests and Benchmarks
OS SoC Drivers

SoC RTL

All RTL, Silicon
Final HW/SW Validation

Tests and Benchmarks
OS SoC Drivers

RTL Models

Virtual Models - VSP

Shift Left

Design Flow

SW stack

RTL Models
Hardware and Virtual/Hybrids

Virtual solutions

Virtual Prototyping

Virtual Emulation

In-Circuit Emulation

Simulation Acceleration

FPGA Prototype

Prototyping

Virtual System Platform (VSP)

Virtual Peripherals

Same Design under Test

Physical Peripherals

DUT

RTL/UVU Simulation

Protium S1
Key Trend: Unified Flow for Emulation and Prototyping
(Source: Toshiba, CDN-Live 2019)

Palladium and Protium used together
Hardware verification and firmware development
Protium at 4.6x of Palladium

Cadence & Green Hills Software
Safe & Secure Aerospace & Defense System Design

- Hardware Debug: RTL
- Cadence & Green Hills Software
- Safe & Secure Aerospace & Defense System Design
- Requirements Engineering
- System Design
- Architecture Design
- Component Design
- Code Development
- Unit Test
- Integration Test
- System Test
- Acceptance Test
- Software Debug: C Code
- Cadence Virtual System Platform
- Cadence Palladium Z1, Protium X1, Xcelium
- Integrality Security Services
- MULTI Integrated Development Environment
### Safety/Security Certified RTOS

#### Safety/Security Certified Embedded Tool Suite

<table>
<thead>
<tr>
<th>Tool Suite</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK OS Sim</td>
<td>Highest speed • Earliest in the flow • Ignore hardware</td>
</tr>
<tr>
<td>Virtual Platform</td>
<td>Almost at speed • Less accurate (or slower) • Before RTL • Great to debug (but less detail) • Easy replication</td>
</tr>
<tr>
<td>RTL Simulation</td>
<td>KHz range • Accurate • Excellent hardware debug • Little software execution</td>
</tr>
<tr>
<td>Acceleration Emulation</td>
<td>MHz range • RTL accurate • After RTL is available • Good to debug with full detail • Expensive to replicate</td>
</tr>
<tr>
<td>FPGA Prototype</td>
<td>10s of MHz • RTL accurate • After stable RTL is available • OK to debug • More expensive than software to replicate</td>
</tr>
<tr>
<td>Prototyping Board</td>
<td>Real-time speed • Fully accurate • Post silicon • Difficult to debug • Sometimes hard to replicate</td>
</tr>
</tbody>
</table>
BENEFITS
More Robust Hardware and Software!

Applications (Basic to Complex)

Middleware (Graphics, Audio)

OS and Drivers (Linux, Android)

Bare-metal SW

System-On-Chip

- Vehicle Processor
- Compute Sub-System
- DSP Subsystem
- AI Accelerator
- Memory Interface
- Sensor Interfaces
- Vehicle Interfaces
- Secure Subsystem

SoC Development

- Architecture Exploration & Spec Definition Phase

- Emulation Chamber

- Functional Bug rate

- Test 1000s of SW Scenarios Before Tape Out!

SoC User (Company B)

Virtual System Platform (Hundreds of users)

FPGA Prototyping (Dozens of users)

Chip Production

Board Bringup

Post Si

Shorten Board Bringup Time!
More Robust Hardware and Software!

*Greatly accelerated Time-to-Market!*

**Applications** (Basic to Complex)

**Middleware** (Graphics, Audio)

**OS and Drivers** (Linux, Android)

**Bare-metal SW**

---

**System-On-Chip**

- Vehicle Processor
- Compute Sub-System
- Sensor Interfaces
- Memory Interface
- Secure Subsystem
- Vehicle Interfaces
- DSP Subsystem
- AI Accelerator

---

**Architecture Exploration & Spec Definition Phase**

**RTOS Enablement & Board Support Package Development**

**Application Software Development**

**Integration & Performance Tuning**

**Chip Production**

**Board Bringup**

**Test & Certification**

**SOP**

**SoC User** (Company B)

**FPGA Prototyping** (Dozens of users)

**Virtual System Platform** (Hundreds of users)

---

**Test 1000s of SW Scenarios Before Tape Out!**
More Robust Hardware and Software!

accelerated Time-to-Market!

SoC Development

Functional Bug rate

Chip Production

Fab

Test 1000s of SW Scenarios before Tape Out!

SoC User
(Company B)

Emulation Chamber

Virtual System Platform
(Hundreds of users)

FPGA Prototyping
(Dozens of users)

SoC Developer
(Company A)

Chip Production
Board Bring up

RTOS Enablement & Board Support Package Development

Application Software Development

Integration & Performance Tuning

Software Development

Improved HW/SW System Integrity & Greatly Accelerated Time-to-Market
Questions

Finalize slide set with questions slide