

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

Nikola Velinov, [nvelinov@ghs.com](mailto:nvelinov@ghs.com), Green Hills Software

Frank Schirrmeister, [franks@cadence.com](mailto:franks@cadence.com), Cadence Design Systems



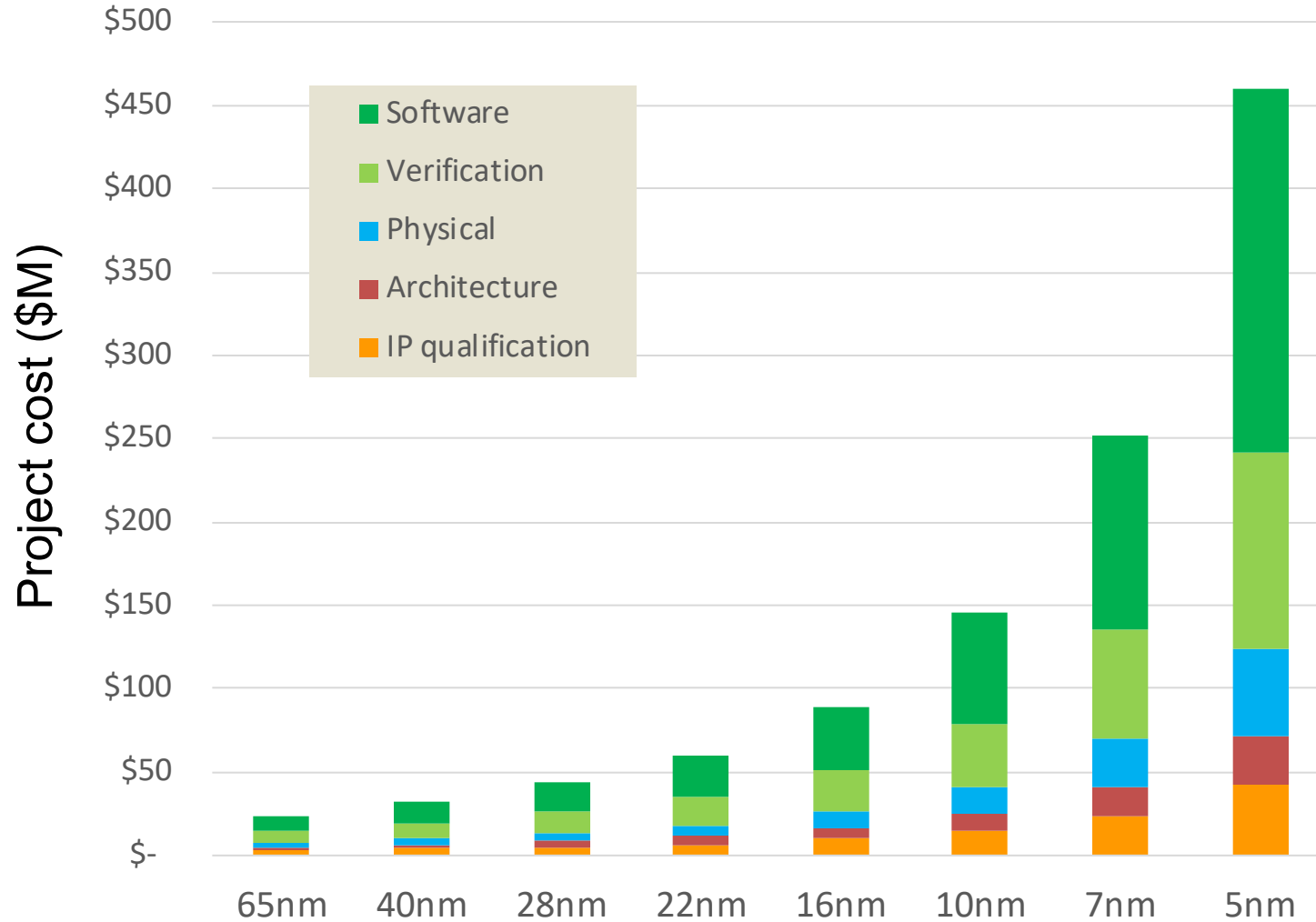
cādence®



# Agenda

- Challenges
- Safety and Security
- Shift Left
- Benefits
- Questions

# Verification Challenge: System & Chip + Software

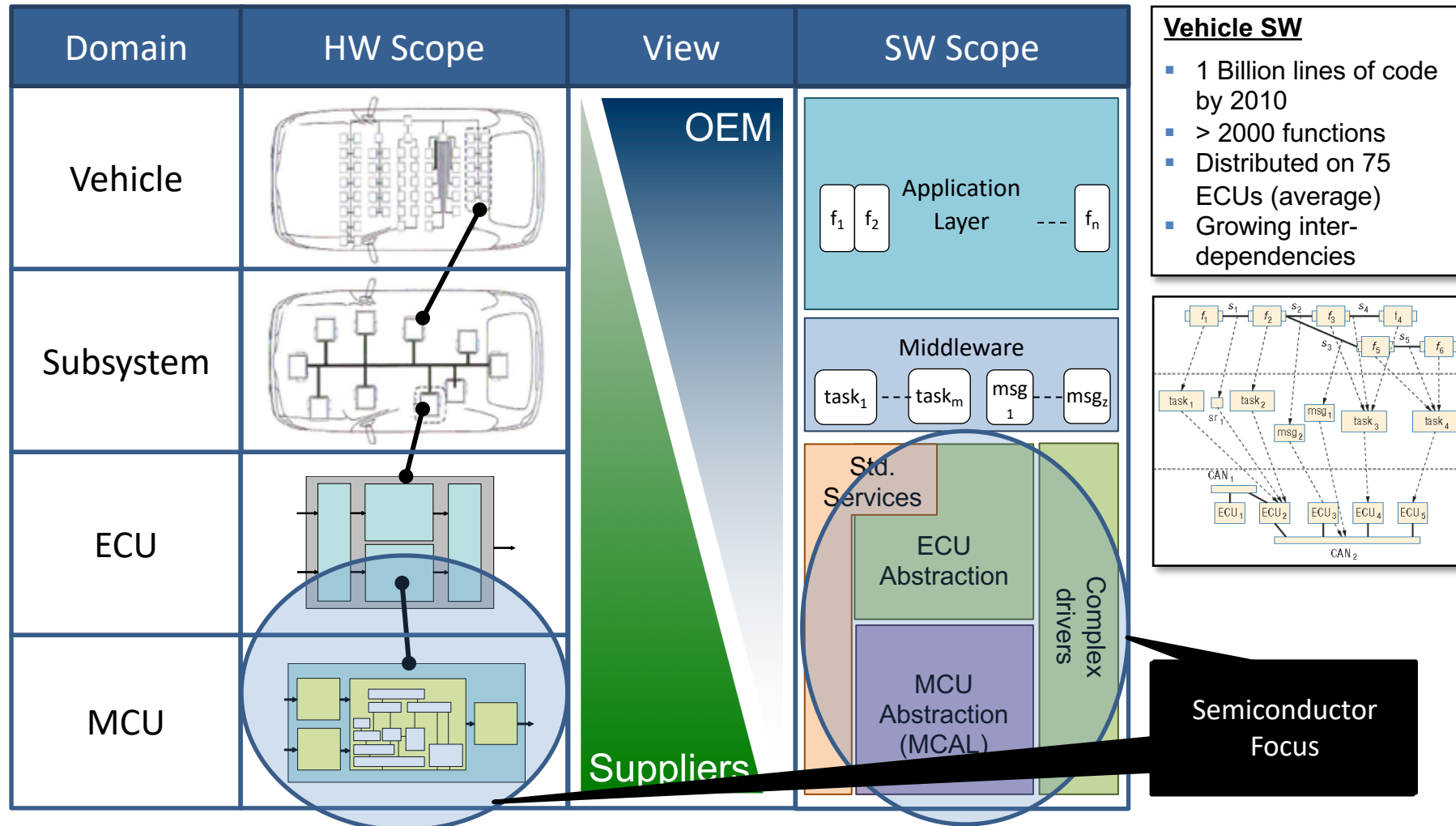


Source: IBS 2018

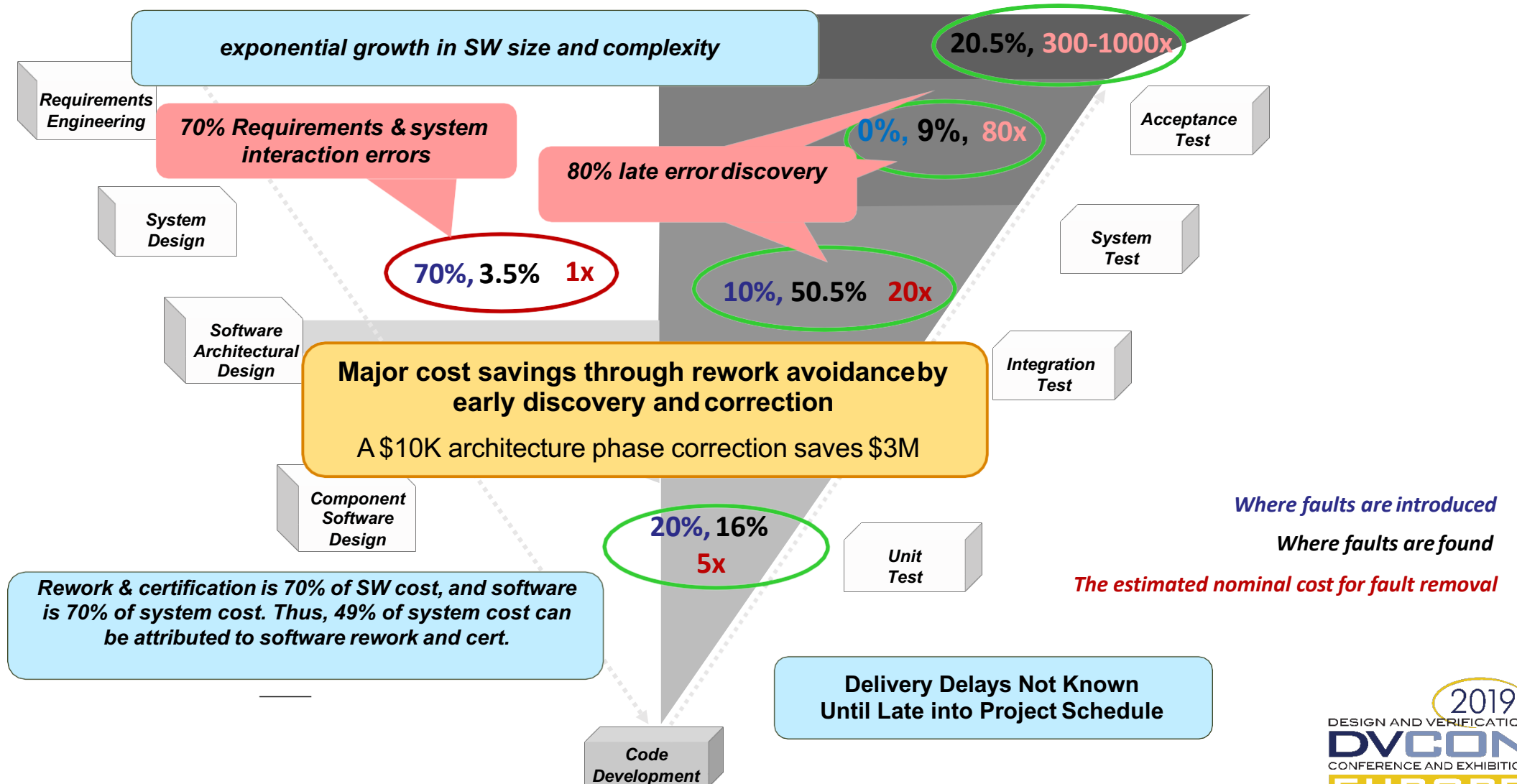
**VERIFICATION  
& SOFTWARE**

**2<sup>N</sup>**

# Automotive Design Chain & Software Challenges



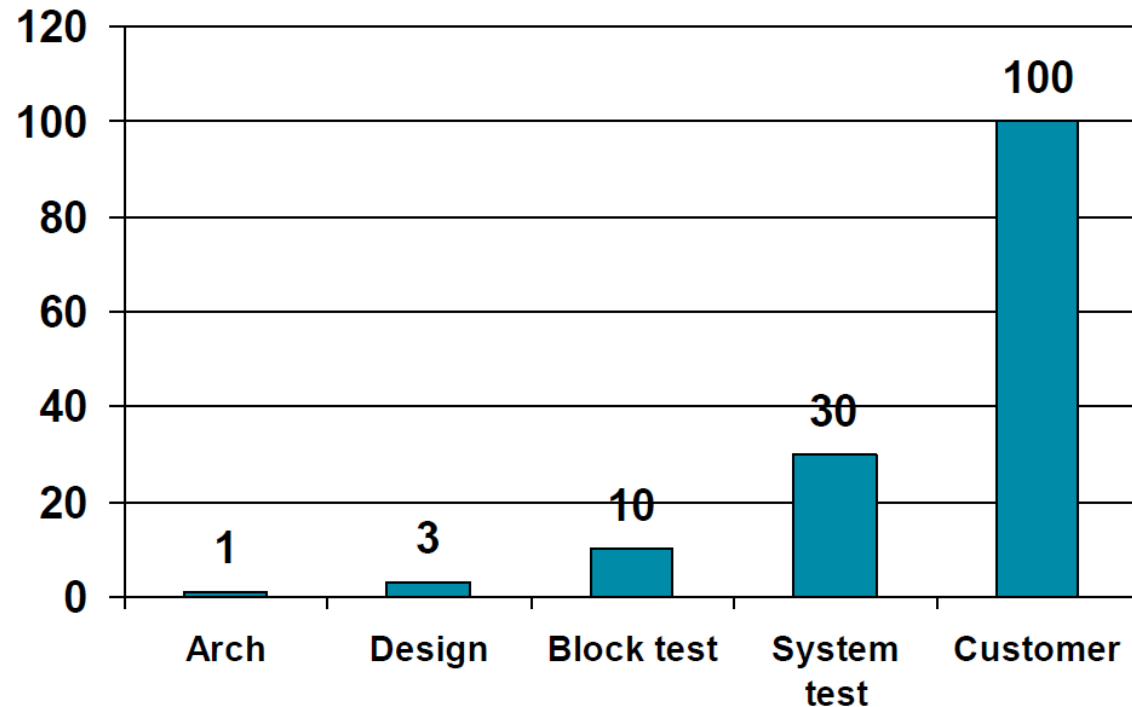
# Software Rework has Major Impact on System Cost



# ... and so does Hardware Rework!

... and requires a "Shift Left"

Relative cost of a bug



Bugs found late are costly

- Hard to debug
- Limited options to fix
- Increasing mask costs

# SAFETY AND SECURITY

# Significance of Security in Safety

FOOD, TRAVEL AND TECH

## These Chinese hackers tricked Tesla's Autopilot into suddenly switching lanes

Published Wed, Apr 3 2019 · 11:17 AM EDT • Updated Wed, Apr 3 2019 · 12:22 PM EDT

LILLY HAY HEWMAN SECURITY 08.09.2018 12:38 PM

## A New Pacemaker Hack Puts Malware Directly on the Device

Researchers at the Black Hat security conference will demonstrate a new pacemaker-hacking technique that can add or withhold shocks at will.

ANDY GREENBERG SECURITY 07.21.15 06:00 AM

## 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

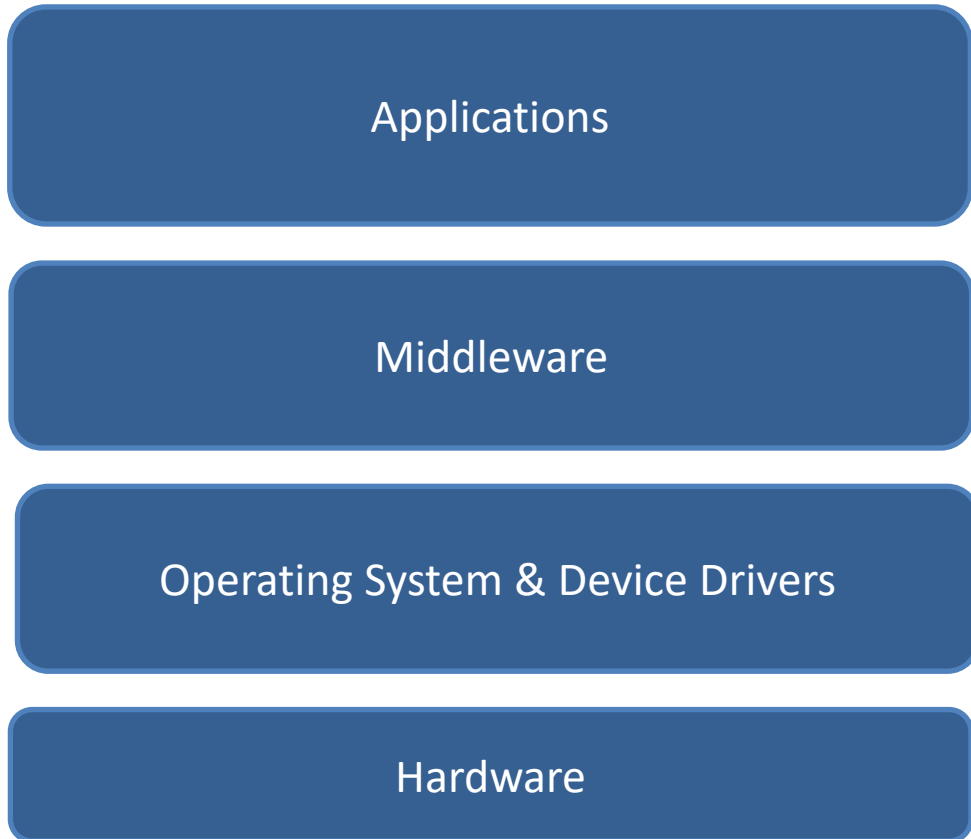
## Safety

- 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

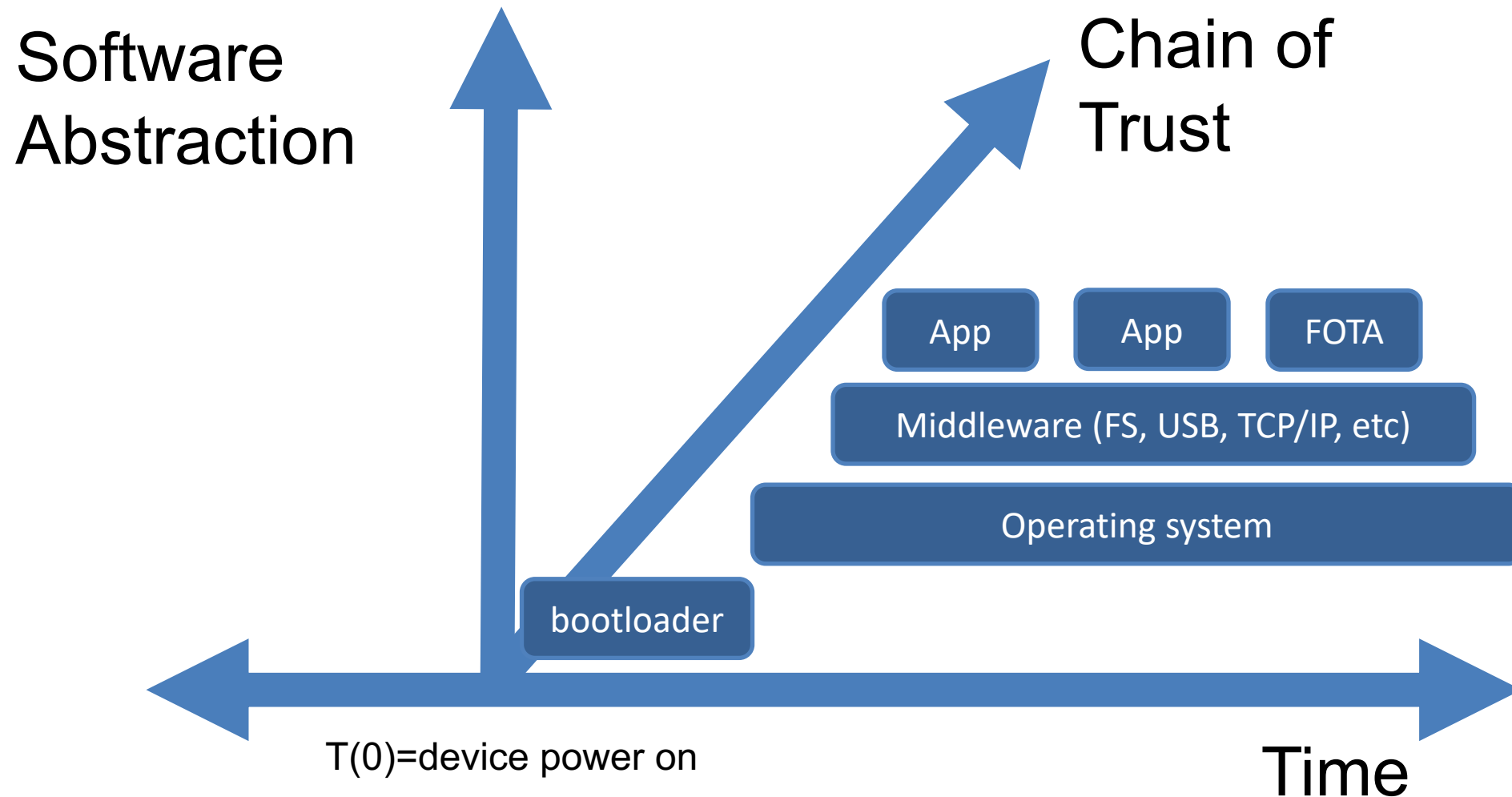
- The system behaves according to the requirements in all cases **and does nothing else**
- Protects **machines from humans**
- **Security does not imply safety**
- Techniques to achieve security
  - Security policy
  - Penetration testing
  - Covert channel analysis
  - Practically all safety techniques

# 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?

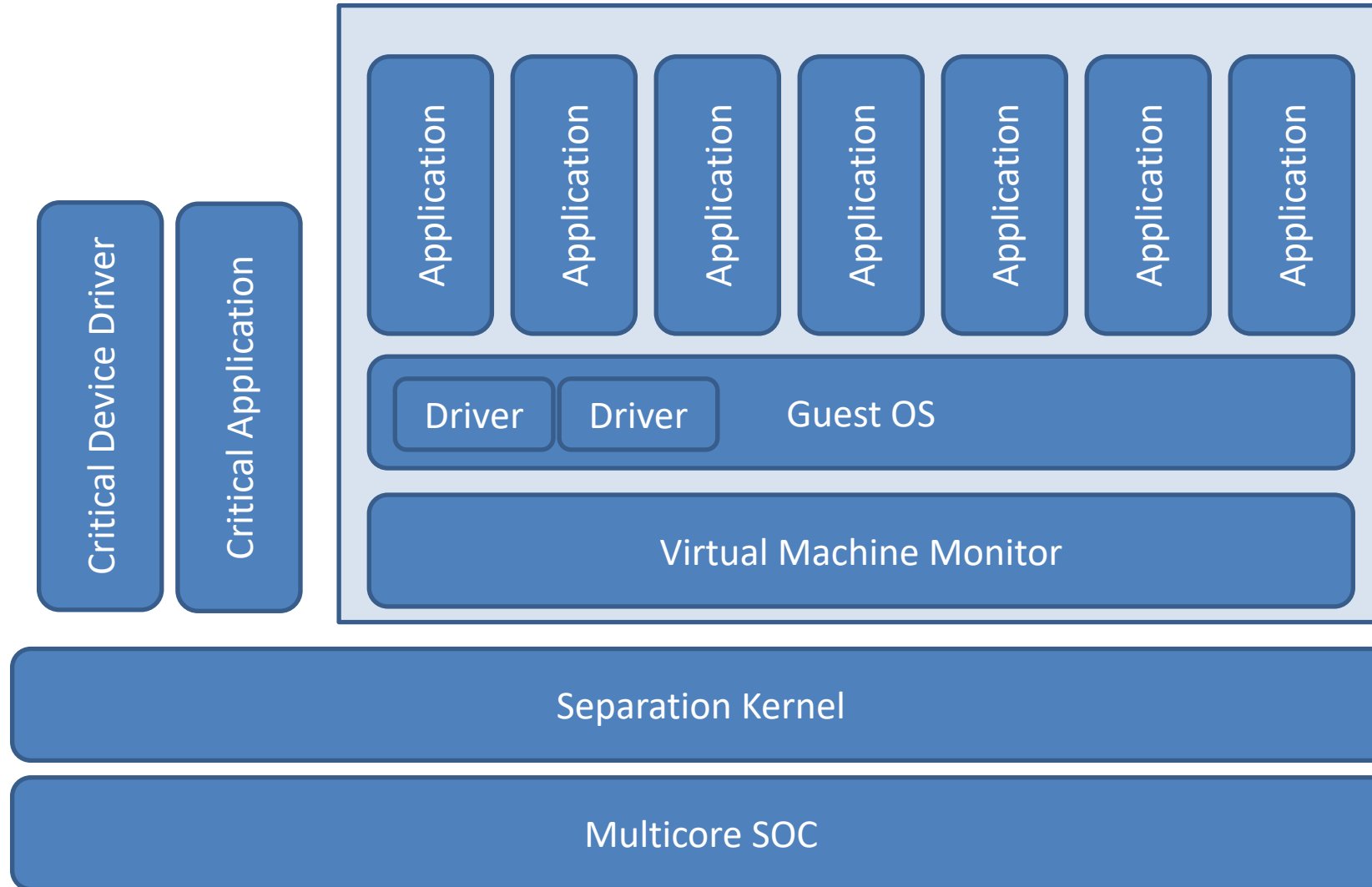
# Building a Chain of Trust



# 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

# Separation Architecture + Virtualization



# Planning for Safety and Security in Software

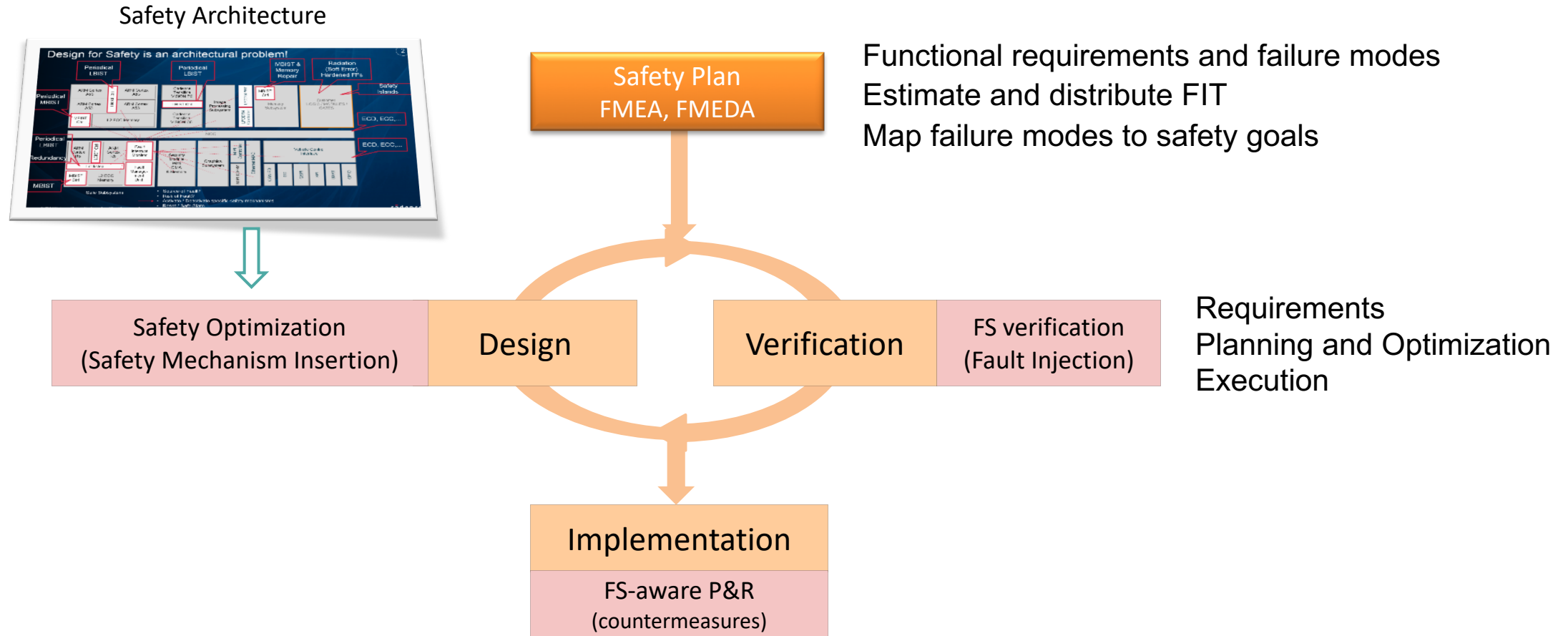
- Partitioning system resources
  - Separation allows greater flexibility
- Certifying mixed criticality components
  - SEooC can be leveraged to reduce total cost
  - Safety decomposition
  - Separation is essential
- Security policy enforcement
  - Separation allows you to be cost efficient
  - Can 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:

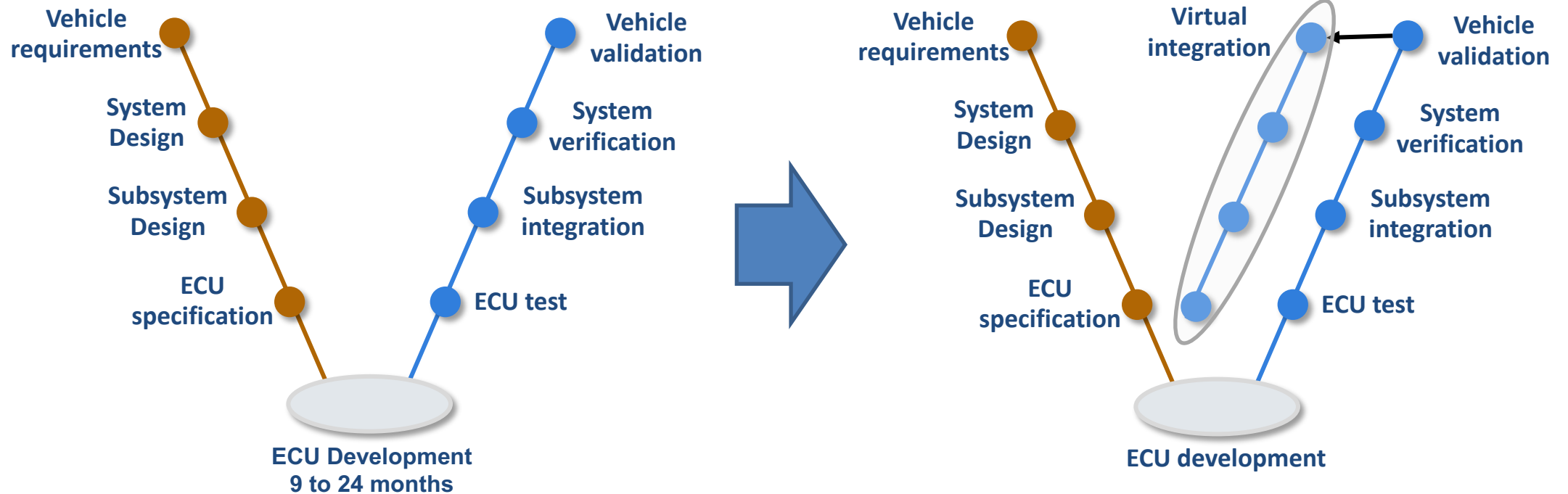
- To include safety mechanisms and meet the HW metrics/ASIL
- Safety metrics, PPA, verification time, automation are all to be considered

# 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!

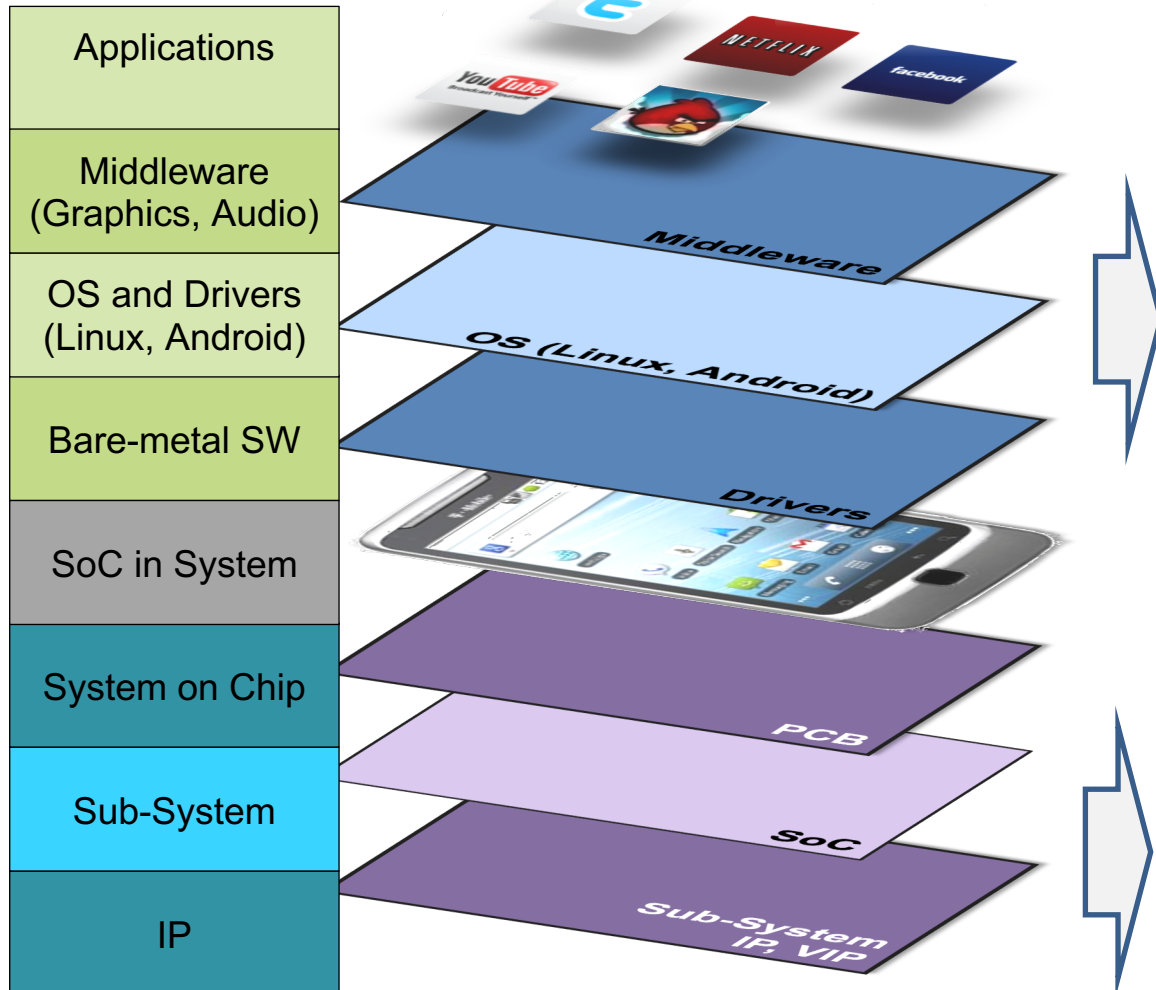
# SHIFT LEFT IN VERIFICATION

# Shift Left in the V Diagram – Automotive Example



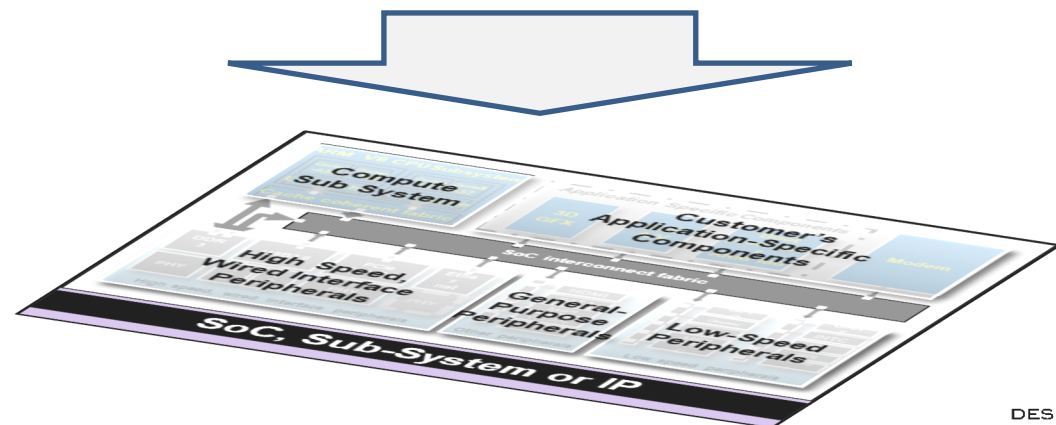
# Embedded Software Challenges ...

... addressed by Virtual and Hybrid Platforms!

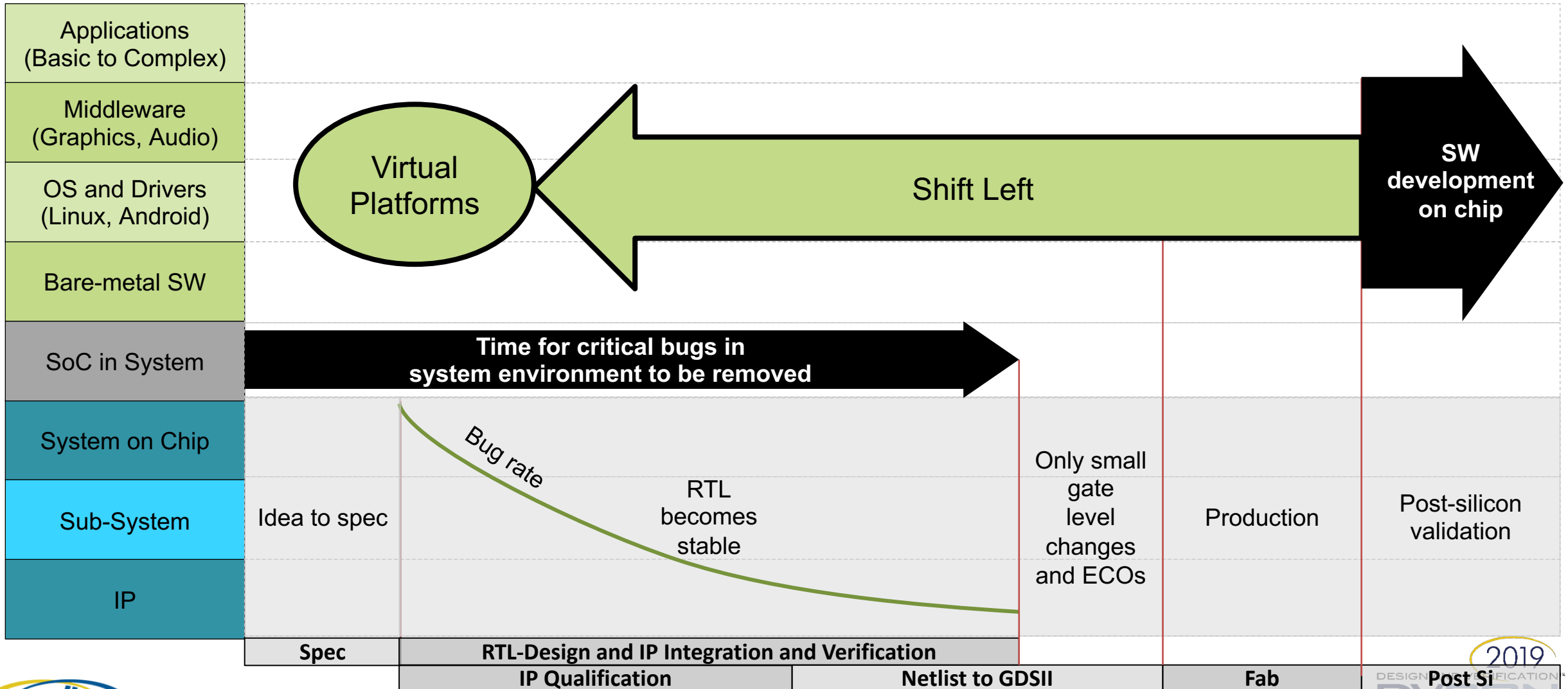


## Embedded Software Challenges


- 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



# What Do Users Care About?

 <p><b>Time of Availability</b> Earlier is better</p>	 <p><b>Speed</b> Faster is better</p>	 <p><b>Accuracy</b> More is better</p>	 <p><b>Capacity</b> More is better</p>	 <p><b>Development Cost</b> Less is better</p>	 <p><b>Replication Cost</b> Less is better</p>
 <p><b>Software Debug</b></p>	 <p><b>Hardware Debug</b></p>	 <p><b>Execution Control</b></p>	 <p><b>System Connections</b></p>	 <p><b>Bring-Up Time</b> <i>Less is better</i></p>	 <p><b>Value Links</b> (Power, Performance)</p>

# 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 ...



Source: Wikimedia Commons: <http://bit.ly/2fFtsK1>



# System Verification Characteristics



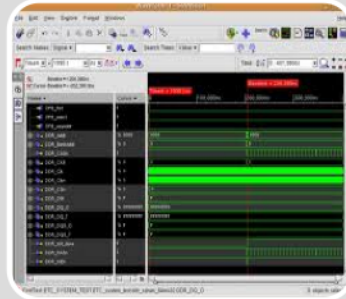
## SDK OS Sim

- 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**



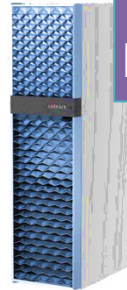
Arm Server

NEW  
2018

X86



Xcelium™  
Simulation



Custom  
Processor

Palladium®  
Z1  
Emulation



FPGA

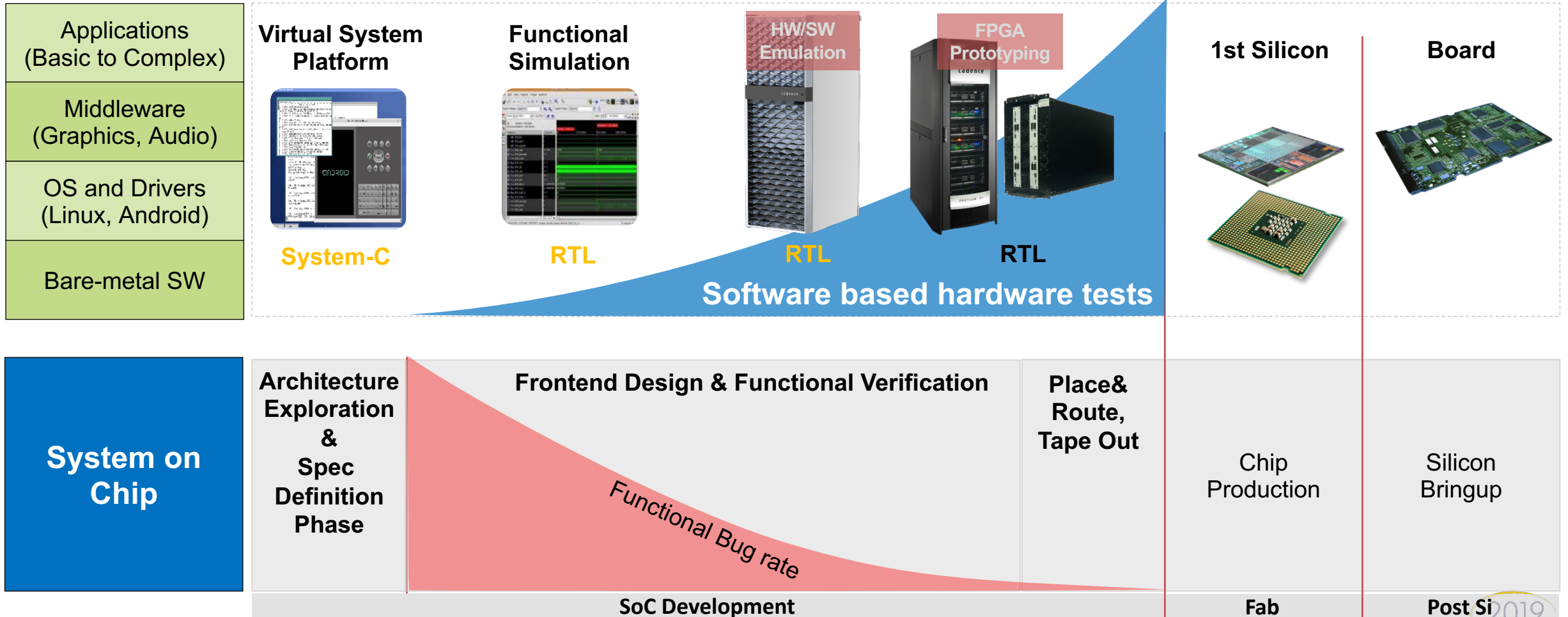


Protium™ X1  
&S1  
Prototyping



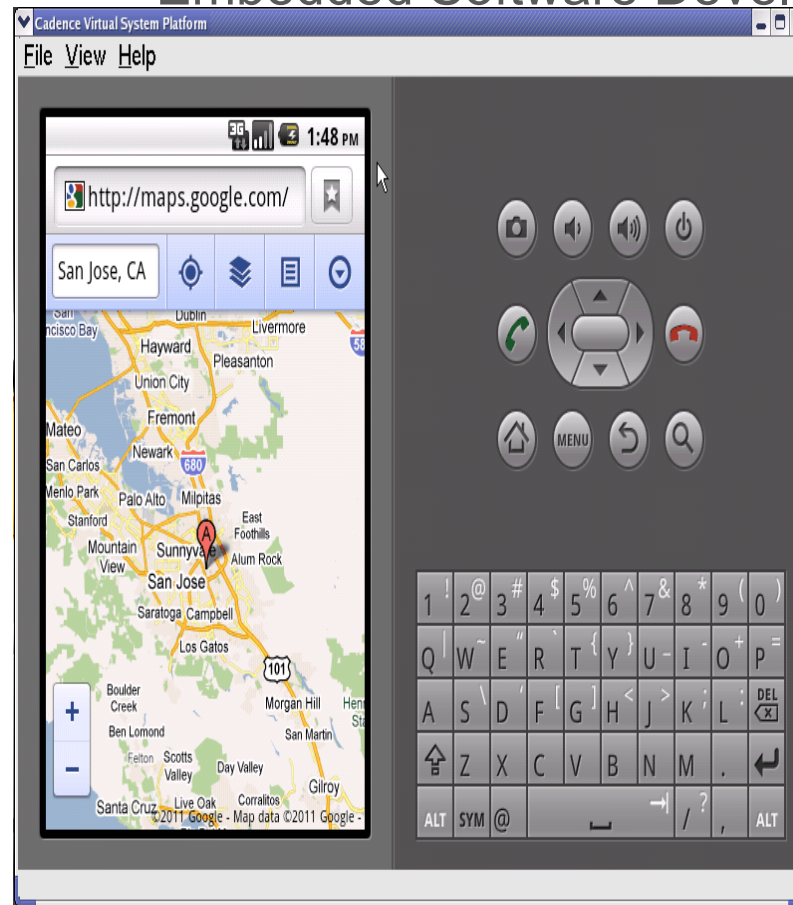
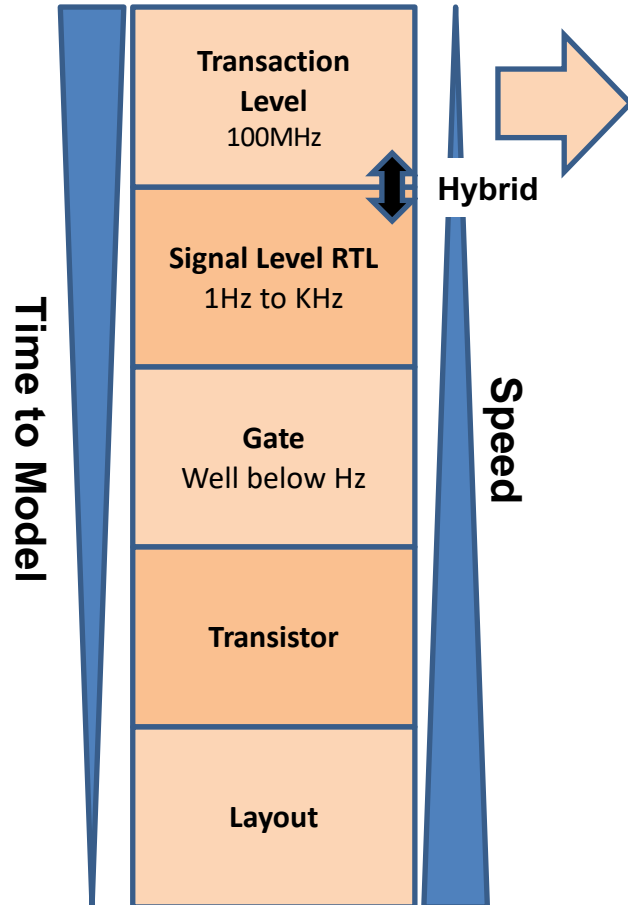
**Debug Flexibility & Compile Time**

# Hardware/Software Co-Verification during SoC Design



# 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

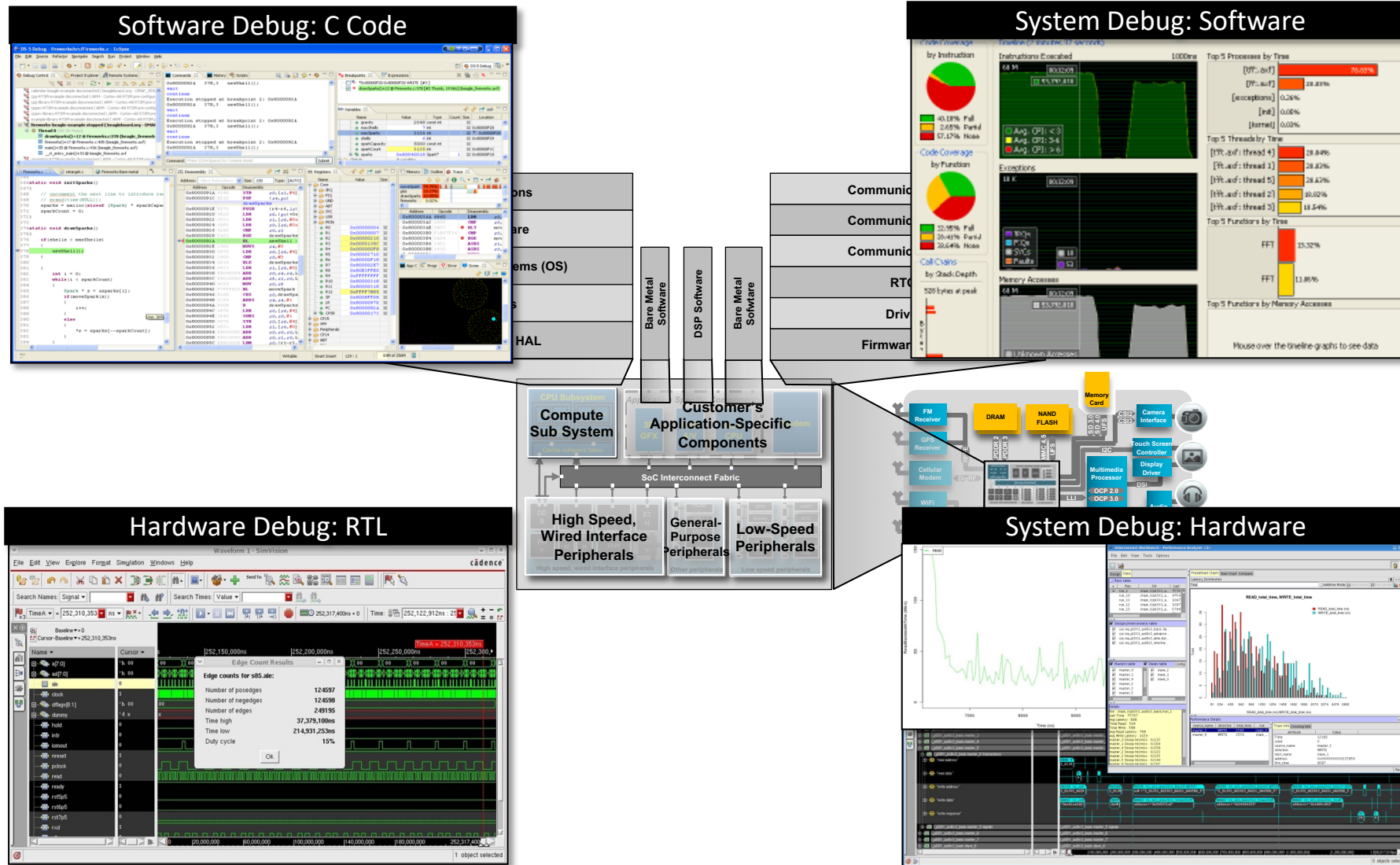


Image sources: Cadence, Arm, Green Hills

# Virtual Platform Debug SystemC/TLM Aware Debug

Activation of SystemC processes

SystemC Threads and Methods in sidebar

Source Code View

coincount_state	Suspended	Suspended	Scheduled	Running	Suspended
center_state	Suspended	Suspended	Scheduled	Running	Suspended
dm_fsm_state	Suspended	Suspended	Scheduled	Running	Suspended
resetThread_state	Suspended	Suspended	Scheduled	Running	Suspended

SystemC Module Hierarchy

The screenshot shows the main IDE interface with the source code editor in the center, a sidebar on the left for SystemC Processes, and a simulation console at the bottom. The source code is for a simple\_dma\_process.cpp file.

Debug and Simulation Console

Call Stack

SystemC/C++/C Variable Watch Window

Device Registers with Bit Fields

This screenshot shows a detailed view of the SystemC module hierarchy on the left, listing modules like simulator, scMain, Top, Consumer1-4, Monitor, and Producer. The main window displays the source code for a producer-consumer example, including a class producer and a void send\_data() function.

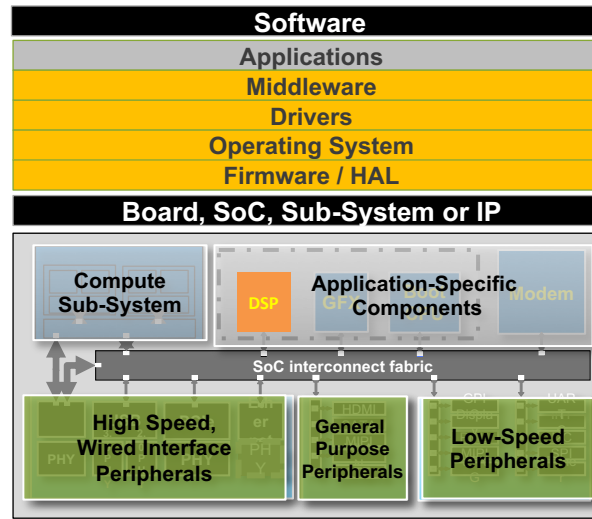


# Green Hills – Cadence Integration Points for Verification



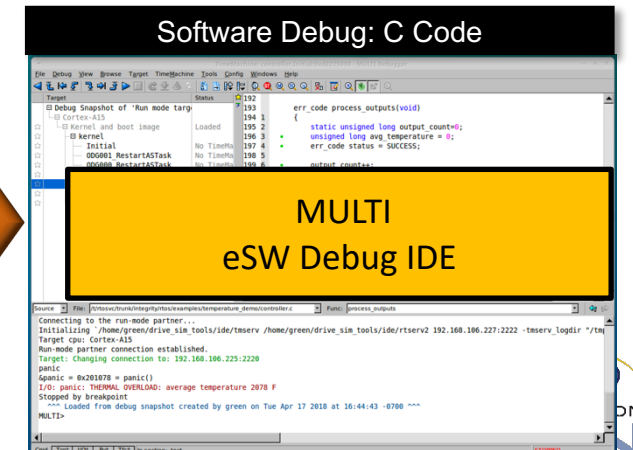
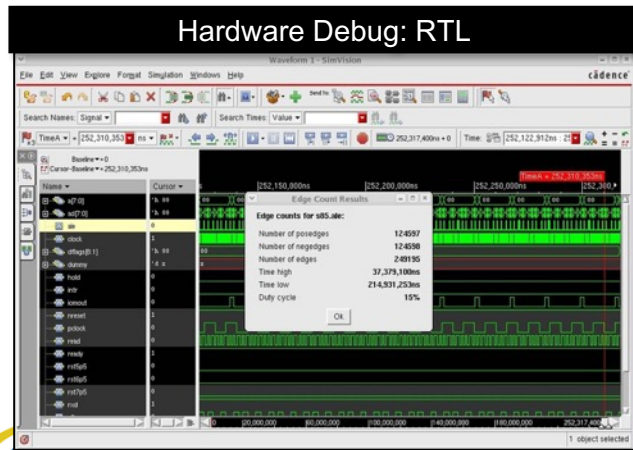
# INTEGRITY

## INTEGRITY SECURITY SERVICES™



Cadence  
Green Hills

# MULTI®





# Example Integra

## MULTI® Advanced Linux® Debugging Xcelium™ Parallel Logic Simulation



### Green Hills Software's MULTI IDE

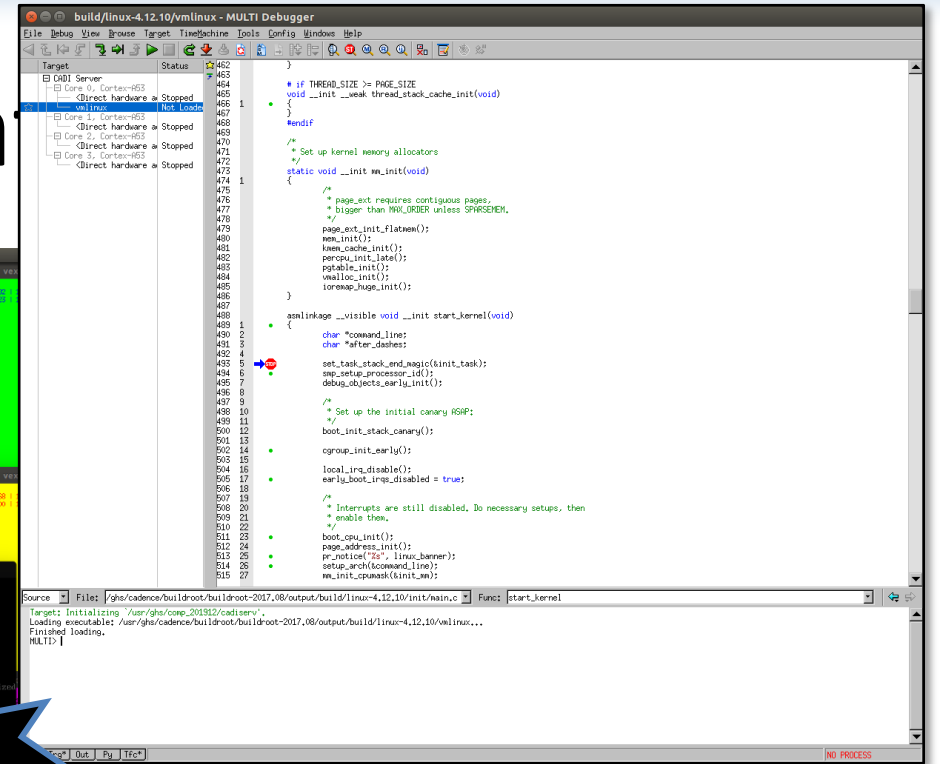
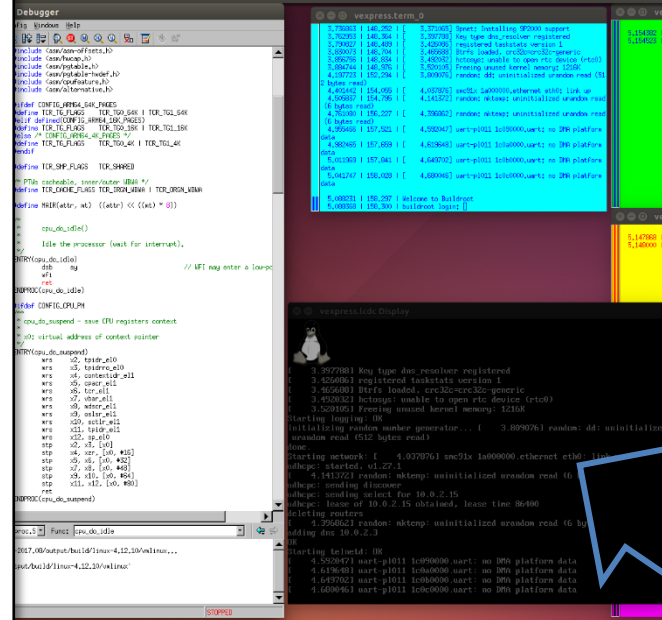
- C/C++ Debugger
- Fix bugs faster
- Find bugs automatically
- Make sense of complex systems
- Prevent new problems
- Spend more time developing

### MULTI & Virtual Platform Demo

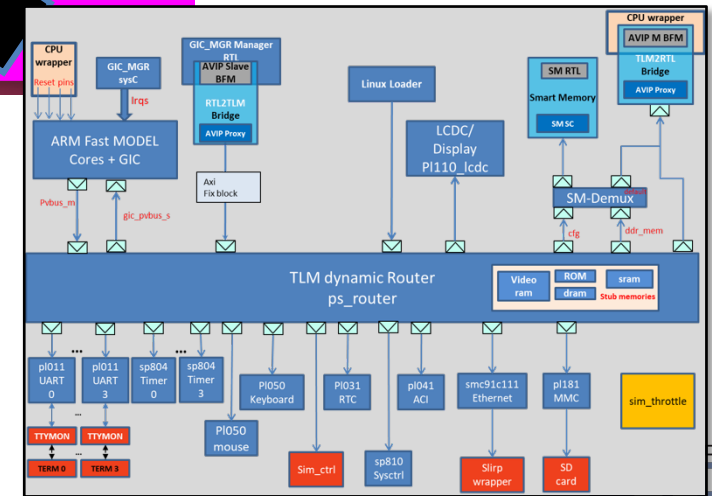
- MULTI Debugger for Linux
- Xcelium-Based Arm® SoC Virtual Platform
- MULTI connection to Virtual System Platform

### 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-

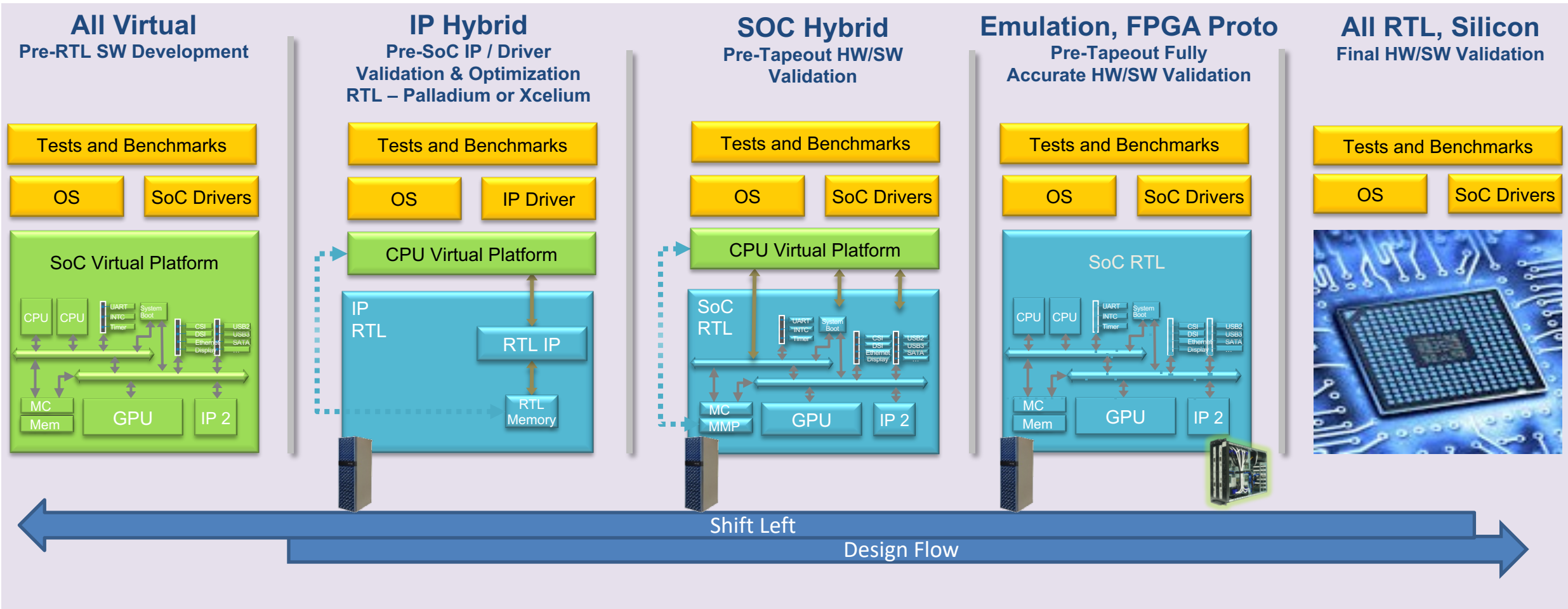
Color Code

SW stack

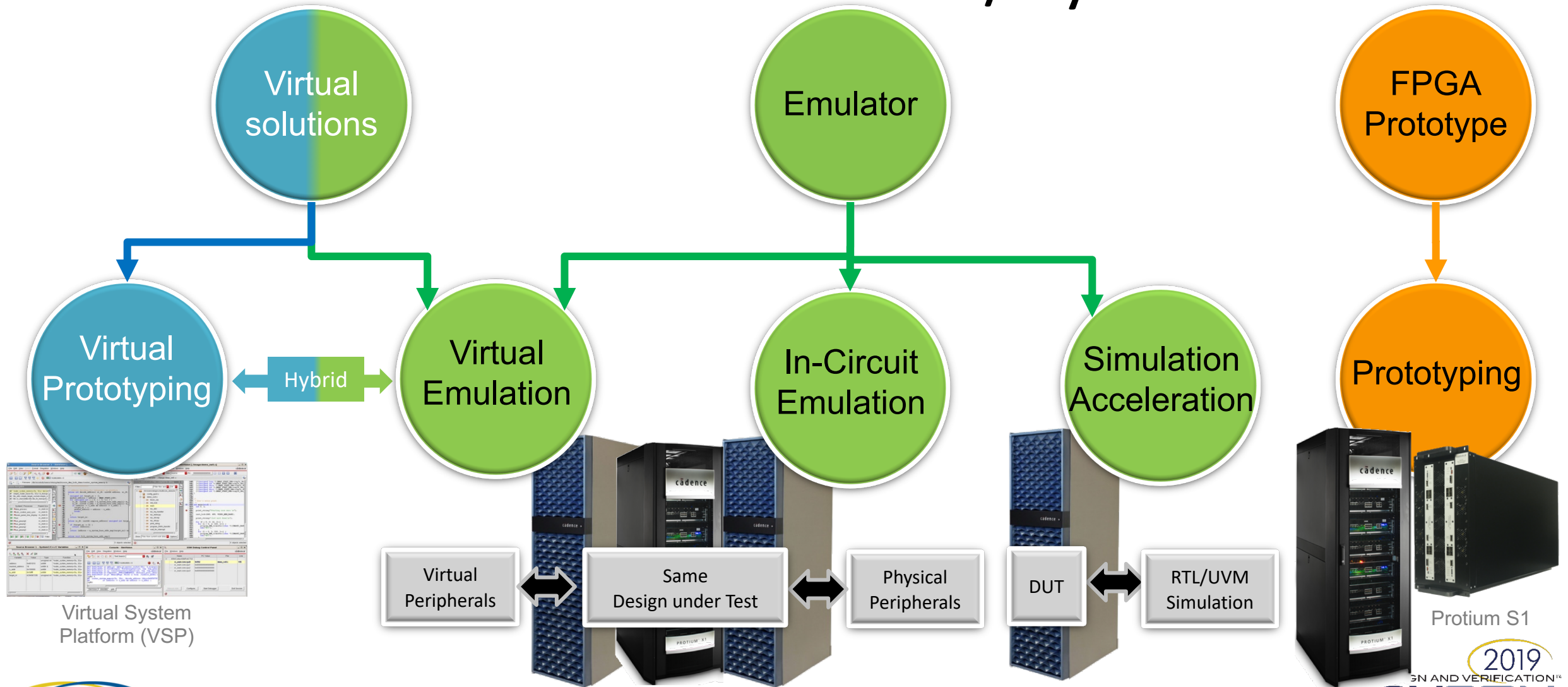
Virtual Models - VSP

RTL Models -

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



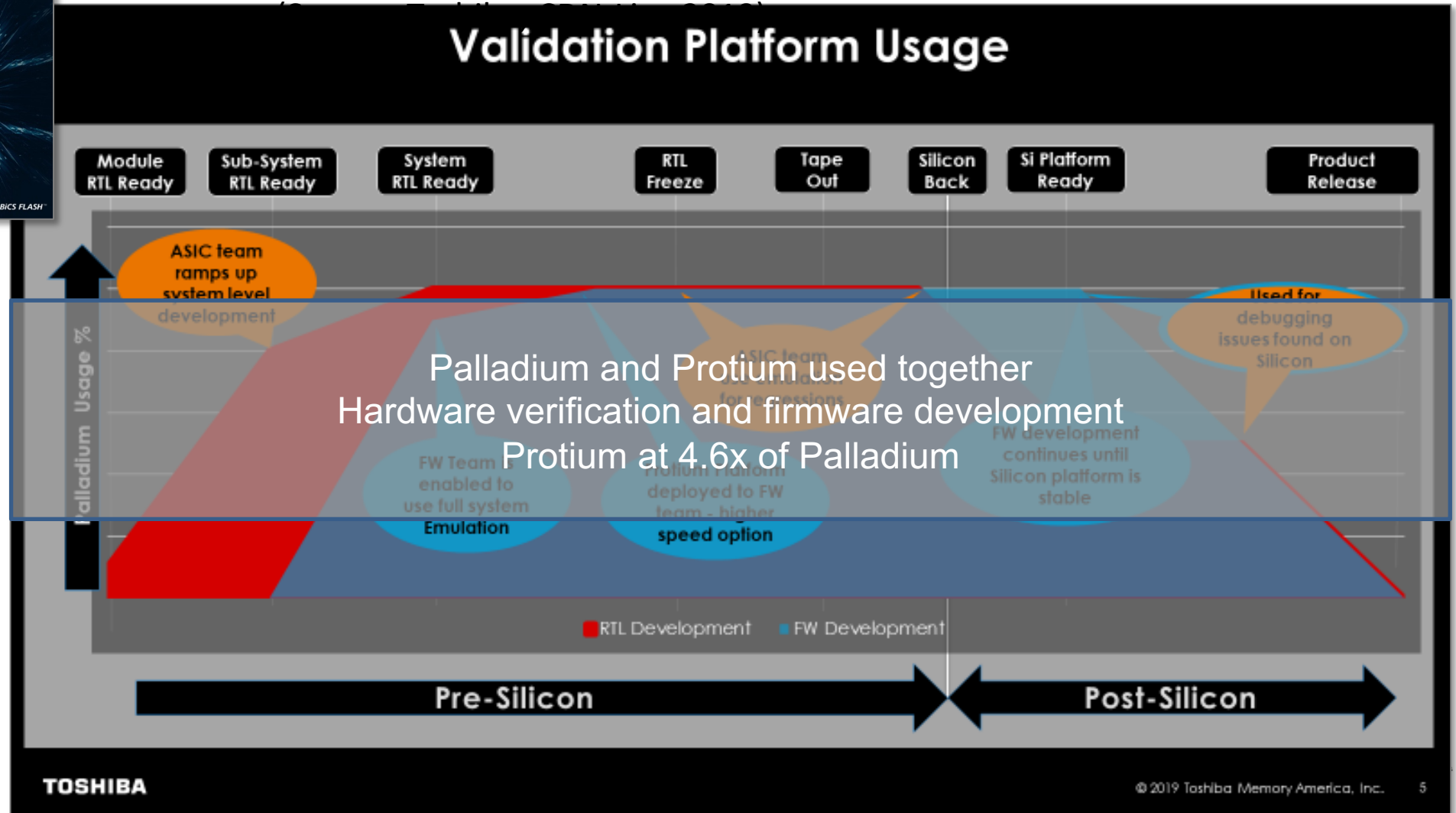
# Hardware and Virtual/Hybrids



Virtual System Platform (VSP)



# Key Trend: Unified Flow for Emulation and Prototyping



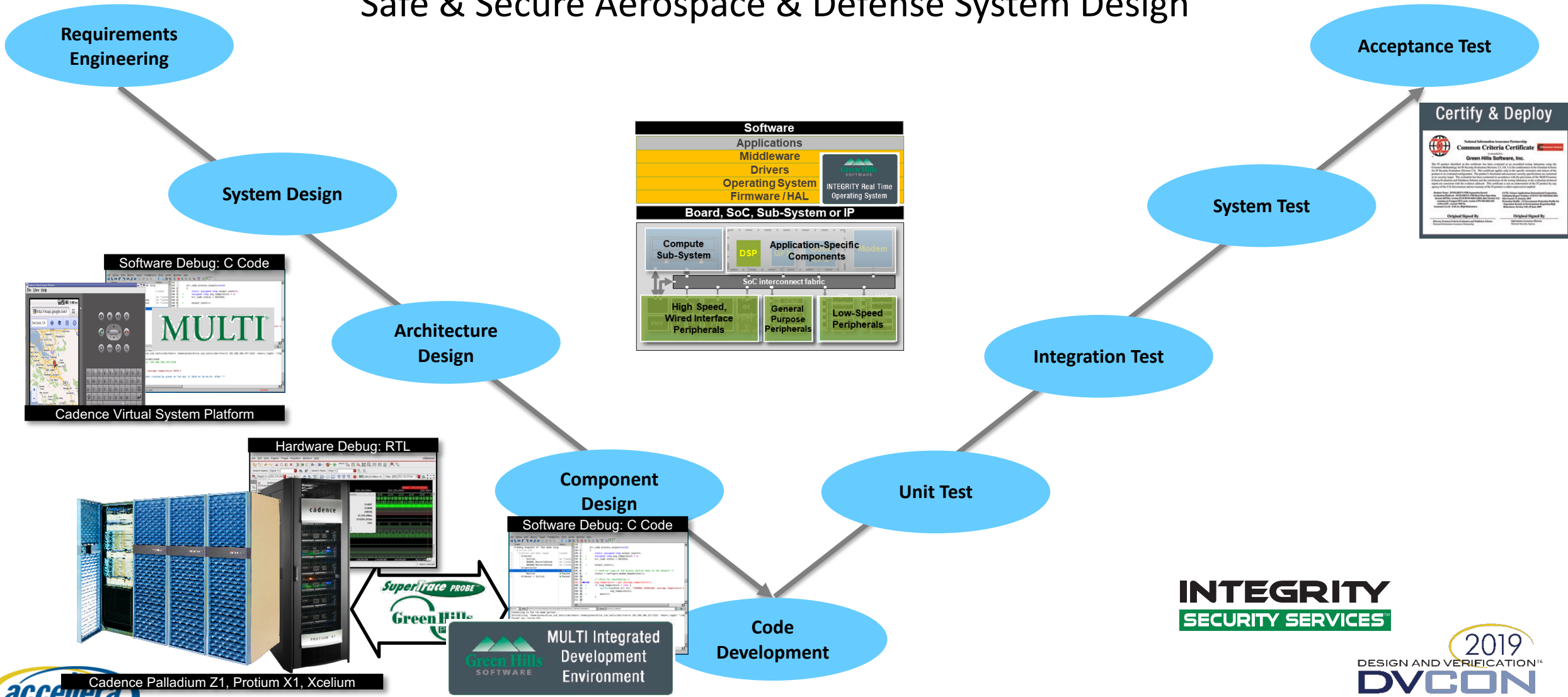
Palladium and Protium used together  
 Hardware verification and firmware development  
 Protium at 4.6x of Palladium

<http://bit.ly/2VXCjcC>



# Cadence & Green Hills Software

Safe & Secure Aerospace & Defense System Design

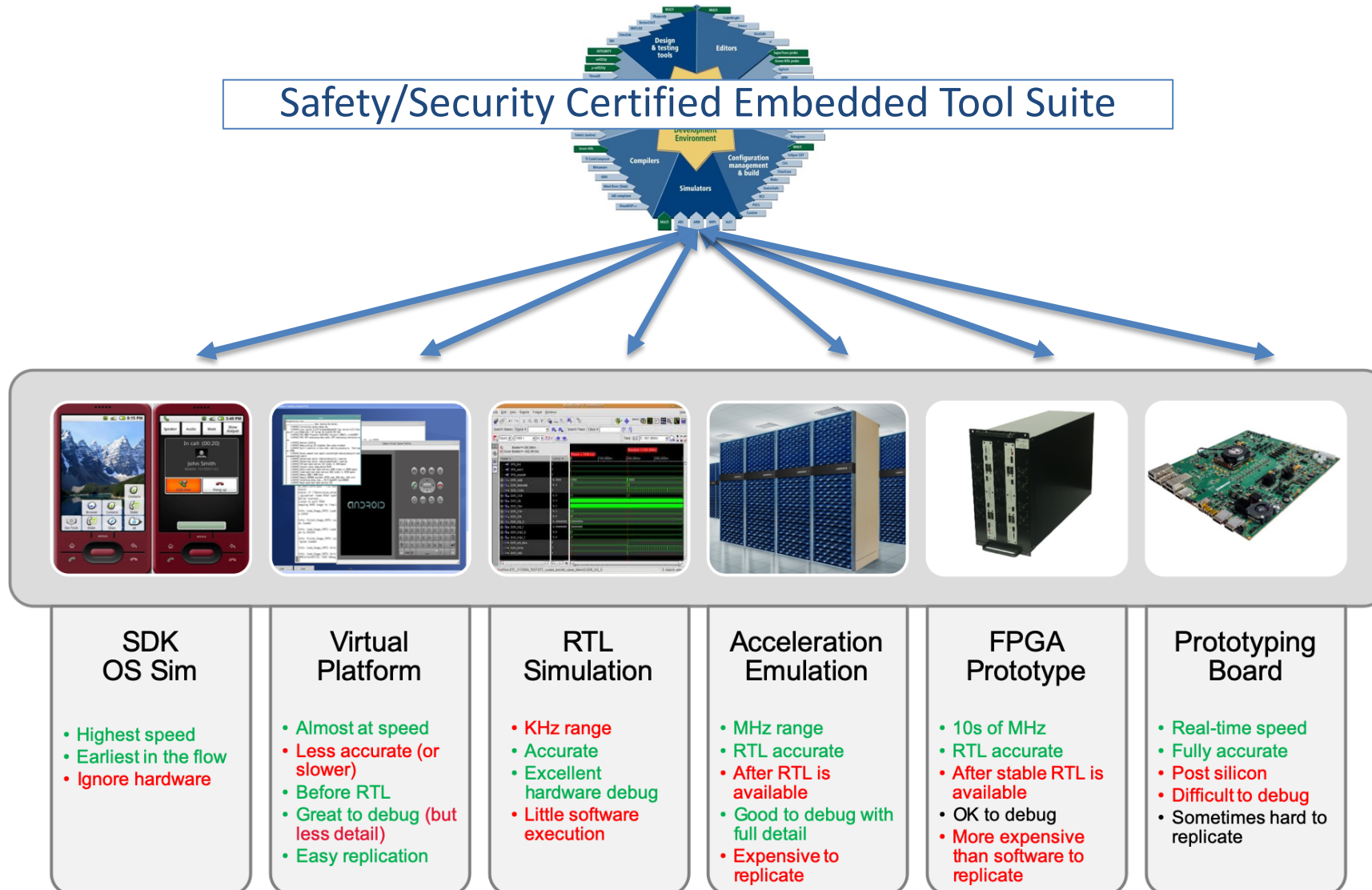


**INTEGRITY**  
SECURITY SERVICES™

2019  
DESIGN AND VERIFICATION™  
**DVCON**  
CONFERENCE AND EXHIBITION  
**EUROPE**

# Safety/Security Certified RTOS

## Safety/Security Certified Embedded Tool Suite



# BENEFITS

# More Robust Hardware and Software!

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

**Test 1000s of SW  
Scenarios  
Before Tape Out!**

Emulation Chamber



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

**SoC User**  
(Company B)



**FPGA Prototyping**  
(Dozens of users)



**SoC Developer**  
(Company A)

**Architecture Exploration & Spec Definition Phase**

Functional Bug rate

SoC Development

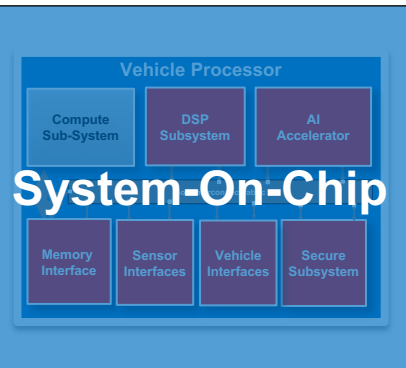
Chip Production

Fab

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

**Test 1000s of SW  
Scenarios  
Before Tape Out!**

Emulation Chamber



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

**SoC User**  
(Company B)



**FPGA Prototyping**  
(Dozens of users)



**SoC Developer**

Application Software Development  
Integration & Performance Tuning  
Chip Production

Board Bringup  
Test & Certification  
Show Board Bringup  
SOP

**Architecture Exploration & Spec Definition Phase**

RTOS Enablement & Board Support Package Development



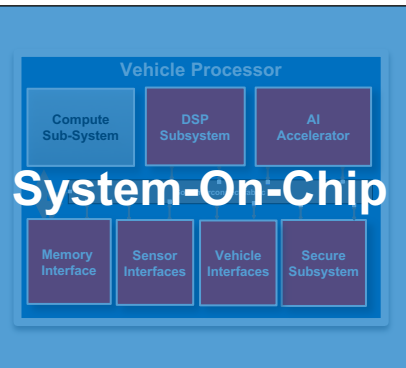
SoC Development

Software Development

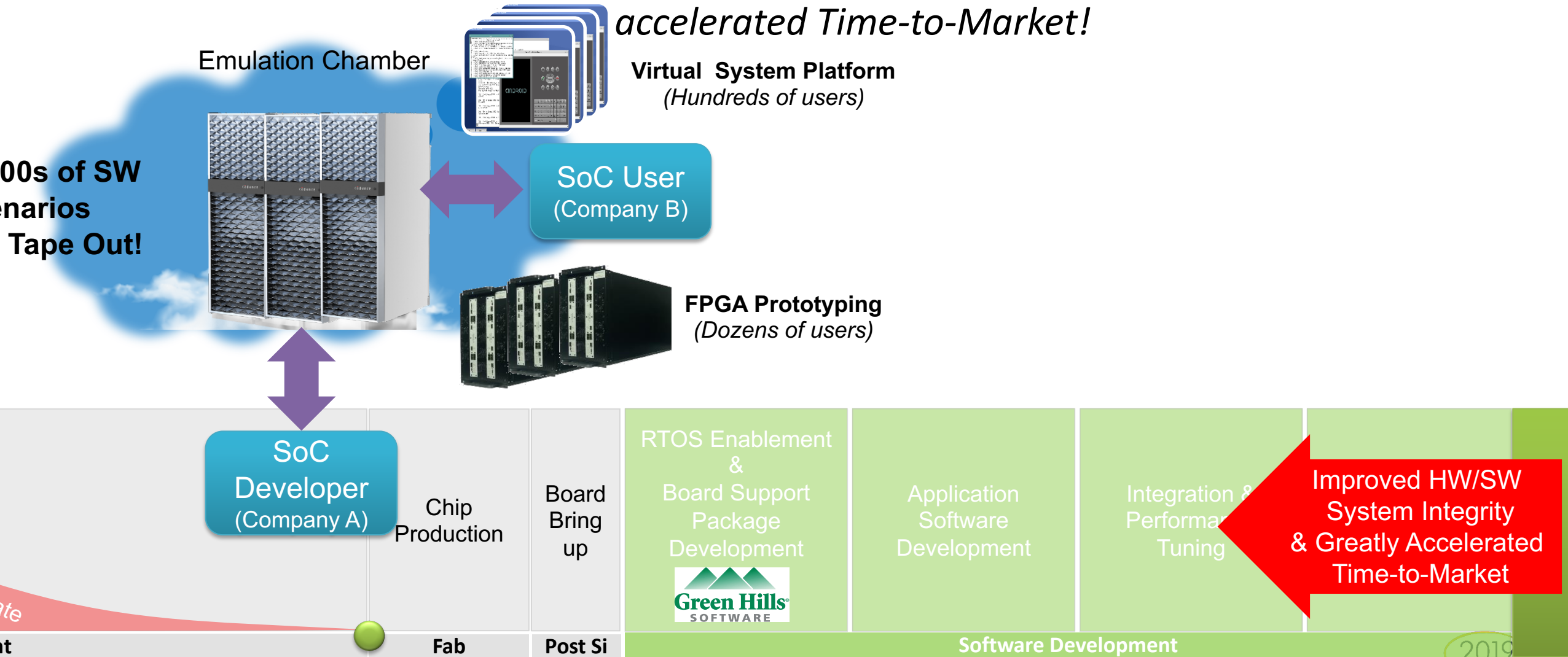
Fab

Post Si

2019



# More Robust Hardware and Software!



# Questions

Finalize slide set with questions slide