

# Accelerating GTM Software Development and Test

In the Software and Hardware Supply Chain,  
by using Virtual Hardware ECUs.

Bart Vanthournout

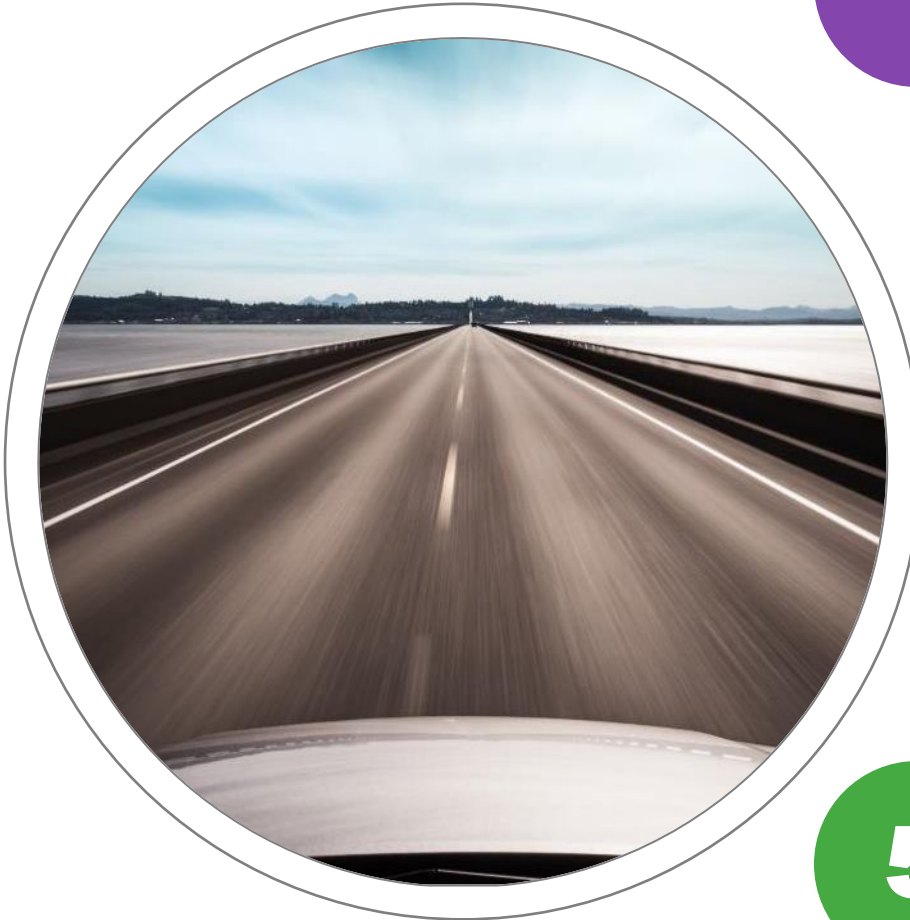
23/09/2022



# Agenda

- Industry Challenges
- Virtual Prototyping
- Supply Chain Enablement with GTM
- GTM reference Model Integration and Features

# Five Biggest Automotive Industry Challenges



1

## Supply Chain Disruption

- Worldwide chip shortage compressing the supply chain
- OEMs, Tier 1s, New Entrants are designing own SoCs

2

## Overhaul of E/E Architectures

- Demand for new HAD/ADAS features, electrified powertrain, infotainment
- New zonal architectures

3

## Complex Software Development & Deployment

- Millions lines of code
- Software updates over the air
- SW Ecosystems

4

## Increasing Security Challenges

- New vulnerabilities from OTA software updates, self-driving cars, 5G connectivity
- Additional focus on security + safety

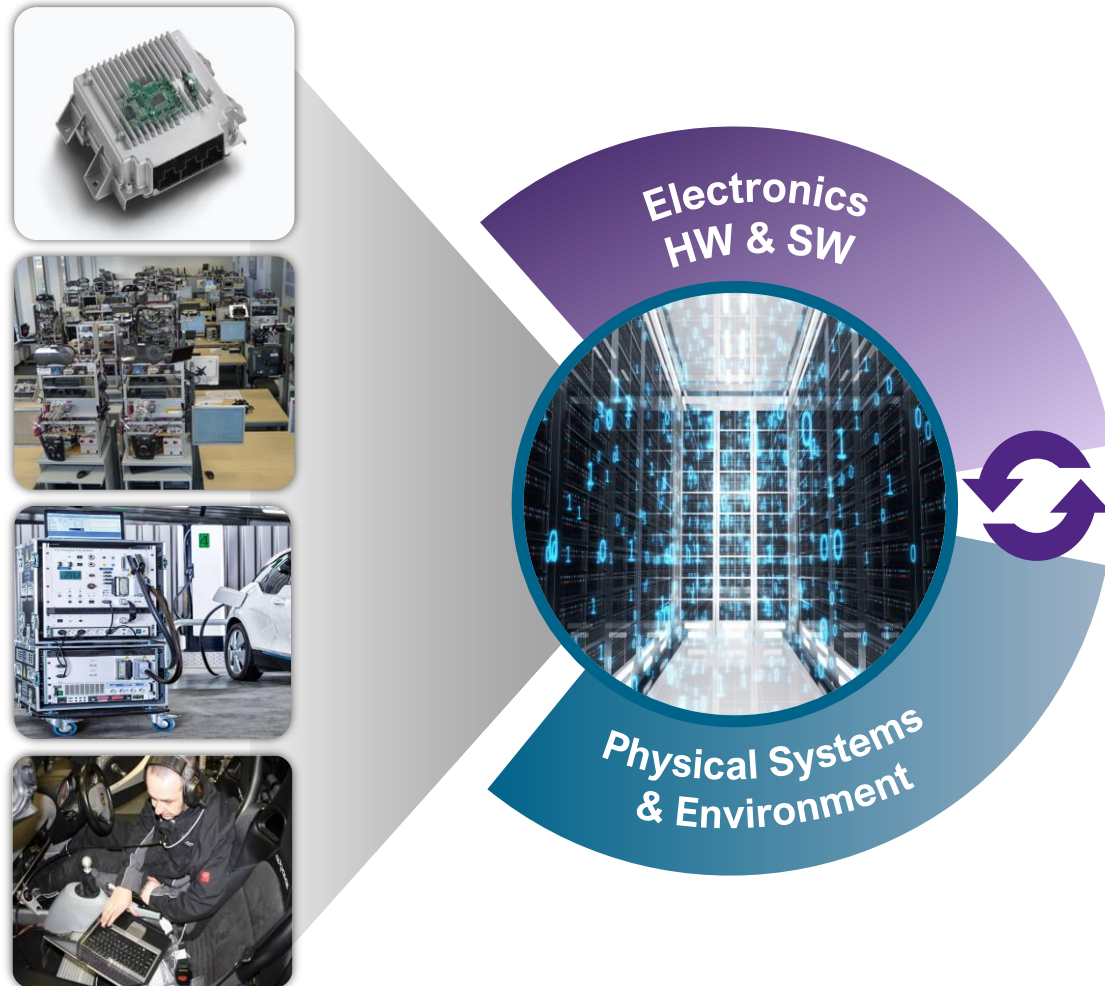
5

## Accelerating Time-to-Market to Compete

- New automotive entrants challenging traditional processes

# Conquer the Challenges of HW/SW System Validation

Virtual Prototyping Enables a Software-First Mindset



- ✓ Supply chain collaboration with executable specification
- ✓ Early SW bring-up
- ✓ More productive system testing with
  - Software-in-the-Loop
  - Virtual Hardware-in-the-Loop
  - Virtual vehicle
- ✓ Supports agile development, continuous integration and delivery
- ✓ Functional safety and security system validation

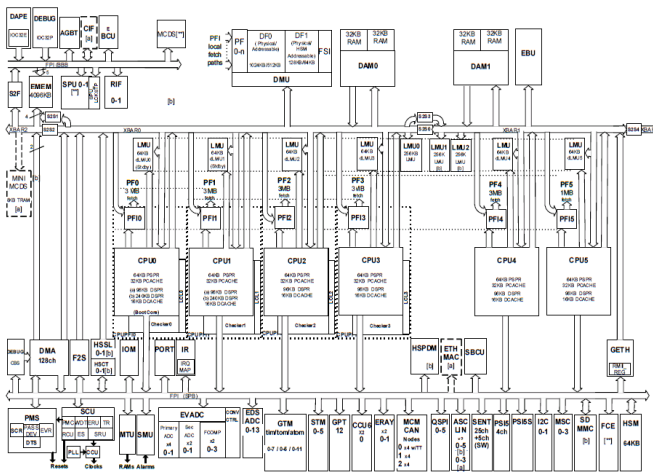
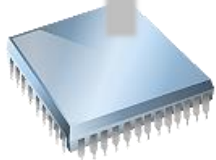
# What is a Virtualizer Development Kit (VDK)?



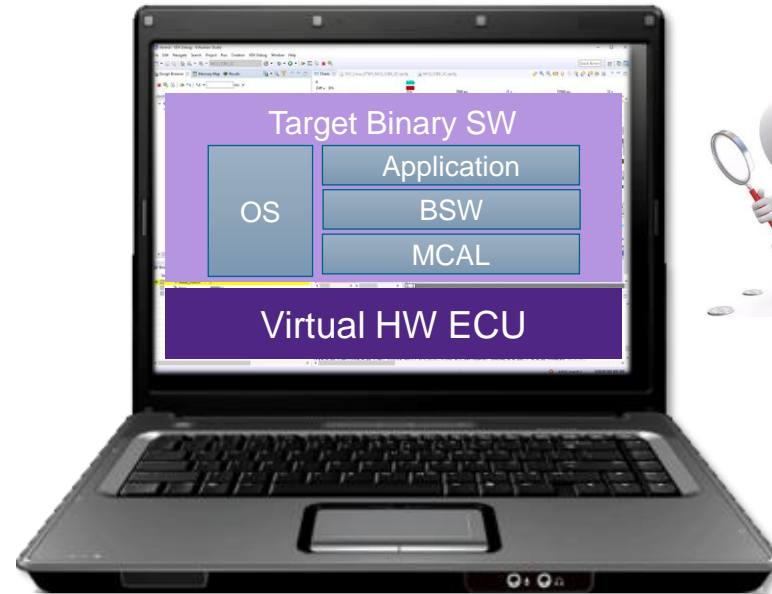
ECU



MCU

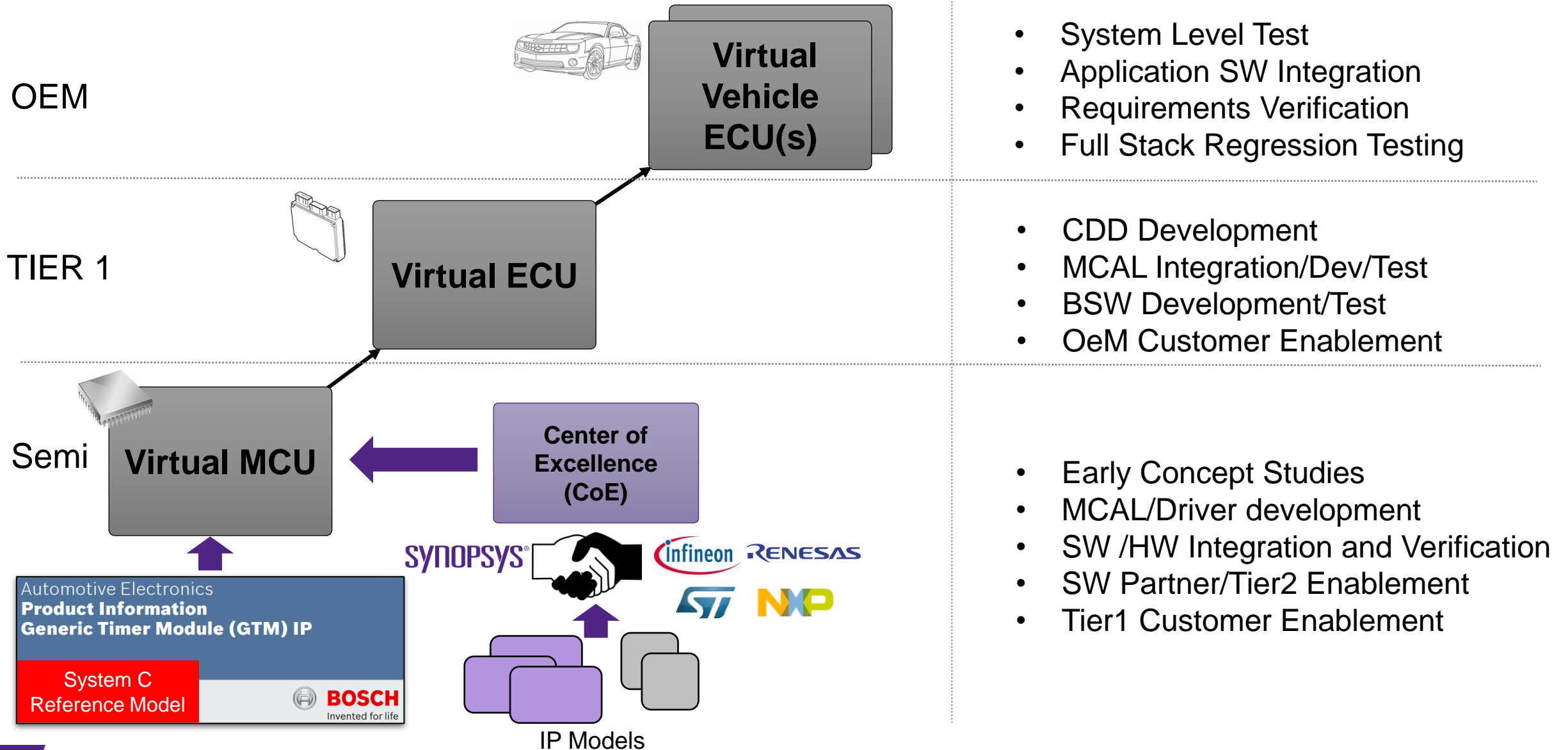


*Model Creation* →



- Fast software model of the digital hardware ECU or MCU
  - Abstracted for Software Execution
- For the purpose of executing unmodified binary production software
  - Without dependency on real hardware
- And providing a higher debugging/analysis efficiency
  - And solving hardware accessibility issues, pre and post silicon

# Supply Chain Enablement with GTM use cases



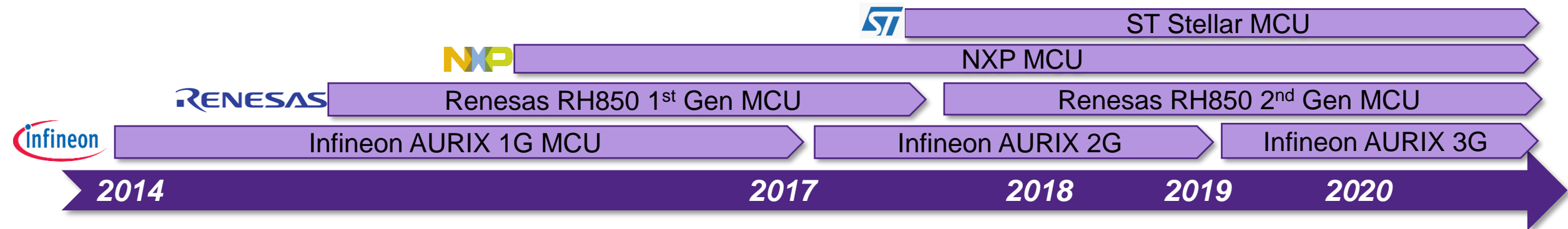
- System Level Test
- Application SW Integration
- Requirements Verification
- Full Stack Regression Testing

- CDD Development
- MCAL Integration/Dev/Test
- BSW Development/Test
- OeM Customer Enablement

- Early Concept Studies
- MCAL/Driver development
- SW /HW Integration and Verification
- SW Partner/Tier2 Enablement
- Tier1 Customer Enablement

# Enabling VDKs with the GTM & Reference Model.

- Multiple VDKs Support GTM Enablement with Multiple Semis
  - SNPS partners with top Semis in Automotive
- Synopsys Integrates Bosch GTM Reference Model Sources.
  - Including enhancements for debugging/analysis and performance
- Synopsys has a long-term relationship with Bosch
  - Delivers feedback from heavy Tier1/OEM usage
  - Delivers feedback and performance enhancements





# Semi Customer and Partner Enablement Example



Internal and Partner AURIX™ TC4xx VDK Use Cases

Dineshkumar Selvaraj

**SAE** INTERNATIONAL <https://event.webcasts.com/starthere.jsp?ei=1396695>

Automotive Electronics  
**Product Information**  
**Generic Timer Module (GTM) IP**

SystemC  
 Reference Model

**BOSCH**  
 Invented for life

OEM

- ASW Development
- Integration Testing
- System Level Test Cases

Tier 1 Customer

- BSW Development
- MCAL/CDD Development

IFX Partners

- Compiler
- Debugger
- SW Tooling Providers

Internal Teams



- Concept Engineering
- Driver Development (MCAL,...)
- Software Team Enablement

SYNOPTYS®



# GTM Reference Model Integration Features

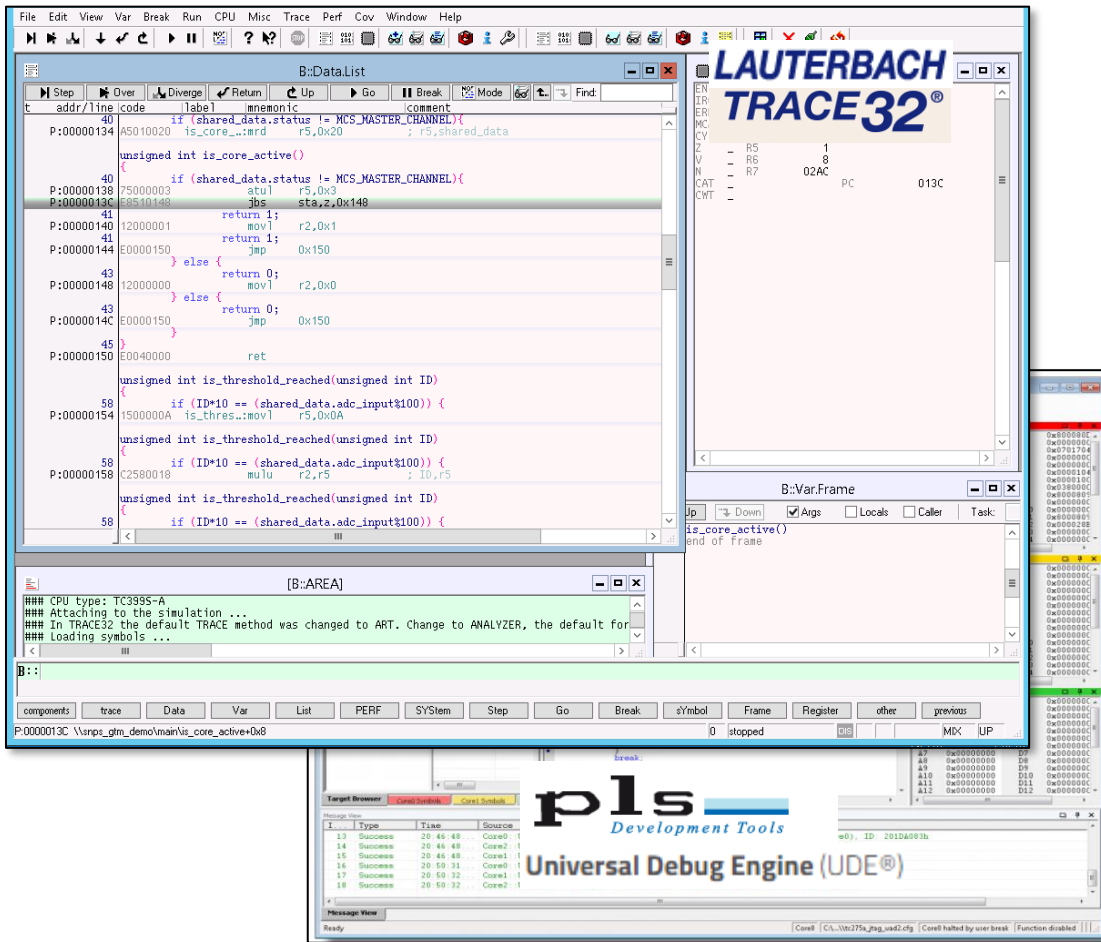
Enabling development, integration and test in the Supply Chain



# Virtualizer GTM Reference Model Integration



# Third Party Debugger Tooling - Multicore Capable



- Support for Full Register Set (PER file)
- MCS memory visibility and interactivity.
- Support for synchronized multicore debugging for MCS channels
  - Synchronised to MCU cores to maximize visibility and correlation.
  - Co-debugging with MCU cores supported
- MCS-asm debugging support
- High Level Language (C/C++) debugging support for MCS engines.

# Virtualizer GTM Synchronized Debugging and Visibility

The screenshot displays the Synopsys SystemC debugger interface with several key components:

- Design Browser:** Shows a hierarchical tree of components including GTM, gtm\_wrapper, gtm\_top, and multiple mcs0 channels (mcs0\_ch0 to mcs0\_ch7).
- Register View:** A table showing register names (R0-R7, CTRL, ACB, PC, MHB, CTRG, STRG, ERR, CTRL\_STAT, RESET, CAT, CWT) and their values. A callout box labeled "Multi-MCS Core Register View" points to this table.
- Disassembly View:** Shows assembly code for a core (mcs0\_ch0) at address 0x00000118. A callout box labeled "Disassembly View" points to the instruction list.
- Memory Views:** A table showing memory addresses and their corresponding hex values. A callout box labeled "Memory Views" points to this table.
- GTM Config Register Interactivity:** A table showing configuration registers (ATOM5\_AGC\_ACT\_TB, ATOM5\_AGC\_OUTEN\_CTL, etc.) and their values. A callout box labeled "GTM Config Register Interactivity" points to this table.
- Virtualizer Simulation Scripting:** A code editor showing a Python script for defining a function entry observer. A callout box labeled "Virtualizer Simulation Scripting" points to the script.

- All MCS executable Channels instrumented as Virtualizer “Cores”.
- Seamless Channel switching with MCU cores and GTM MCS “cores”
- Interactive MCS Register Views
- Interactive Config Register Views
- Virtualizer Debugger Scripting supported for integrated system debugging.

# Virtualizer GTM Tracing and Analysis

The screenshot displays the Synopsys SystemC Processes tool interface. Key components include:

- SystemC Processes Pane:** Shows a call-graph for MCS Channel Function Tracing, with a call stack including `_START`, `main`, `master_task`, `is_control_loop_active`, `is_core_active`, `process_data`, `trigger`, `trigger_core`, and `wait`.
- IntegerTrac Pane:** Shows SW Variable Monitoring for variables like `scratch_data_ch7`, `scratch_data_ch6`, `scratch_data_ch5`, `scratch_data_ch4`, `scratch_data_ch3`, `scratch_data_ch2`, `scratch_data_ch1`, `scratch_data_ch0`, `scratch_adc_input`, and `scratch_adc_input=80`.
- Simulation Output Pane:** Shows Detailed MCS Instruction Tracing (mcs-asm) with columns for Time (ps), Function/Address, Context, Address, and Disassembly.
 

Time (ps)	Function/Address	Context	Address	Disassembly
896538180	slave_task + 0x18		0x228	MOV R2, R6
905520000	slave_task + 0x1c		0x22c	CALL 0x154
905560000	is_threshold_reached + 0x0		0x154	MOVL R5, 0xA
905600000	is_threshold_reached + 0x0		0x158	MULU R2, R5, 0x18
905628571	is_threshold_reached + 0x0		0x15c	MRD R5, 0x28
905657142	is_threshold_reached + 0x0		0x160	MOVL R3, 0x64
905685713	is_threshold_reached + 0x0		0x164	DIVU R5, R3, 0x18
905714284	is_threshold_reached + 0x14		0x168	ATU R2, R4
- Console Pane:** Shows GTM Internals Trace Debug Messages, including `<terminated>-Hitachi-STM-ERROR (VSP-Simulation) C:\usr\synopsys\virtualizer-comet-k-2015.06\amp;3\bin\sys_loader.exe` and various INFO messages.

- Virtualizer Integration for Software Analysis
  - MCS Function Call-Graph Tracing
  - MCS Instruction Tracing
  - GTM config register tracing
  - MCS Memory access tracing
- Integrated GTM Ref Model “Internals” tracing in unified console.
  - Controlled via scripting or interactively
- Viewing and Debugging in context of mainline MCU SW execution.

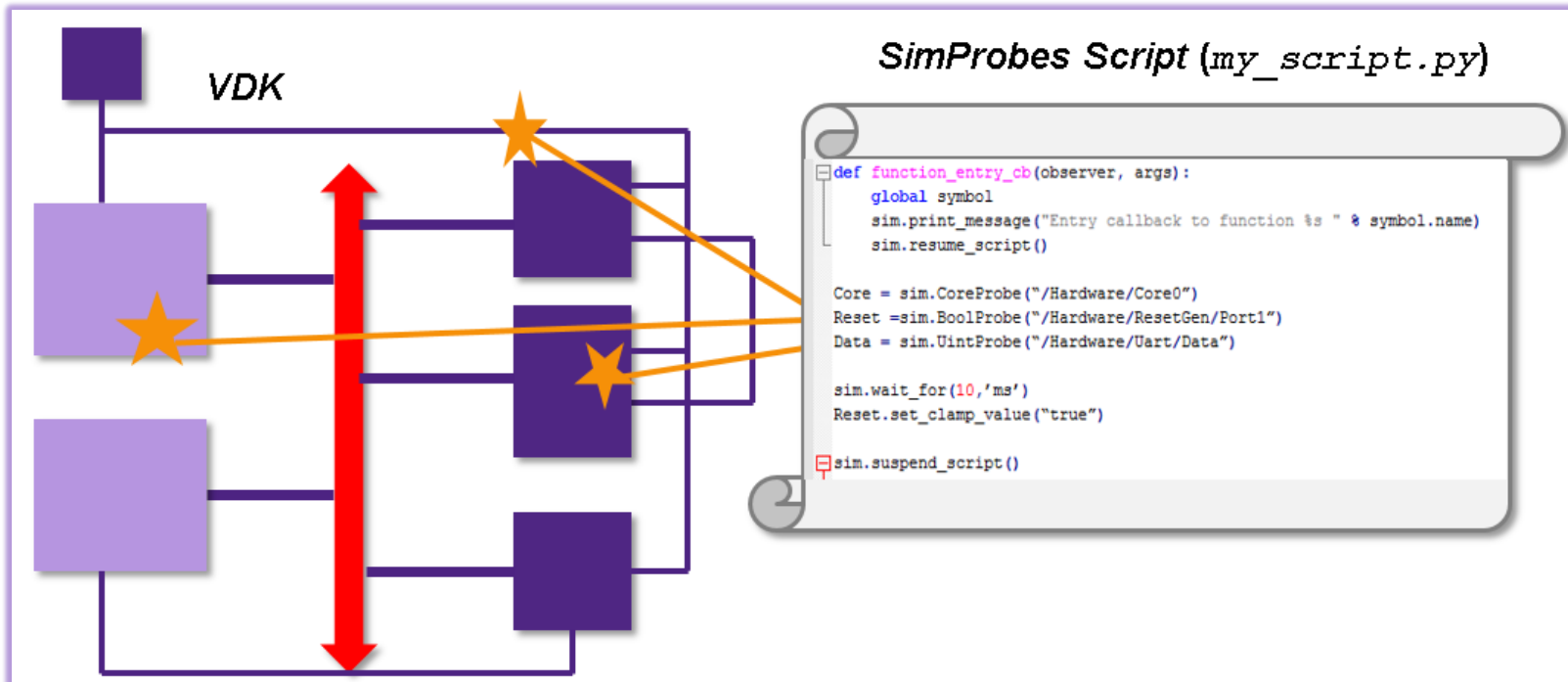




# Virtualizer Simulation Probes

(stimulus, fault injection, and analysis)

- GTM integration is compatible with Virtualizer Simulation Probes
- What is Virtualizer Simulation Probes.
  - An in-simulation python scripting framework for fault injection, signal stimulus and analysis.



- Allows you to override and control signals anywhere in the MCU VDK model
- Override and control mechanisms are uniquely supported in the Synopsys SystemC kernel.
- Unlimited number of scripts allowed to run in parallel.
- It runs inside simulation process , so has high simulation speed.

# Practical GTM use Case on Powertrain Virtual HIL (vHIL)

- Target –
  - Reliable testing methodology with regressions
  - Replacing HIL stimulus with scripted stimulus
    - Crank/Cam/Injection....
- Methodology -
  - Scriptable waveform generation for repeatable stimulus scenarios.
  - Inject faults into input stimulus or output signals to establish system impact.
  - Custom analysis for debugging purpose
  - Regression environment (CI/CD, Jenkins...)
  - VDK including an integrated GTM reference model critical part of the verification.

### Stable & Automated Regressions for Auto SW

Presented by Punch Torino at Synopsys Virtual Prototyping Day 2021

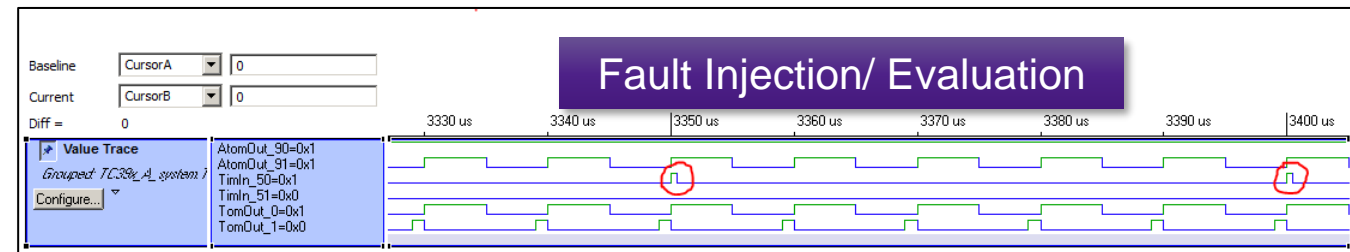
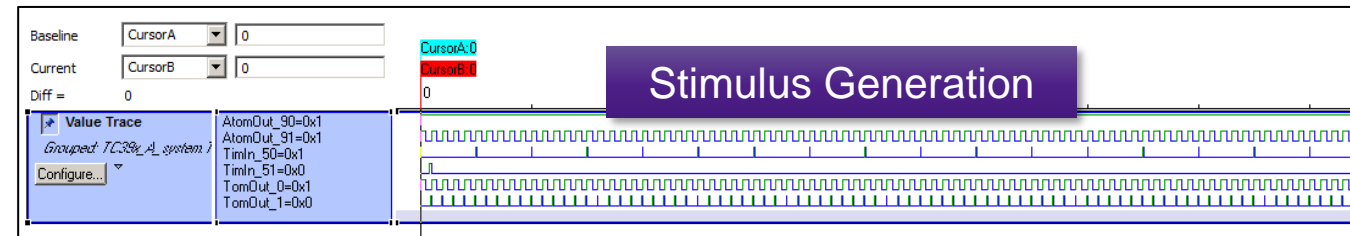
**Conclusion**

- Final SW Validation **MUST** be executed on HW Bench with HW sample
- But... to speed up the development process, the proposed solution:
  - is highly flexible, scalable and portable
  - is faster: even if VDK simulation requires more time than real micro, the overall test time duration divided by # of parallel nodes gives a good result
  - is cost effective if compared to the simplest test bench target for automotive grade level SW testing
  - is definitely more stable => High Availability (HA)
  - is automatic: developers can launch tests with 1 "click"

Improve performance still possible using orchestrator / K8s

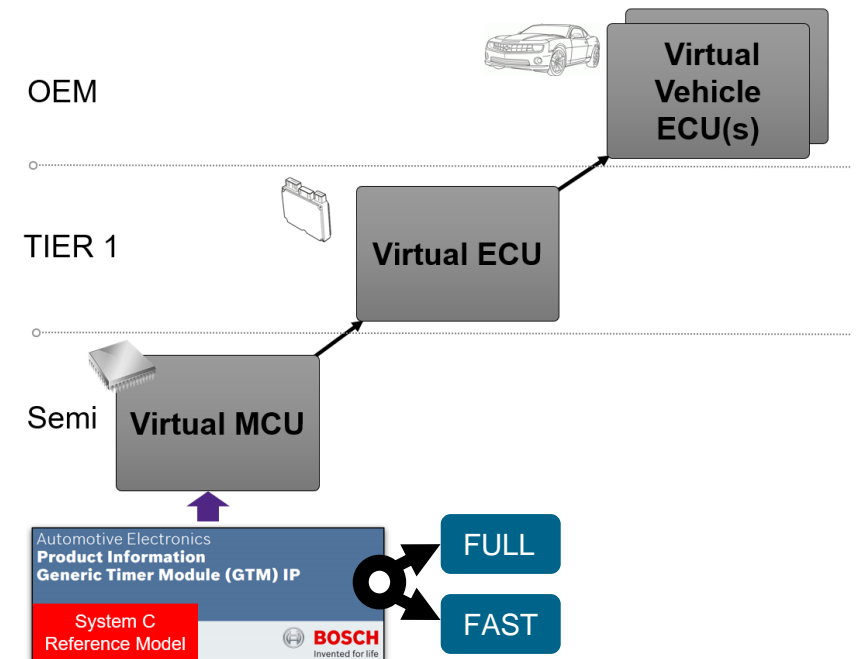
**PUNCH | Torino**

- Automotive SW challenges: growing complexity, sophisticated functionalities, connectivity, active safety
- HW setup with HIL, board, test PC complex and unreliable
- Jenkins based regression setup with VDK and Python scripting enables DevOps methodology in auto SW development process
- Modular setup repos for models, test SW, test scripts, and test framework and docker containers for regression node



# Working with GTM through the supply chain

- There is a need for more abstractions as we move up the design chain and systems become more complex
  - Mixing different representations VDK, Silver, ...
- GTM is used along the supply chain from Semi to OEM
  - We need an “Optimized Fast GTM” based on the use case
  - OEMs can often get away with some abstraction of detail
  - Tier1 and Semi may not need all details, especially if GTM is not the focus of a particular test
- Use a switchable GTM model
  - A combination of the reference model with a more abstract implementation
  - User can decide which one to use, full detail or fast model



# Summary

- Virtual Prototypes are essential to address today's automotive design challenges
- The GTM is a critical component of many MCUs and requires advanced debugging and analysis features to fully exploit its feature set
- Synopsys has a comprehensive solution around the GTM enabling SW development in the context of the full system at any point along the supply chain
- Synopsys is actively involved in integrating and optimizing the GTM reference model provided by BOSCH.

**Thank You**

