

Accelerating GTM Software Development Integration and Test using Virtual Hardware ECUs

Jason Niatas
Synopsys Inc
October 2017

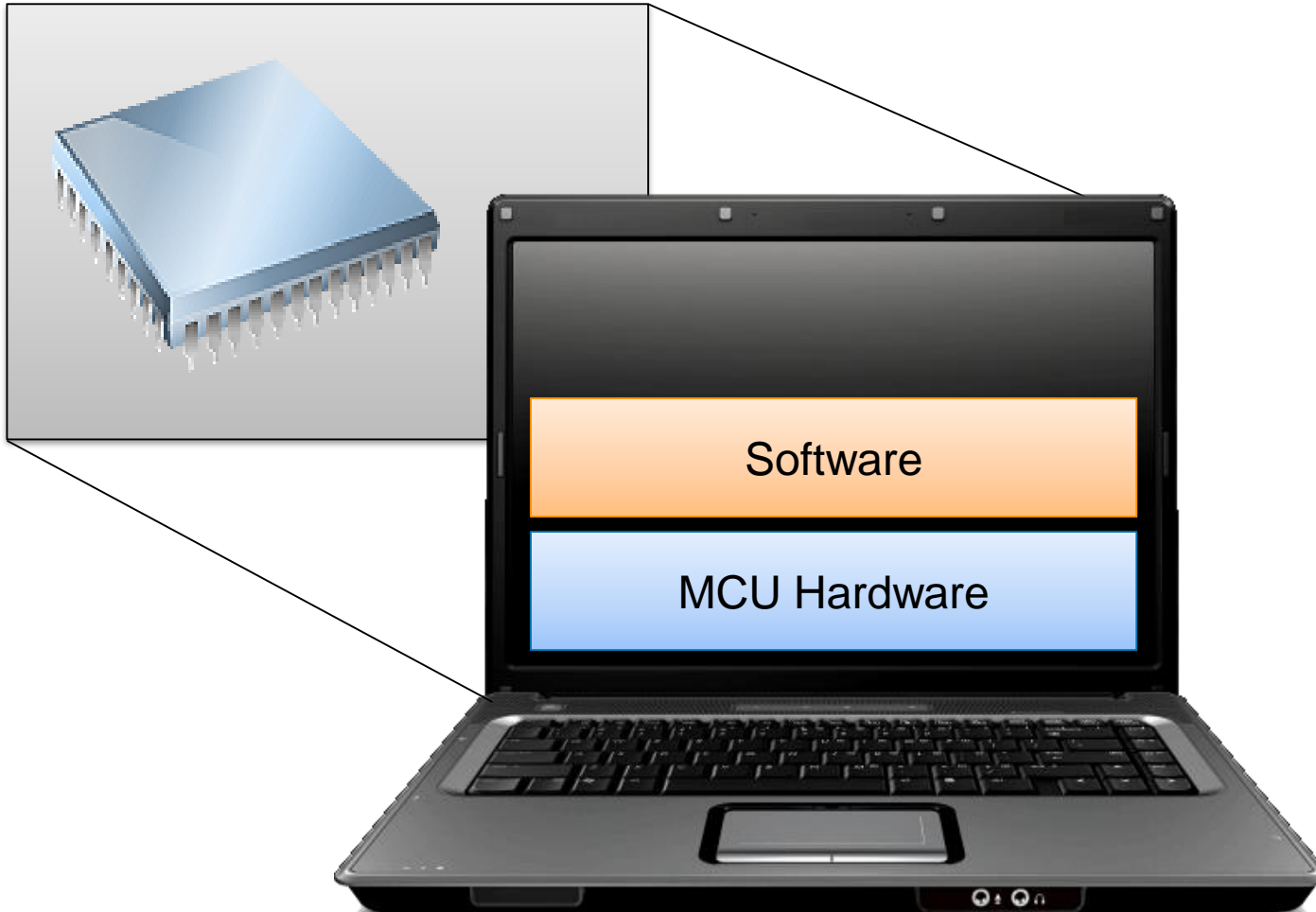


Agenda

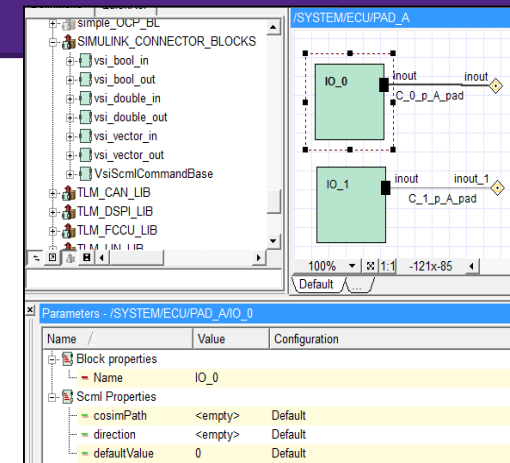
- Virtual Hardware ECU introduction
- Increasing GTM software development efficiency
- Comprehensive solution for GTM based software development and test

Virtual Hardware ECU introduction

What is a Virtual Prototype?



Fast model of a microcontroller (MCU) or system-on-chip (SoC) that can execute an unmodified binary executable



Example NXP MPC5777M

Start HW/SW Development Early

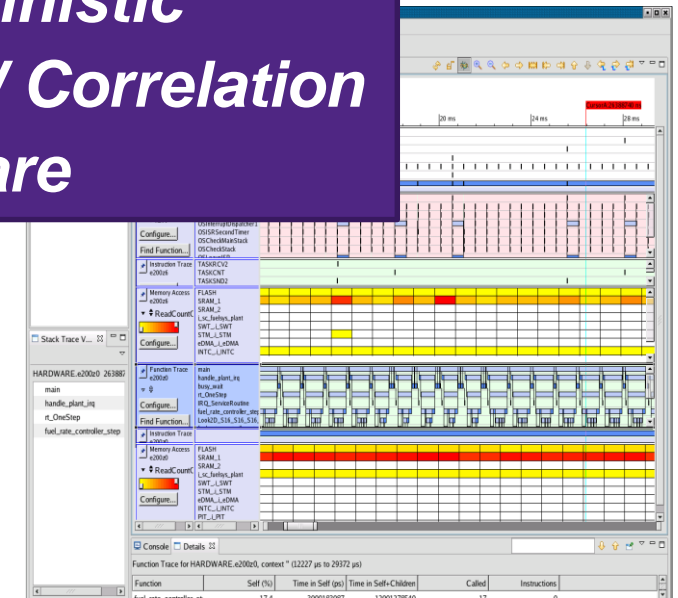
Better than Hardware – available earlier, easier to use for debug!

SW development 9-12 months earlier

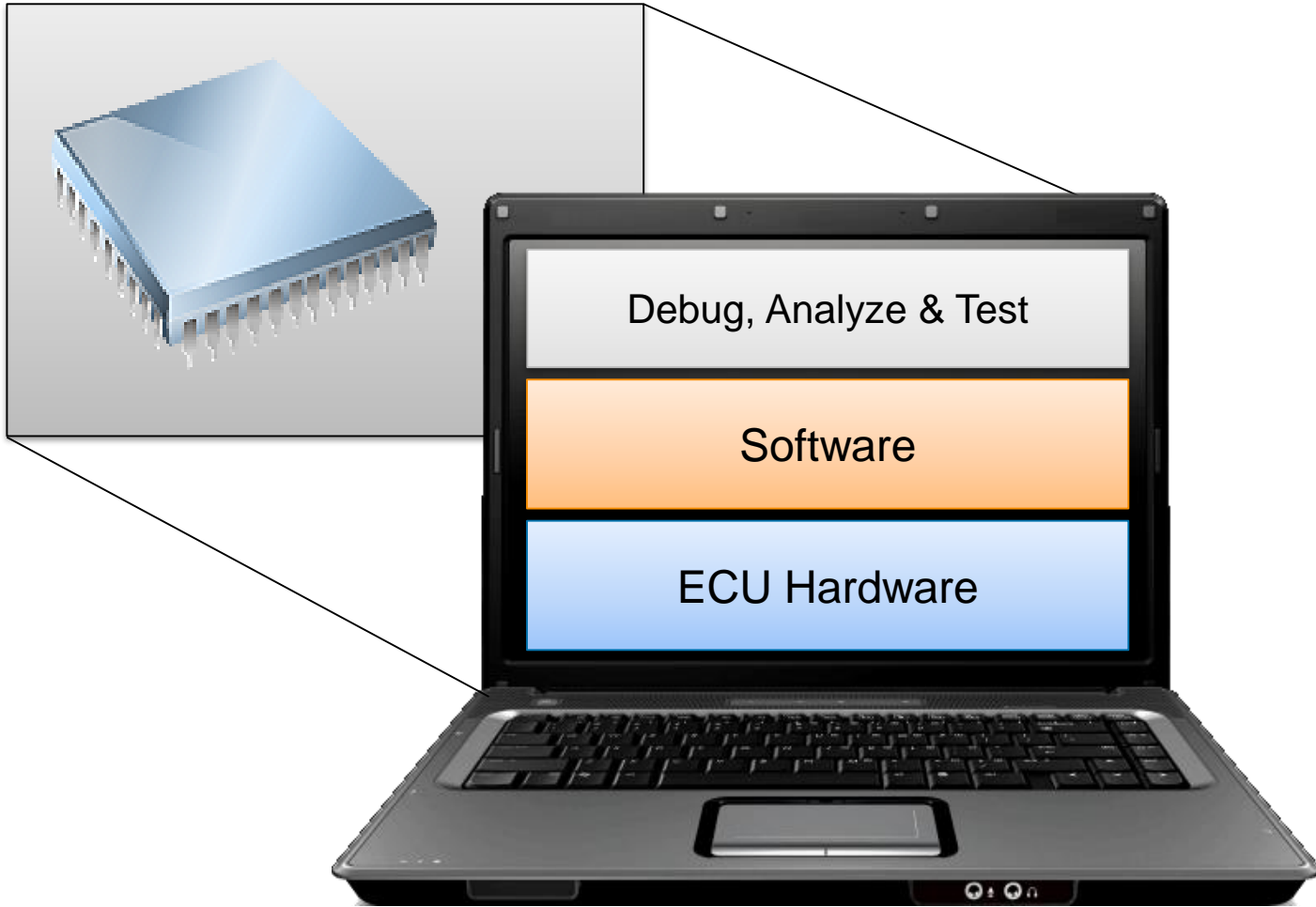
Experience from Automotive Semi and Tier1/OEM SW Teams

- Pre-silicon Dual OS (AUTOSAR & Linux) + Hypervisor bring up.
- Complex driver and communication
 - GTM, CAN, Ethernet communication, vision accelerators/sub-systems, ...
- Algorithm flow from Matlab/Simulink to embedded software

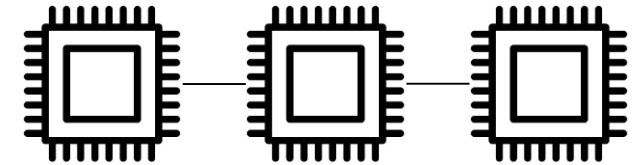
- ***Faster Debug***
- ***Non Intrusive***
- ***Deterministic***
- ***HW/SW Correlation***
- ***OS aware***



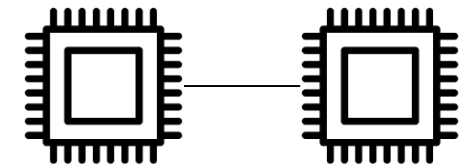
What is a Virtual Hardware ECU?



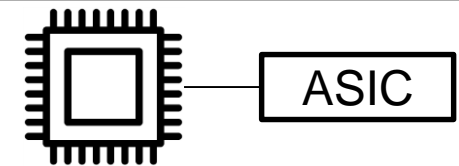
Fast model of the ECU Hardware with the benefits of virtual prototypes



Infotainment ECU



ADAS ECU



Powertrain ECU

Hardware-in-the-Loop

Development Gap

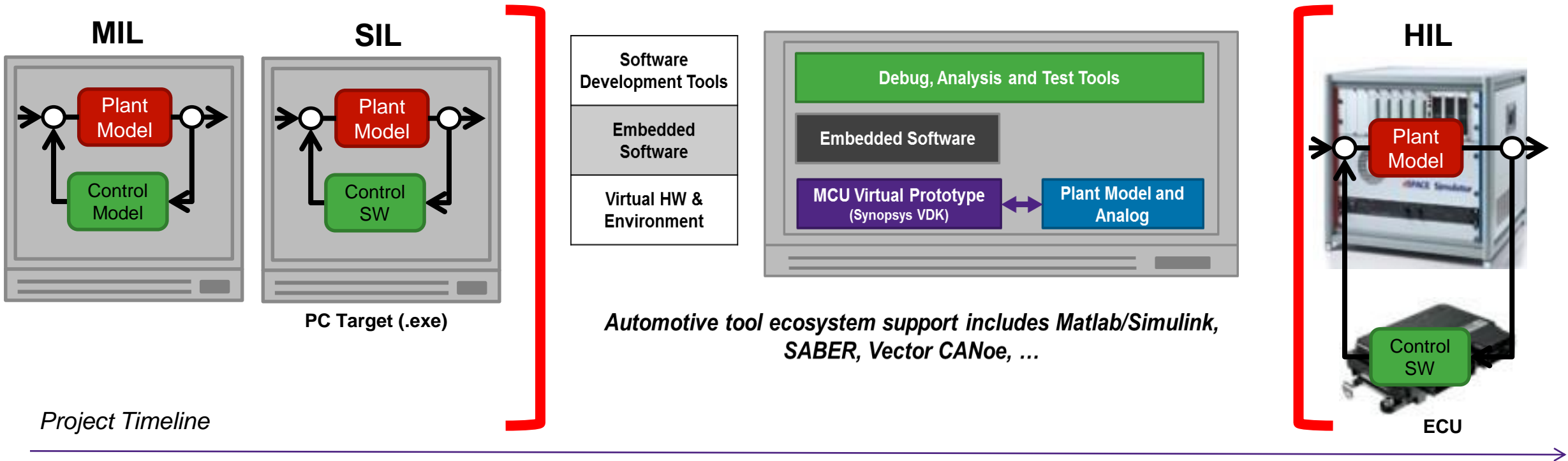


HIL Limitations:

- Access due to limited number of HIL systems (cost and access)
- Limited visibility and controllability
- Hard to deploy in regression
- Complex to set up, share, maintain and archive

Virtual Hardware-in-the-Loop

Start Before Test Benches are Available

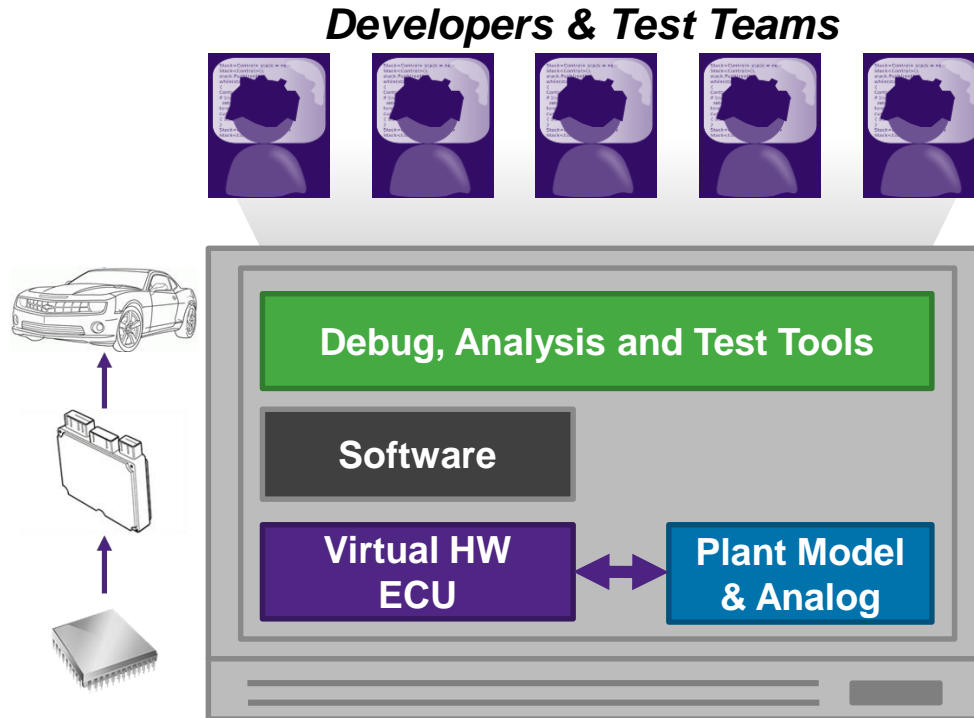


SW Development Early & Increased Testing Throughput

From Virtual SoCs to Virtual Hardware ECU(s) with Virtualizer and Automotive VDKs

Start early & accelerate development

- When SoC evaluation boards or ECU HW boards are not available
- 12-18 months earlier
- Easier and more efficient debug



Start earlier, test faster and better

- Frontload test development
- System SW testing
- Fault & coverage testing
- Regression

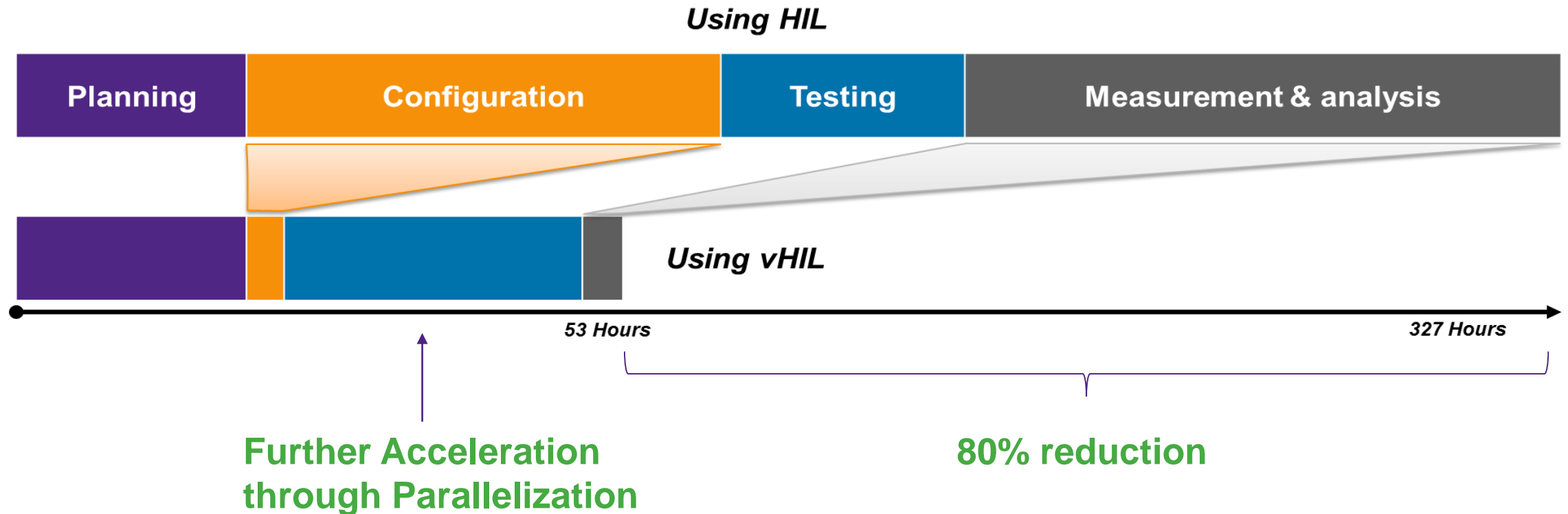
Early Software Development

HW
Availability

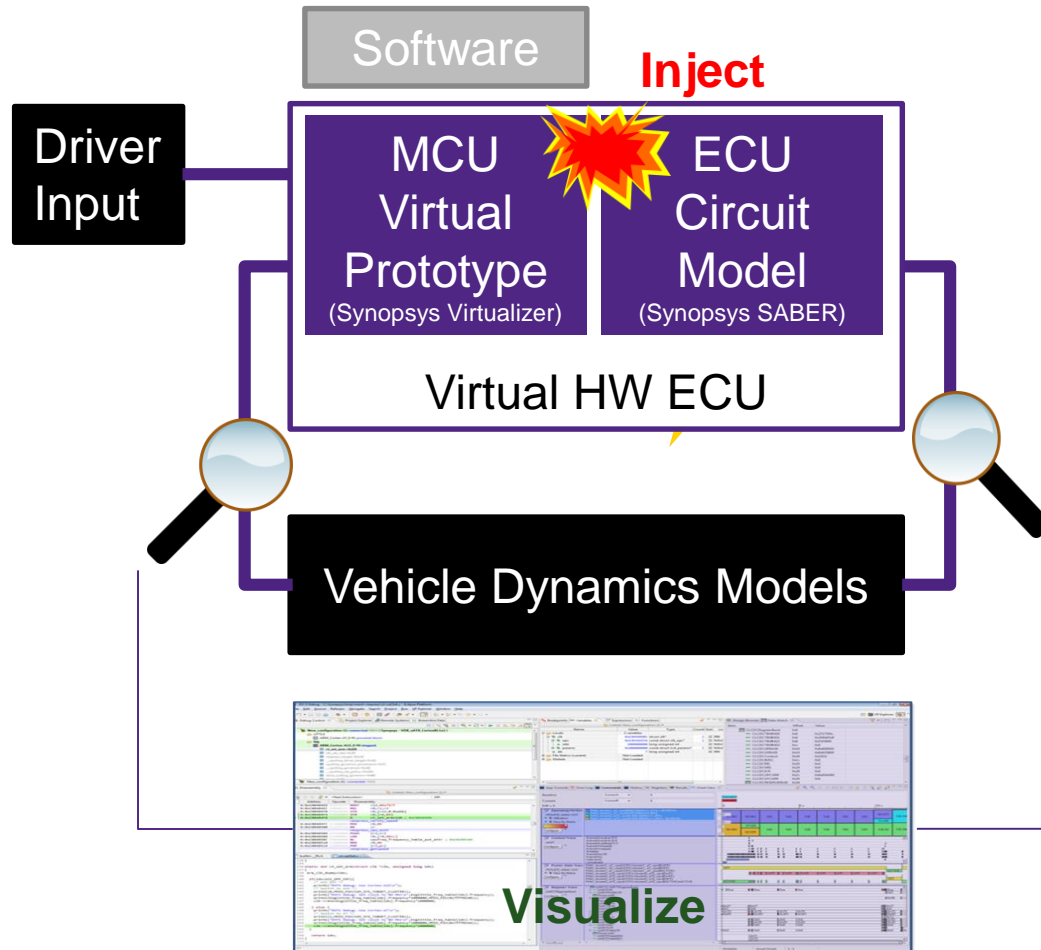
Increase Testing Throughput

Significantly Increase Test Throughput

Test More and Faster – Higher Software Quality Earlier, at Every Development Milestone



Application Example to vFMEA



Actual Results

- Increased test coverage 200 to 900 tests
- Testing effort reduced from 3 man/months to 2 man weeks
- Reusable, safer and distributed access
- Faster analysis of result and change iteration

Increasing GTM software development efficiency

Challenges

- Understanding and debugging GTM software efficiently
- Getting the right visibility into GTM itself for debug, analysis and coverage
- Getting visibility in the context of other processor core and debuggers
- Stimulating the GTM – stimulus, fault injection and associated analysis

MCU Virtual Prototypes and Virtual Hardware ECUs accelerate and simplify GTM based development!

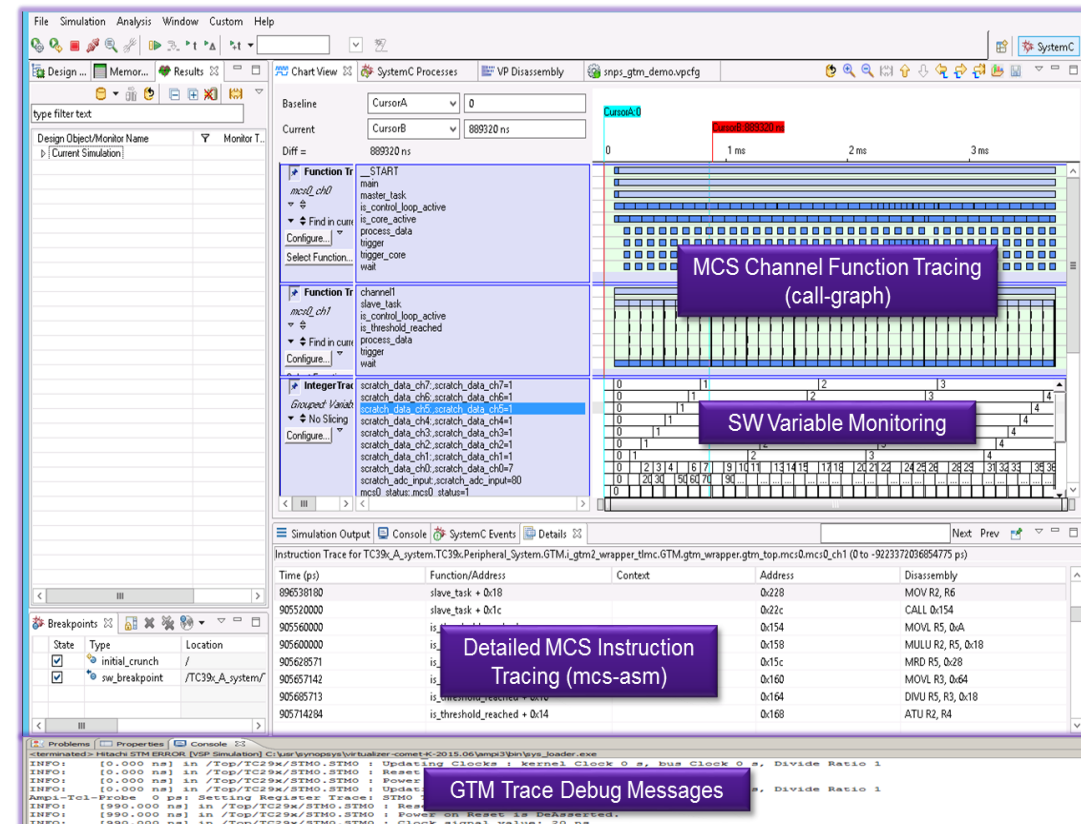
Integrated and Comprehensive GTM Debug, Tracing and Analysis

- Visibility

- MCS Function Tracing
- MCS Instruction Tracing
- MCS core register and GTM config register tracing
- MCS Memory access tracing

- Integrated and comprehensive

- Integrated GTM “internals” tracing in unified console.
- Viewing and Debugging in context of mainline MCU SW execution.
- Interactive and offline



Synchronized Debugging

LAUTERBACH TRACE32

Source code view showing assembly and C code for a core. The register window shows PC at 013C. The variable window shows 'is_core_active()'.

pls Development Tools

Universal Debug Engine (UDE)

Target browser showing connection to TC395-A. Message view showing successful connections and program loading.



Multi-MCS Channel Debugging

Multi-MCS Core Register View

Dissassembly View

Memory Map showing GTM wrapper and mcs0 channels. Register view shows R0-R7, CTRL, ACB, PC, MHB, CTRL_STAT, STRG, ERR, RESET, CAT, CWT. Disassembly view shows instructions for channel7 and channel14.

Memory Views

GTM Config Register Interactivity

Details window showing memory at 0x00000000. Register configuration window showing ATOMS_AGC_ACT_TB, ATOMS_AGC_OUTEN_CTL, ATOMS_AGC_OUTEN_STA, ATOMS_AGC_FUPD_CTRL, ATOMS_AGC_INT_TRIG.

Virtualizer Simulation Scripting

```

def function entry_0 {
    global symbol;
    register + xax MemoryProbe(100, memory, 4, 0);
    # Breakpoint probe and function probe
    # GoodOverwrite + overwrite create logic_instruction observer {function_entry_0, symbol_start_address}
    }
    
```

3rd Party Software Debuggers

+

GTM advanced visibility

GTM MCS Code Coverage

- Non-Intrusive Code Coverage for high level language MCS software
- No code modifications needed in MCS code
- Industry standard LCOV Reports
- Per or multi MCS channel coverage reports

Current view: [top level - snps_gtm_demo - main.c](#) (source / file)

Test: [coverage.TC39x_A_system.TC39x.Peripheral](#)

Date: 2017-04-18 16:21:41

Legend: Lines: hit not hit | Branches: + taken - not taken

Line	Branch data	Line data	Source code
1			/*
2			* main.c
3			* Created on: M
4			* Author: c
5			*/
6			#include "config
7			
8			#define BUSY_WAITING 0
9			
10			volatile data_struct shared_data;
11			
12			693: void wait(unsigned int ID)
13			{
14			693: #if BUSY_WAITING == 0
15			693: wurmX(& STRG, 1 << ID, 1 << ID); // wait until channel gets triggered
16			693: CTRG = 0xFF; // clear trigger bit for current channel
17			693: #else
18			693: while(shared_data.active_channel != ID);
19			693: #endif
20			693: }
21			
22			void trigger(unsigned int ID)
23			{
24			693: #if BUSY_WAITING == 0
25			693: STRG = 1 << ID; // trigger specific channel
26			693: #else
27			693: shared_data.active_channel = ID;
28			693: #endif
29			693: }
30			
31			693: }

LCOV - code coverage report

Current view: [top level - snps_gtm_demo](#)

Test: [coverage.TC39x_A_system.TC39x.Peripheral_System.GTM.i_gtm2_wrapper_tlmc.GTM.gtm_wrapper.gtm_top.mcs0.mcs0_ch0.18220](#)

Date: 2017-04-18 16:21:41

Legend: Rating: low: < 75 % medium: >= 75 % high: >= 90 %

	Hit	Total	Coverage
Lines:	27	62	43.5 %
Functions:	0	17	0.0 %
Branches:	0	0	-

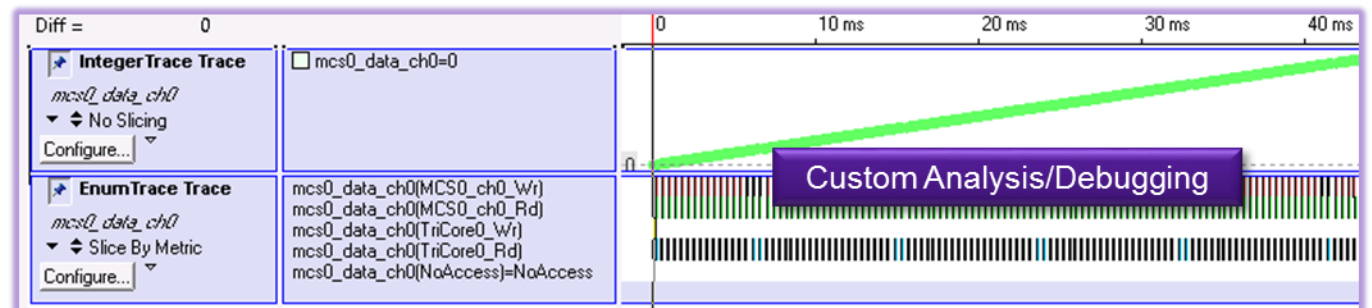
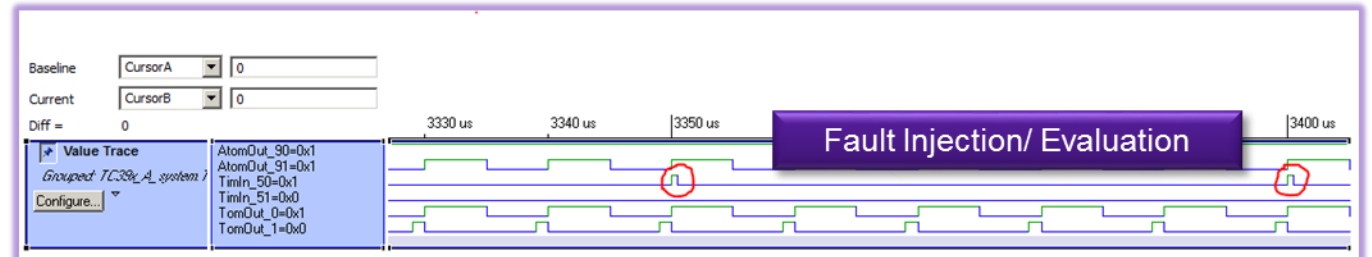
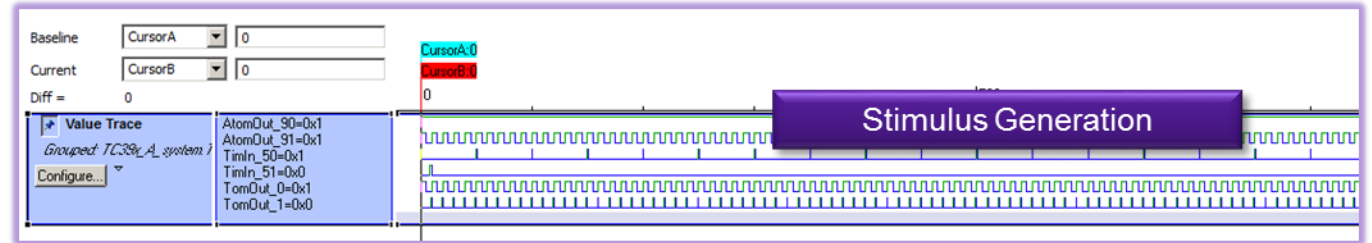
Filename	Line Coverage	Functions	Branches
main.c	43.5 % 27 / 62	0.0 % 0 / 17	- 0 / 0

Generated by: [LCOV version 1.11](#)

Testing and Understanding GTM System Impact

scripting framework for fault injection, signal stimulus and analysis.

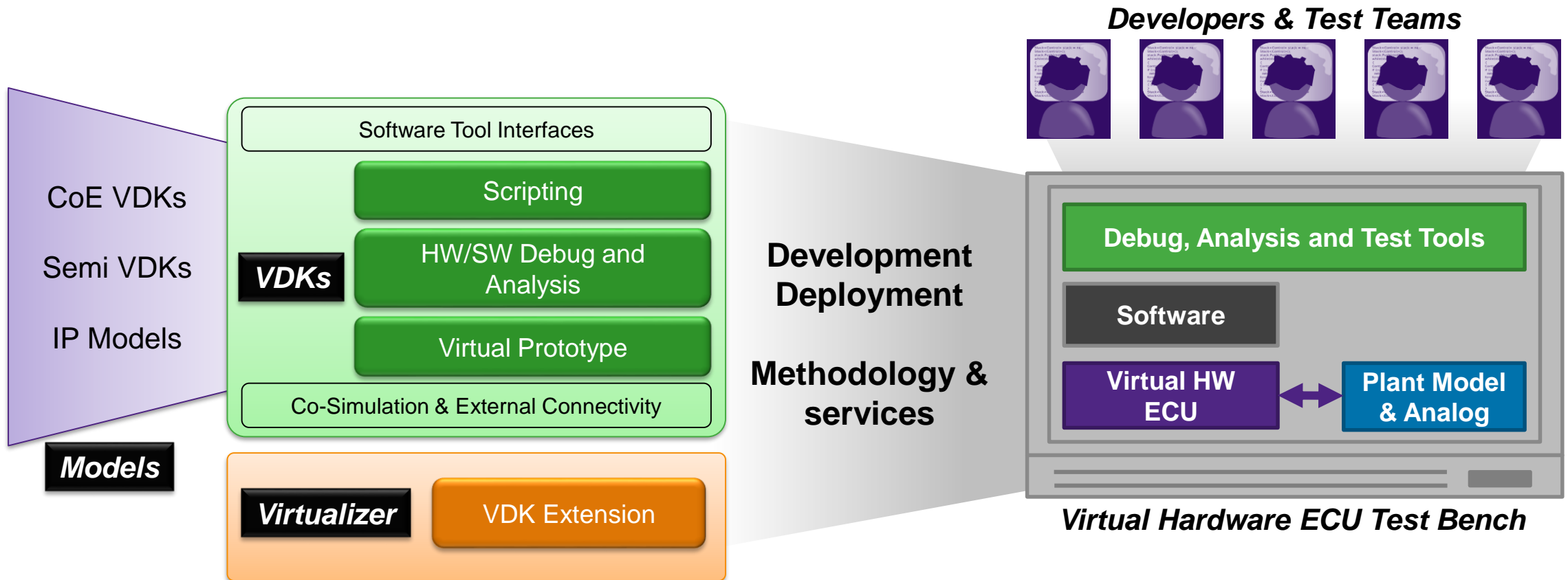
- Scriptable waveform generation for repeatable stimulus scenarios.
- Inject faults into input stimulus or output signals to establish system impact.
- Custom analysis or debugging to investigate
 - Verify data processing
 - Investigate race conditions
 - Playback of “field” issues through the GTM Reference model in MCU VDK.



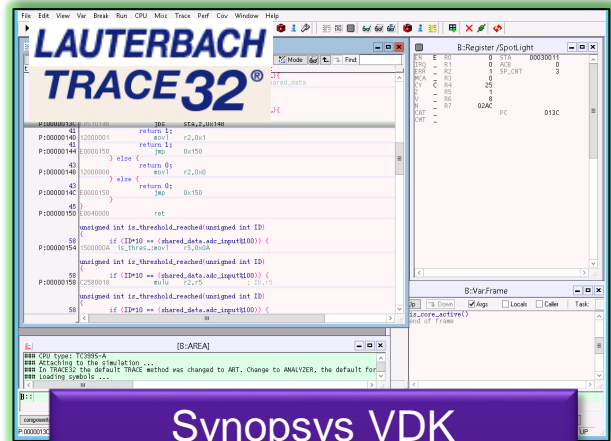
Comprehensive solution for GTM based software development and test

The most comprehensive Virtual Hardware ECU Solution

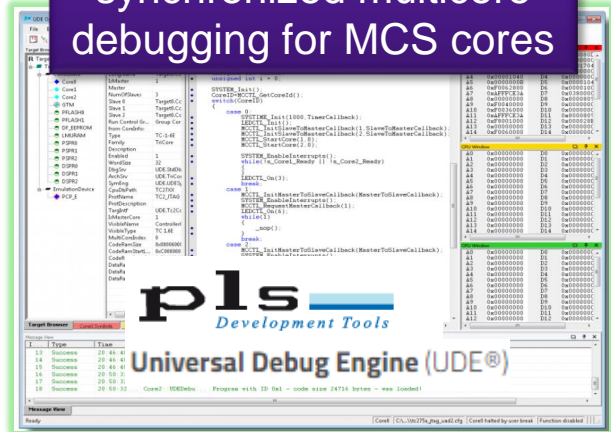
Synopsys Solution from Modeling to Test Bench Deployment



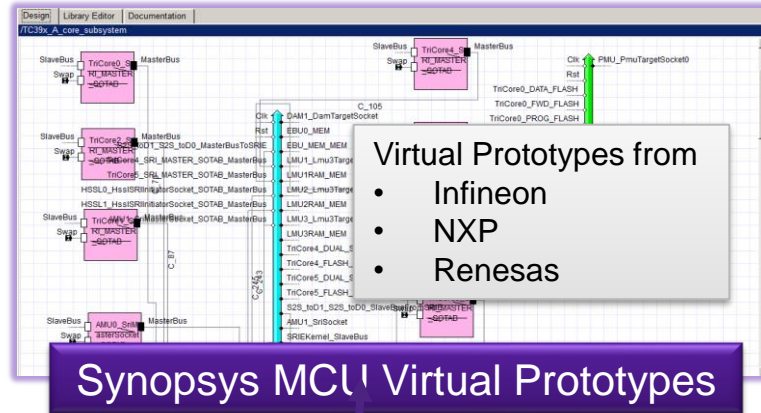
Integrated Environment to Efficiently Develop and Test GTM Software



Synopsys VDK
synchronized multicore
debugging for MCS cores



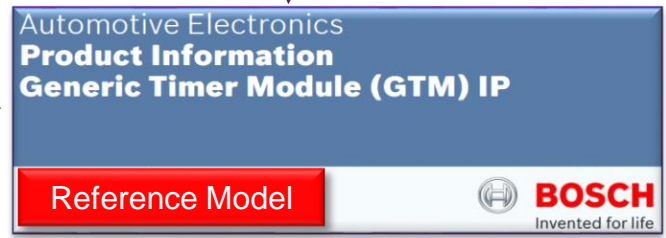
p1s
Development Tools
Universal Debug Engine (UDE®)



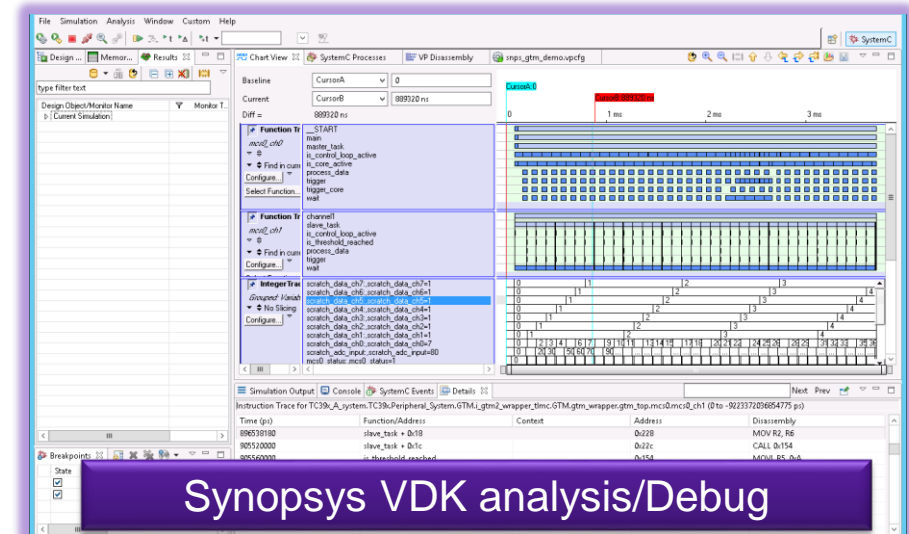
Virtual Prototypes from

- Infineon
- NXP
- Renesas

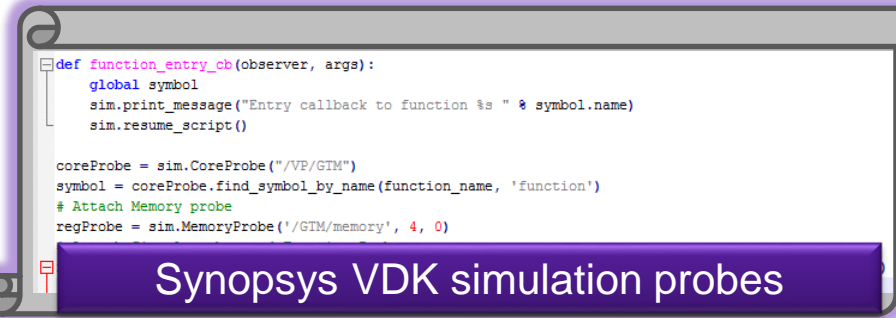
Synopsys MCU Virtual Prototypes



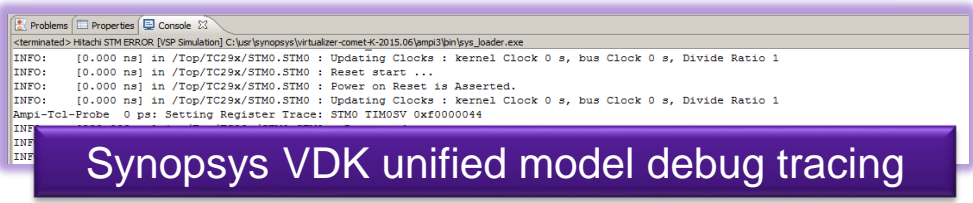
Execute unmodified GTM binary code on the reference Model



Synopsys VDK analysis/Debug



Synopsys VDK simulation probes



Synopsys VDK unified model debug tracing

Summary

- New approaches are needed to start development earlier and increase testing throughput for GTM based designs
- Virtual prototypes and Virtual Hardware ECUs deliver key benefits – earlier availability and faster testing throughput
- Synopsys has the most comprehensive solution in the market specifically tailored for GTM development
 - Tools enhancing GTM debug visibility and control
 - Full MCU models from companies such as NXP, Infineon and Renesas
 - Proven deployment for more than 10 years
 - Financially stable, long term vision and recognized industry leader

The advertisement features a purple and blue background with a white box containing the Synopsys logo and product details. The main title is 'Most Comprehensive GTM-based Virtual Development Solution'. Below the title are four bullet points: 'Accelerate early software development 12-18 months before MCU silicon is available', 'Increase testing throughput using Virtual Hardware ECU', 'Most complete GTM debug and analysis solution', and 'GTM support in MCUs from Renesas, NXP, Infineon and ST'. To the right, a white box contains six blue boxes with white text: 'GTM synchronized debugging and visibility', 'GTM integration with 3rd party debuggers', 'GTM tracing and analysis', 'GTM code coverage', 'GTM fault injection', and 'Broad MCU support with GTM'.

synopsys®

Most Comprehensive GTM-based Virtual Development Solution

- Accelerate early software development 12-18 months before MCU silicon is available
- Increase testing throughput using Virtual Hardware ECU
- Most complete GTM debug and analysis solution
- GTM support in MCUs from Renesas, NXP, Infineon and ST

GTM synchronized debugging and visibility	GTM integration with 3rd party debuggers
GTM tracing and analysis	GTM code coverage
GTM fault injection	Broad MCU support with GTM

SYNOPSYS[®]
Silicon to Software[™]