

Model-Based Design and Virtual ECUs Enable Early Testing for Electric Vehicle Controls

Authors

Kevin Brand
Synopsys

Dineshkumar Selvaraj
Infineon

Overview

With the shift towards the Software Defined Vehicle (SDV), automotive-focused, heterogeneous system-on-chips (SoCs) with highly parallel specialized hardware acceleration are required to address new, more complex functions with higher performance. The difficulty for parallel computing architectures is that there are many different programming models available, which are not well aligned with safety-critical software development paradigms. Leveraging Model-Based Design (MBD) can overcome the challenge of moving algorithms from a classic scalar architecture to a parallel architecture. However, traditional MBD flows must also evolve with compute needs, so that tooling can generate high-performance code targeting those heterogeneous compute architectures. Virtual Electronic Control Units (vECUs), or virtual prototypes, can play an important role by avoiding hardware dependencies and taking advantage of the visibility and debugging power of a pure virtual environment during software testing.

This whitepaper provides a comprehensive case study on the modeling and software verification of a Field-Oriented Control (FOC)-based motor control system using MBD and vECUs. The focus is on the development of an electric motor control ECU for electric vehicles, from initial design concepts to full software implementation, deployed across multiple cores of Infineon® AURIX™ TC4x architecture. The AURIX TC4x virtual prototype was developed with Synopsys Virtualizer™ tooling, which is crucial for scaling with the increasing complexity and computational demands of modern electric vehicle systems.

The study highlights the utilization of the FOC algorithm, essential for optimizing power usage in Lithium-Ion (Li-ion) battery-driven electric motors (Figure 1). Additionally, it explores sensor-less position estimation using an Artificial Intelligence (AI) technique based on a Multilayer Perceptron (MLP) architecture. It also discusses the generation of hardware-optimized code for the ARC™ EV-based Parallel Processing Unit (PPU) of AURIX TC4x Microcontrollers.

Results show that an enhanced workflow for Virtual Processor-In-The-Loop (vPiL) and Virtual Hardware-In-the-Loop (vHiL), executed with a vECU, significantly reduces turnaround times for application developers.

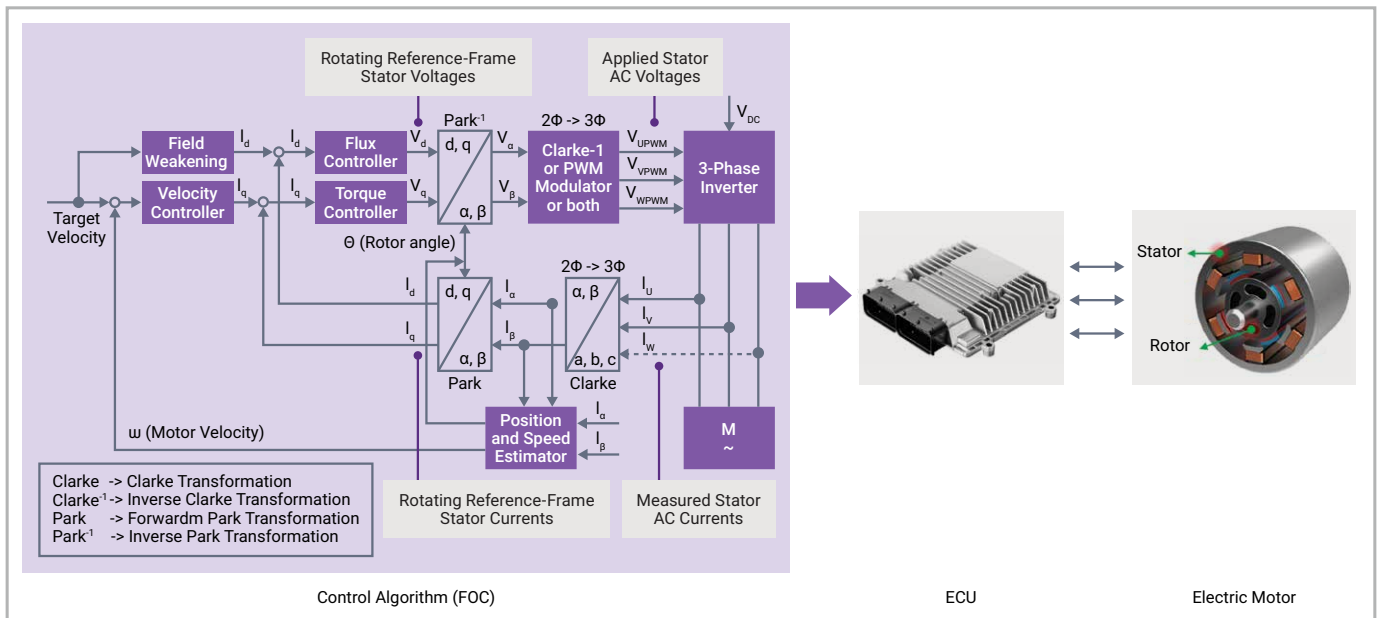


Figure 1: FOC algorithm for electric motor

Electric Vehicle Control Application

The implementation of MLP functionality can be efficiently achieved through software running on a specialized hardware accelerator, capable of meeting the high computational bandwidth required by the application. Traditional software components of the FOC algorithm can be managed by more generic "host" cores. By utilizing specialized cores for the compute intensive software components with efficient communication between these cores, we can realize a heterogeneous closed-loop functionality. Leveraging MBD workflows allows for the development and deployment of algorithms and specialized code generation flows for newer heterogeneous cores, moving beyond the constraints of generic software design and single processing unit implementations.

MBD workflows (Figure 2) typically follow a system-to-implementation approach, where verification is conducted in a closed loop at each step, using simulations that match the design's detail level at each stage. These verification stages include Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), and Hardware-in-the-Loop (HiL). The development workflow remains independent of the target hardware until the SiL stage. Upon reaching the PiL stage, there is a dependency on at least one development board to execute the cross-compiled software algorithm on the real target, ensuring numeric accuracy, execution time, and functional correctness.

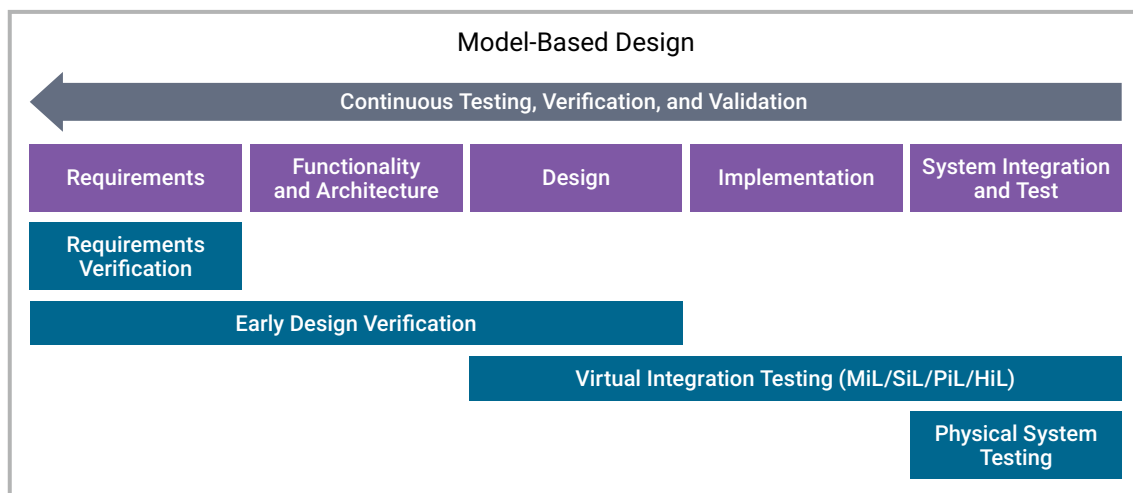


Figure 2: vHiL in a MBD flow

As the software is refined and becomes more complete, incorporating numerous hardware-dependent drivers, transitioning to a HiL setup becomes essential. By replacing hardware dependencies in the development process with vECUs, we can extend hardware independence further along the development timeline. This shift enables more scalable deployment, transforming the PiL and HiL stages into vPiL and vHiL flows.

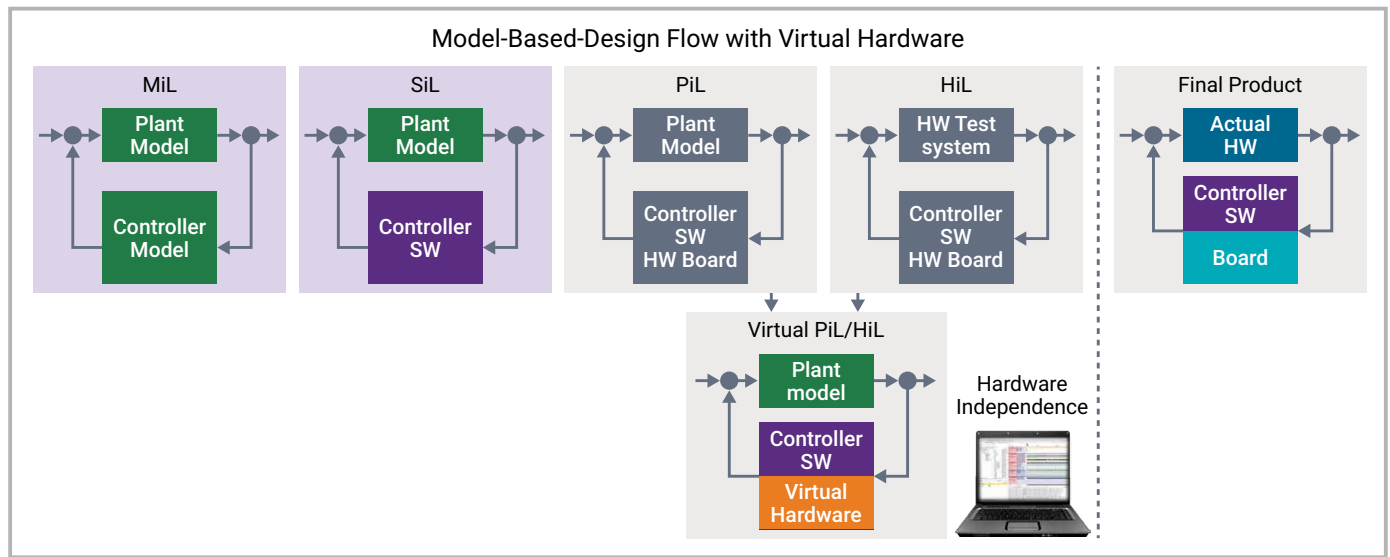


Figure 3: Gaining hardware independence with virtual hardware

vECU, Target Model, and Software

This case study uses the Infineon® AURIX TC4x System-on-Chip (SoC), which features an advanced six TriCore™ host architecture along with a programmable hardware accelerator, the PPU, based on the ARC™ EV71 core. This core integrates a SIMD vector DSP and a scalar core, which is ideally suited for AI workloads.

For the development process, we utilized the MATLAB® and Simulink® toolchain due to its compatibility with Infineon's Embedded Coder® Support Package for AURIX TC4x Microcontrollers. This toolchain facilitates rapid prototyping on single cores and, when combined with the SoC Blockset™ Support Package for AURIX Microcontrollers, it enables the design and deployment of multi-core models. It is particularly effective for partitioning algorithms across multiple TriCore cores and generating optimized code for the SIMD PPU using code replacement techniques and inter-process communication (IPC) mechanisms between cores.

In addressing the requirements for sensor-less FOC in electric vehicles (xEV), this case study necessitates the execution of multicore TriCore processes that communicate with the MLP implemented on the PPU. The PPU leverages the Synopsys ARC EV71 Core IP software libraries, and the following generation and build flow is established (Figure 4).

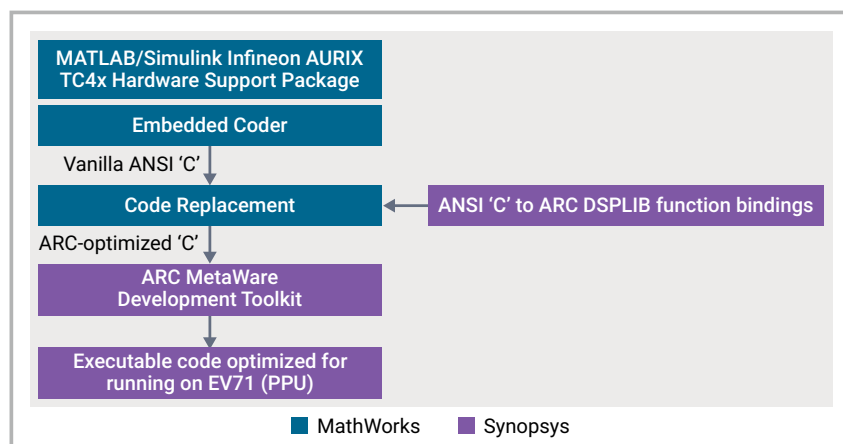


Figure 4: Optimized code generation flow for ARC-based PPU

Embedded Coder facilitates automatic code generation from Simulink models and performs automatic code replacement of standard ANSI/ISO C functions with highly optimized functions from the MetaWare™ DSP libraries. The MetaWare Development Toolkit then compiles the generated vector DSP C code. This comprehensive software generation and build flow is employed throughout the case study. With the TriCores also executing, the heterogeneous output is fully operable on the level 4 virtual prototype (Figure 5) of the TC4x, which was developed with Virtualizer tooling.

Synopsys TC4x Virtual Prototype

A virtual prototype, or vECU, can be represented at various levels of abstraction. Setting aside the highly abstracted MiL and the physical HiL for the moment, vECUs can generally be categorized into two main groups: "host compiled," where the embedded software is compiled for the host system, and "target compiled," where the embedded software is compiled for the target hardware board.

At levels 1 through 3, where SiL is employed via Synopsys Silver™, the software undergoes refinement starting from model-based code generation and progressively incorporating more elements of the final software stack. In this case study, we concentrate on level 4 vECUs, where the software is compiled specifically for the target hardware (Figure 5). Consequently, the abstraction level of these vECUs must be adequate to meet the requirements of running the target binary build.

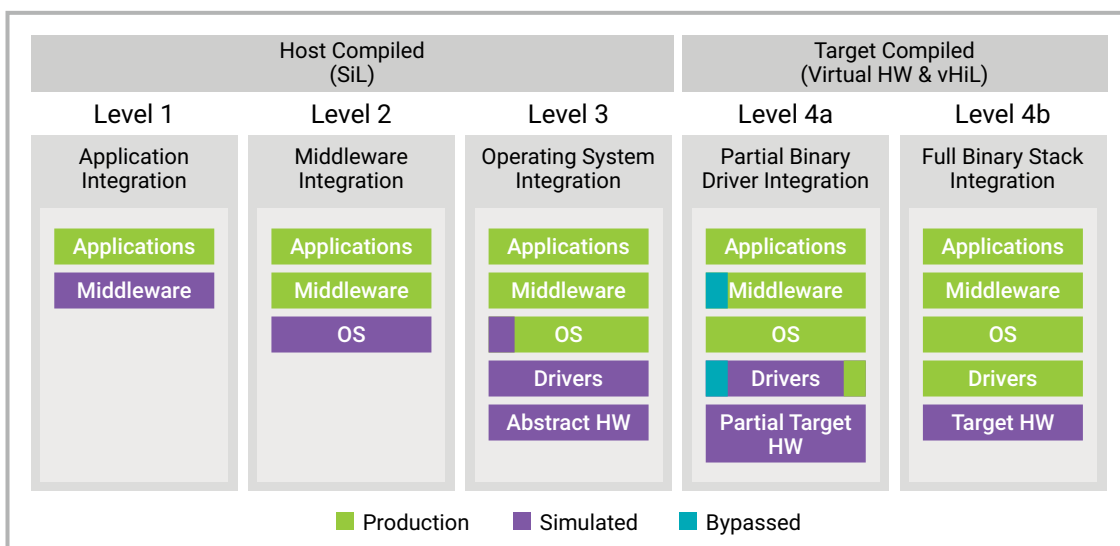


Figure 5: vECU abstraction levels

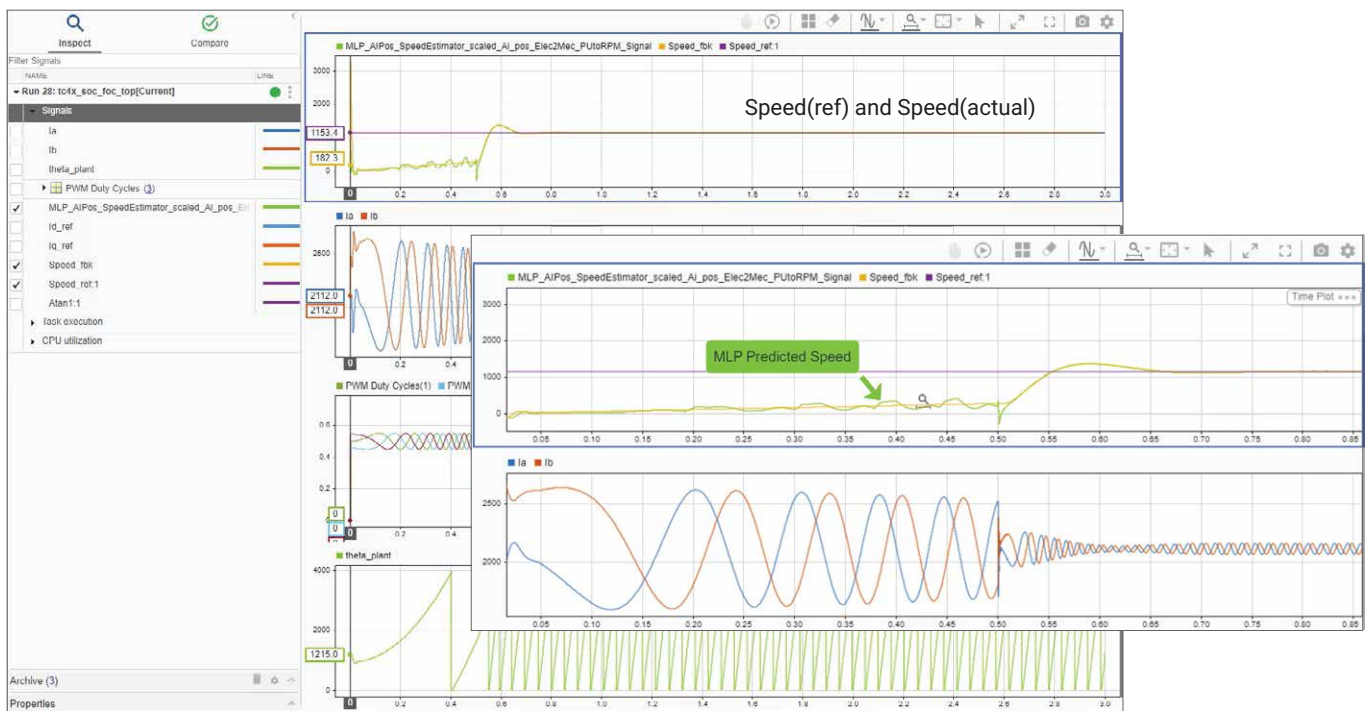
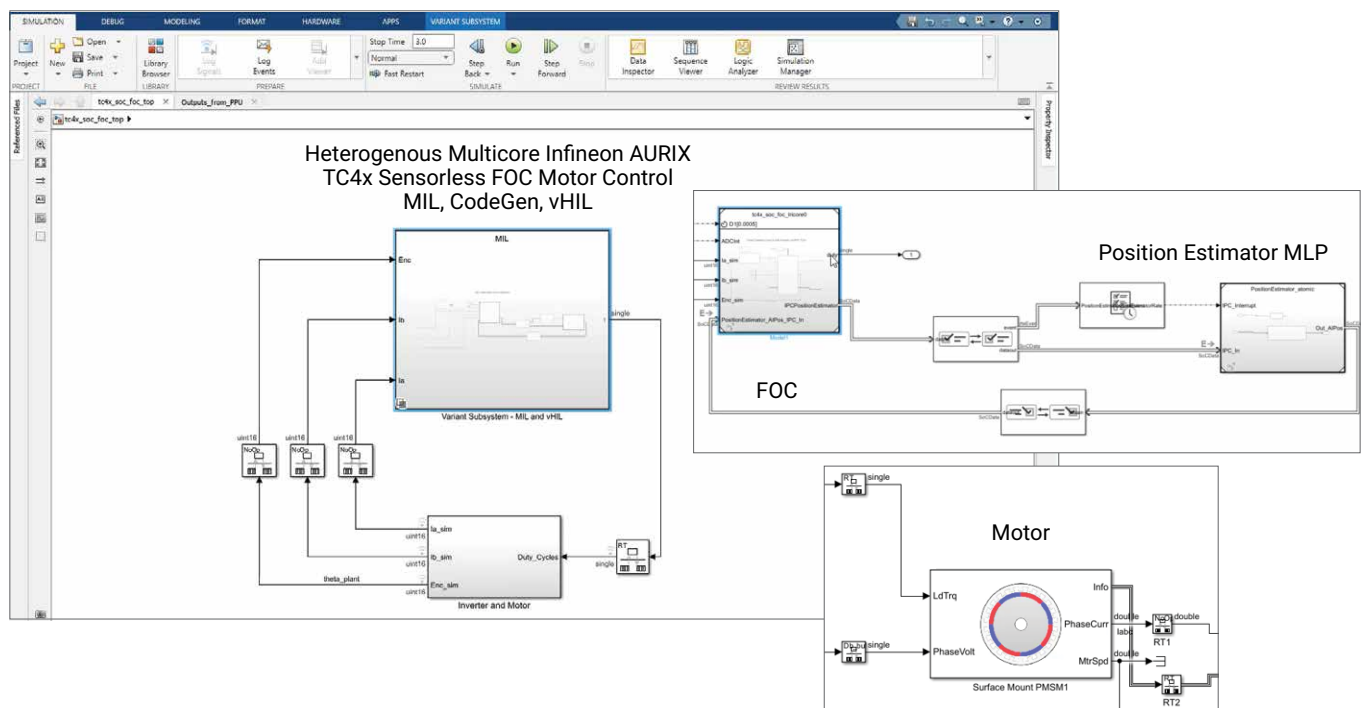
The TC4x virtual prototype utilized in this case study was developed with Virtualizer tooling and is designed to run multicore software on the TriCore host cores and the MLP position estimator on the PPU, all accurately modeled within the virtual prototype. Additionally, the FOC algorithm depends on certain hardware functionalities such as the ADC for sampling current inputs and PWM motor inverter drive. These hardware components are fully available and can be executed using production drivers within the virtual prototype environment.

Results

The results of this case study focus on design, simulation, and verification processes for the three key phases: MiL, vPiL, and vHiL.

Model-in-the-Loop

The initial phase involved system-level design captured at a high level, which encompasses the system requirements and behaviors of the control and MLP algorithms, as well as the motor plant model. One of the primary objectives was to explore the potential for efficiency improvements by creating a single-source design that could run a MiL top-level reference design, generate code, and facilitate virtual prototyping. The use of variant subsystems in Simulink enabled the creation of this single-source design (Figure 6), which is crucial for an efficient workflow. Upon completing the MiL design in Simulink, the reference results were achieved through simulation (Figure 7).



Virtual Processor-in-the-Loop

In the next phase, we advanced the design flow using the virtual PiL technique. This involved co-simulation between Simulink and the virtual prototype in Synopsys Virtualizer™. Typically, when Simulink runs a PiL simulation with real hardware, it uses a serial link to communicate test commands and exchange data (usually via the XCP protocol) with the application running on the target. For the co-simulation setup, virtual COM ports were employed to provide the same interface to Simulink, which remained unaware that it was communicating with a simulation. During this vPiL phase, only single cores were evaluated simultaneously, as a multicore PiL flow was not available in the MathWorks/Simulink tooling at that time. Despite this limitation, the vPiL phase was instrumental in confirming the target-compiled operation without needing a hardware board (Figure 8).

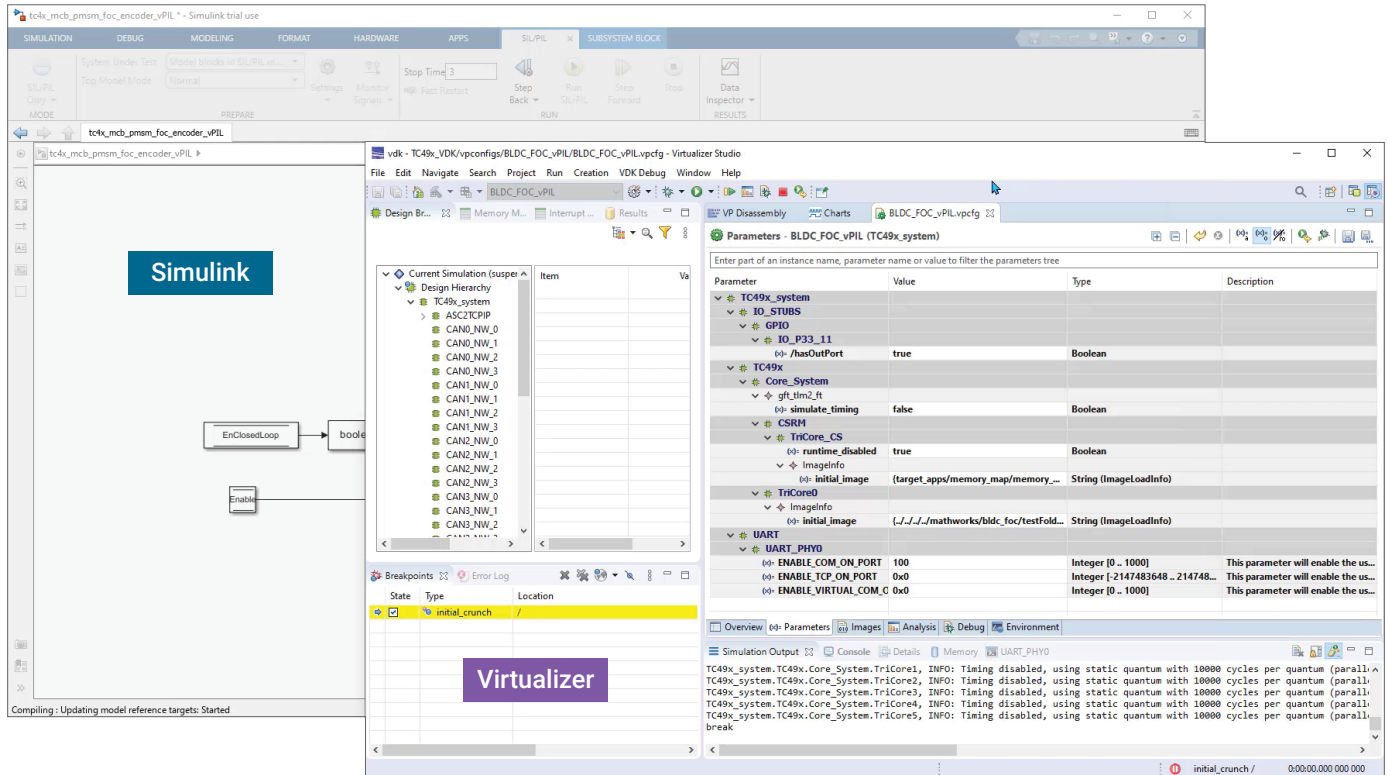


Figure 8: vPiL with Simulink and Virtualizer

Virtual Hardware-in-the-Loop

The final phase involved performing a full code generation step for the entire application, targeting virtual hardware while still simulating the electrical motor plant model in Simulink (Figure 9). Utilizing the variant subsystem feature in Simulink allowed for a seamless switch between simulating the MiL model, generating code (resulting in a target binary), and representing the model-based target image with the virtual hardware running the compiled target image. As the final binary targets hardware, it includes all necessary hardware-dependent software.

The inputs and outputs of the virtual prototype can be at various abstraction levels to match the abstraction of the plant model in Simulink. Initially, at a higher abstraction level, it is possible to connect the I/O signals in Simulink directly to the variables in the software, achieving a 1:1 match between software variables and signals. Running the full software stack allows many peripherals to be exercised and verified at this first abstraction level in a closed loop. At a lower abstraction level, real I/O on hardware pins, such as ADC inputs and digital signal outputs, can be connected directly to the Simulink model.

The vHiL setup uses a fully synchronous co-simulation interface, providing a repeatable and debug-capable simulation for both hardware models and software execution, even though it does not run in real-time. The co-simulation results from both the Simulink signal aspect and the HW/SW aspect of Virtualizer are shown in Figure 10. The highlighted traces indicate a close correlation with the original MiL results, demonstrating that the MiL results have been verified and we have reached the implementation binary stage with vHiL, achieving correlated results without any dependency on hardware.

Conclusion

This case study underscores the effectiveness of virtual prototyping and co-simulation in modern automotive ECU development. MathWorks Embedded Coder and its associated tools generated optimized, efficient model-based software. This was combined with a virtual hardware environment for the AURIX TC4xx family of MCUs, developed with Virtualizer.

By adopting these methodologies, developers can achieve higher efficiency, greater flexibility, and more robust solutions while reducing dependency on physical hardware. This approach not only accelerates the development process but also ensures that the software meets the rigorous requirements of next-generation automotive systems.

For over 15 years, Infineon and Synopsys have collaborated to create level 4 virtual prototypes of their leading MCU titles, allowing Infineon customers to explore and validate software solutions before hardware becomes available. For the latest generation MCU, AURIX TC4x, a virtual prototype has been developed using Virtualizer tooling. This includes functional models of nearly all relevant hardware blocks necessary for running software binaries, from bare metal application software to full production software hardware-dependent stacks. This capability makes it an excellent target for reducing hardware tests in the design flow using vECUs.

Acknowledgement

Tito Tang, Mathworks, co-author of "A New Era for Software Verification: Heterogenous Multicore Compute with Model-Based Design and Virtual ECUs", ELIV, International VDI Congress, 2024.