Enhancing Automotive ECU Design with Digital Twin Simulation: Comparative Study of Virtual Platform, FPGA Prototyping, and Edge Device Configurations

Sara M. Abd AlWahab∗†, Mohamed AbdElsalam∗, Ahmed H. Khalil†, Hassan Mostafa†

{sara.abdalwahab, mohamed.abd_el_salam_ahmed}@siemens.com, hmostafa@uwaterloo.ca

†Electronics and Communications Engineering Department, Cairo University, Cairo, Egypt

∗Siemens Digital Industries Software, Cairo, Egypt

***Abstract*- Increasing complexity of automotive Electronic Control Units (ECUs) demands a seamless integration between pre-silicon and post-silicon design phases to accelerate development and improve reliability. This paper presents a novel methodology leveraging Digital Twin Simulation to bridge this gap. We demonstrate the approach using an adaptive headlight application across three ECU representations with different setups: a virtual SoC platform, an FPGA prototyping system (pre-silicon) running with Interface Communication Extension (ICE) or co-model channel/Testbench Acceleration (TBX) support, and an edge device NVIDIA Jetson TX2 (post-silicon). Each configuration maintains the same stimuli from a scenario simulator and uses the same ego vehicle dynamics, enabling progressive refinement and performance evaluation of the ECU design. This work contributes to building a scalable and reusable framework for automotive ECU verification and deployment.**

***Keywords*- Automotive ECU, Virtual Platforms, FPGA Prototyping, Digital Twin Simulation.**

I. INTRODUCTION

In the automotive industry, the development of Electronic Control Units (ECUs) must be tightly integrated with real-world vehicle dynamics and operational contexts to ensure functional reliability, safety, and performance. Digital twins play a pivotal role in achieving this alignment by enabling continuous synchronization between physical systems and their virtual counterparts, thus supporting predictive validation and lifecycle optimization across product, process, and system levels [1,2].

Modern ECU designs typically follow a structured, multi-stage development process consisting of model-in-the-loop (MiL), software-in-the-loop (SiL), processor-in-the-loop (PiL), hardware-in-the-loop (HiL), and real-vehicle testing phases. This systematic approach allows engineers to verify control algorithms and system behavior progressively—from virtual simulation environments to physical hardware prototypes—thereby ensuring functional correctness and robustness. Implementing such a structured development methodology, which integrates simulation, rapid control prototyping, and hardware validation, requires a flexible, scalable and reusable framework for ECU verification and deployment. Digital twin simulation can play a pivotal role in this framework to connect pre-silicon and post-silicon designs with sensors, scenarios, and mechatronics simulations.

A digital twin is a virtual replica of a physical asset capable of mimicking its behavior to better assess and understand its operation under both hypothetical and real scenarios [3, 4]. The digital twin's ability to mimic real-world operating conditions and system behaviors provides great insight into the design's performance characteristics. Furthermore, building a digital twin-based methodology to deploy the software developed on the cloud on a Pre-Silicon representation of an edge ECU has the benefit of evaluating different design architectures of an ECU before fabrication.

This paper proposes a novel approach that uses Digital Twin simulation to seamlessly integrate the pre-silicon and post-silicon phases of the ECU. The proposed digital twin framework demonstrates exceptional flexibility through its adaptable configuration architecture, enabling seamless validation across multiple platforms and development stages.

This setup allows for effortless transition between virtual platform simulations, FPGA prototyping platform operating either with Interface Communication Extension (ICE) or Co-model channel/Testbench Acceleration (TBX) [5], and real board implementations while maximizing the amount of reused work between different configurations. This unified approach ensures that verification scenarios remain consistent across all platforms, while the underlying infrastructure can be easily modified to accommodate different validation requirements and debugging needs.

II. COMPOSITIONAL SIMULATION INTERCONNECT FRAMEWORK

In [6, 7], we presented a compositional simulation interconnect framework for exploring Software-defined Vehicle implementations from cloud to edge, as shown in Fig. 1. The framework enables heterogeneous clients' connections to create digital twins of Electronic Control Units (ECUs) of a Vehicle at different abstraction levels connected to real traffic scenario simulators, sensor models, and mechatronic systems.

In this paper, we extend this work to Pre-Silicon/Post-Silicon Integration using ProFPGA with ICE interface and NVIDIA Jetson TX2 board HW integration to demonstrate how we can seamlessly move from one representation of an ECU design to a higher fidelity one, while preserving the stimulus, mechatronics components, and other digital twin components of the previous step.

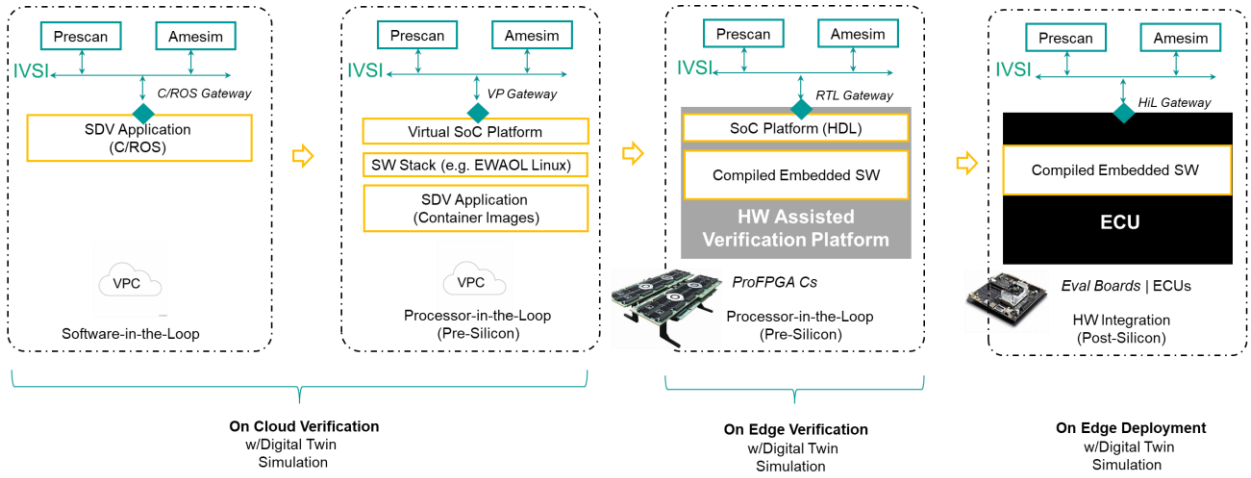

Figure 1. Verification and deployment stages with Digital Twin simulation

The framework allows connections of different client types e.g. sensor/scenario simulators, mechatronic systems simulators, dashboard software, cloud services, compute/think modules at different abstraction levels including C/C++ modules, Python and any platforms that have C/C++/Python interfaces (e.g. ROS), Virtual SoC Platforms, RTL designs on digital simulators/HW assisted verification platforms and external HW boards with physical connections (e.g. Ethernet, CAN) [8].

The design methodology behind the framework is based on three enabling pillars: the gateway concept, Digital Twin Description Language (DTDL), and automation flow [8], as shown in Fig. 2.

A. The Gateway Concept

Each digital twin component port is connected to the backplane server through a "gateway". A gateway executes data conversion and routing into simulated & physical communication protocols. Gateways are categorized into Language2Protocol (C++/C#/SystemC/Python/ROS), Pre-Silicon (VPs, RTL), Post-Silicon (Physical), and Specialized (Sensors, Actuators, Cloud Services, 3rd party Tools). Communication Protocols supported by the interconnect backplane include, for example: Ethernet, CAN-FD, LIN, GPIO, I2C, SPI, and AXI.

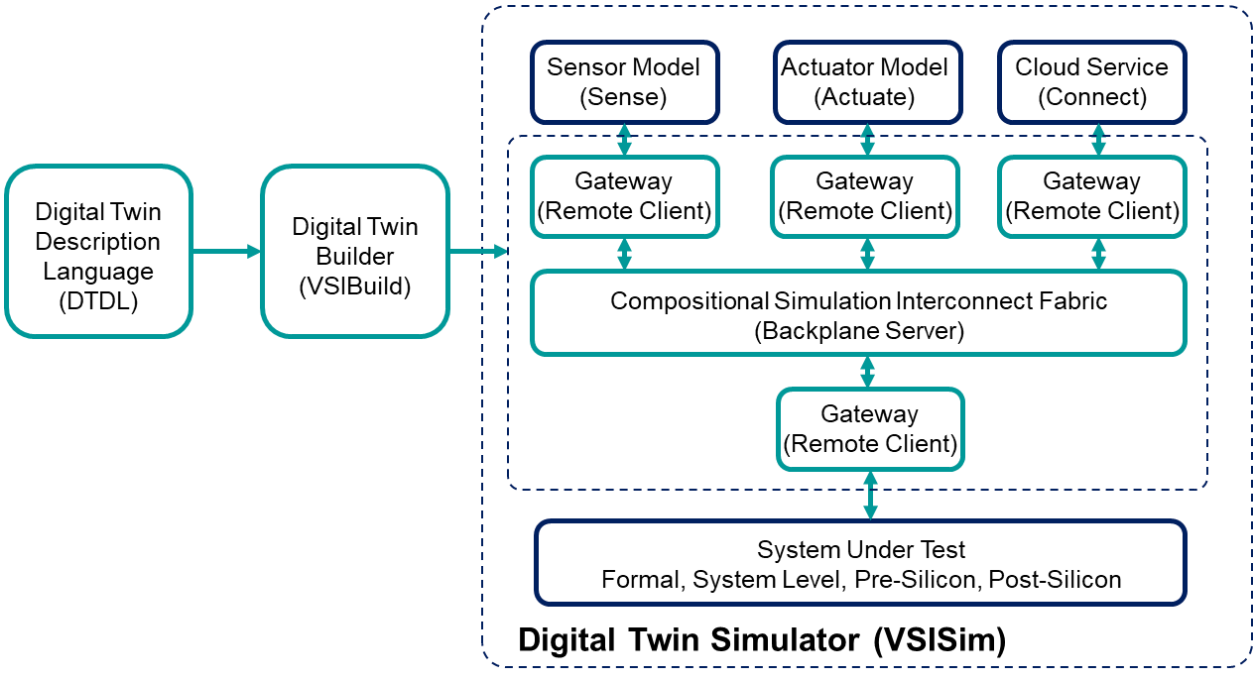

Figure 2. Digital Twin Interconnect Framework

B. Digital Twin Description Language

This is an intermediate & proprietary language format upon which system modelling languages can be built on top. It defines the connectivity between different digital twin components. Details on the syntax and semantics of the language is beyond the scope of this paper.

C. The Automation Flow

This flow mainly eases the generation of the infrastructure of the backplane server and remote clients' gateways needed to connect the heterogeneous components of an ECU.

- VSIBuild (Digital Twin Builder): Developing a user's own digital twin configuration requires a relatively high level of knowledge and expertise in many areas and a considerable amount of development time. VSIBuild provides an automated flow where users can create the skeleton of their desired digital twin configuration easily. Users can invoke simply as: vsiBuild [options] [-f <commandFile>]
- VSISim (Digital Twin Simulator): The simulator provides an interface to the user to be able to control the simulation time and display information about the digital twin during the simulation. An X-Term window pops up for each of the DT components, and a command prompt interface for "VSI Simulation Control" that provides full control over the time advancement and can display some useful information about the system. Users can invoke simply as: vsiSim <digital_twin_name.dt> [options]

In summary, the user design entry is the DTDL to describe the architecture of the digital twin and a builder that parses the language and generates the infrastructure needed to connect the different models and tools, then again comes the role of the user to insert his/her System Under Test (SUT) or Design Under Test (DUT) and kick-off the operation of entire compositional simulation using the digital twin simulator managing the simulation time with a unified timing engine while allowing the visualization of the Gateways' interconnect signals.

III. CASE STUDY

Our Case Study is structured around Simcenter Prescan (a physics-based simulation platform that enables the development of virtual verification for Advanced Driver Assistance Systems (ADAS) and autonomous vehicles) [9] connected to an ECU over the interconnect framework using proper gateways at each design phase. An RCarM3 Virtual Platform (VP), proFPGA CS board (pre-silicon w/ICE interface or Co-model Channel), and NVIDIA Jetson TX2 board (post-silicon) were selected for real-time application execution. Each configuration integrates with the

same simulation scenario, ensuring consistency across testing environments. Different Software platforms for the same application (Adaptive headlight) were used: Ubuntu Linux for the RCAR M3 Virtual Platform, Petalinux/Zynq MPSoC for the ProFPGA Cs with ICE interface, bare-metal binary for the RISC-V SoC running on ProFPGA Cs with Co-model channel interface, and Ubuntu Linux for the NVIDIA Jetson TX2.

The "Adaptive headlight" application receives the steering angle from the Prescan sensor, then calculates the headlight angle and sends it over the Ethernet backplane (Simulated or Physical) back to the scenario simulator. The basic operation of this client is as follows: Wait to receive an Ethernet packet from the sensor containing the steering angle, calculate the headlight angle from the steering angle value, and send the new headlight angle value to the scenario simulator to change the headlight angle of the car.

The interconnect framework facilitates seamless communication between components via Ethernet protocol, while data signals such as headlight Angle and steering Angle are transmitted across configurations. Fig. 3 illustrates the transition paths between different platforms

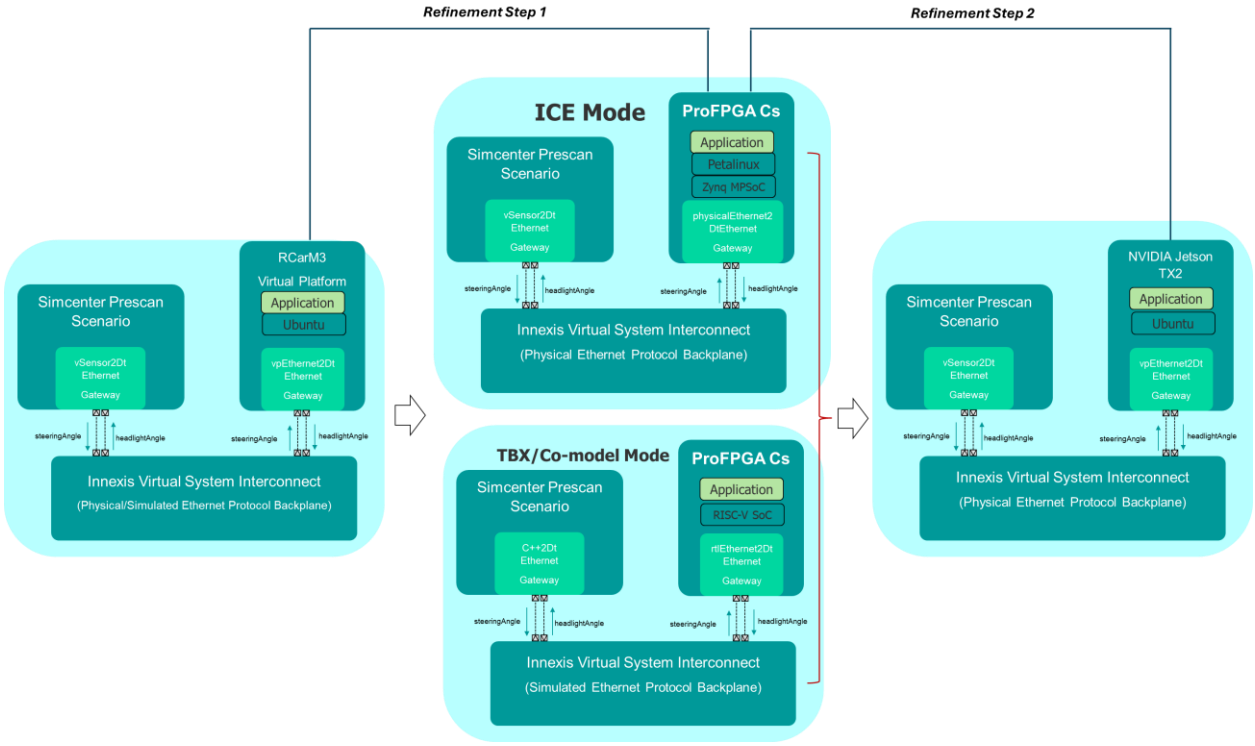

Figure 3. Transitioning between different design abstractions with digital twin simulation for an Adaptive headlight ECU

We used the DTDL depicted in Fig. 4 to define two ports with appropriate gateway types. This included configuring the ports by specifying their signals and directions and setting up data exchange based on the protocol used, which could be the physical or Simulated Ethernet protocol.

Note that this is easily achieved by changing two lines of configuration (e.g., from Virtual Platform to Physical Board on a Physical Ethernet backplane or from Virtual Platform to RTL implementation on a Simulated Ethernet backplane). The setup is considered a plug-and-play approach, with just changing the gateway type.

```
add component -type vSensor -vSensorType prescan -name steeringAngleSensor
add port -componentName steeringAngleSensor -gateway vSensor2DtEthernet -name
    steeringAngleSensor
add component -type VirtualPlatform -vpPath <Path to VP installation directory> -name
    rcarM3Vp
add port -name rcarM3Vp -componentName rcarM3Vp -gateway vpEthernet2DtEthernet
```

```
add component -type vSensor -vSensorType prescan -name steeringAngleSensor
add port -componentName steeringAngleSensor -gateway vSensor2DtEthernet -name
    steeringAngleSensor
add component -type Hil -name physicalBoard
add port -name physicalBoard -componentName physicalBoard -gateway
    physicalEthernet2DtEthernet
```

Figure 4. Digital Twin Description Language for digital twin configuration (snippet of code)

Incorporating a physical Ethernet gateway into digital twin simulation architecture introduces substantial advantages that enhance the fidelity and practicality of the validation environment. This integration enables real-world network behavior modeling. The physical gateway serves as a crucial bridge between the virtual and physical domains, allowing realistic data exchange and enabling the verification of complex network scenarios that would be challenging to replicate in a simulated environment. On the other hand, using a simulated environment would allow connectivity with Hardware Assisted Verification platforms with SV-DPI/Co-model channel interface support.

The proFPGA CS Petalinux, with APU (Application Processing Unit), is running at a frequency of 1.4 GHz. Fig. 5 shows the proFPGA CS hardware setup connected to the EB-FM-CS-XCVP-R1 extension board (Ethernet interface included) to allow the physical Ethernet gateway connection. The novelty of this setup, in particular, is using the latest Veloce proFPGA CS hardware [10] with Versal Interface board [11], including the Ethernet interface, which provides essential connectivity capabilities crucial for modern FPGA-based prototyping systems.

These extensions enhance the validation environment by implementing configurable ethernet interfaces that support various speeds and protocols, enabling real-time communication between FPGA prototypes and external systems, which is the Physical Ethernet Gateway in our case, in addition it can be used for other protocols using different ICE interfaces (e.g. USB interfaces) which shows the high level of flexibility provided by such a prototyping platform. Fig. 5 shows also the setup connection with NVIDIA Jetson TX2 and compiling the adaptive headlight application C application on Ubuntu OS.

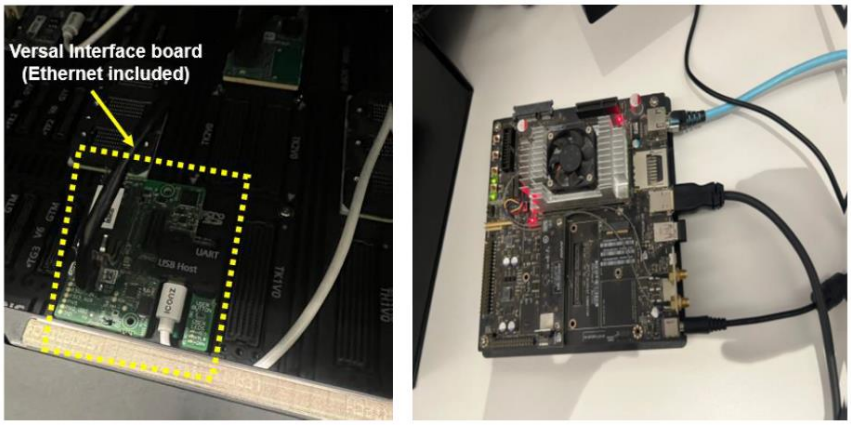

Figure 5. proFPGA CS prototyping board w/EB-FM-CS-XCVP-R1 Extension board (Ethernet interface included) and w/NVIDIA Jetson TX2

The system is fully operational, including sensor data, the Simcenter Prescan model, the C application for all configurations as outlined in Fig. 3, satisfying the criteria of having the ability to preserve the stimulus across different ECU design implementations (Virtual Platform, Pre-Silicon, and Post-Silicon). Fig. 6 shows the results of scenario simulation from Prescan and the signals scope results from MATLAB Simulink.

Figure 6. Application Use-Case (Adaptive Headlight) - Scope of Steering Angle Vs. Headlight Angle values from Simulink

A comparative analysis of the three experimental platforms was conducted as shown in Table I, showing the differences between the three platforms for a scenario that was run for 50 seconds.

TABLE I

	Virtual Platform	**Veloce proFPGA CS**	**NVIDIA Jetson TX2**
Wall clock time	2350.283 (longest time)	1291.818 (substantial improvement)	1274.58
Gateway type	Ethernet simulated	Ethernet physical	Ethernet physical
OS type	Ubuntu	Petalinux	Ubuntu

IV. Conclusion

This paper successfully demonstrated a novel methodology for enhancing automotive Electronic Control Unit (ECU) design through Digital Twin Simulation. By leveraging a consistent stimulus from a scenario simulator and ego vehicle dynamics, the study showcased the seamless transition of an adaptive headlight application from pre-silicon to post-silicon phases. This was achieved by simply exchanging gateway connections on the interconnect backplane, highlighting the reusability and scalability of the proposed framework. A key contribution of this work lies in the effective integration and utilization of FPGA prototyping Extension Boards within the ECU design methodology, proving their value in bridging the gap between early design stages and final implementation.

References

[1] H. Liu, B. Zhang, V. Wu, X. Yang, and L. Wang, “Review of digital twin in the automotive industry on products, processes, and systems,” International Journal of Automotive Manufacturing and Materials, 2025.
[2] D. Piromalis and A. Kantaros, “Digital twins in the automotive industry: The road toward physical–digital convergence,” Applied System Innovation, Jul. 2022.
[3] M. Grieves, *Origins of the Digital Twin Concept*, Technical Report, Vol. 8, Florida Institute of Technology, 2016. doi: 10.13140/RG.2.2.26367.61609.
[4] F. Tao et al., "Digital Twin-driven product design framework," Int. J. Prod. Res., 2019.
[5] Siemens Digital Industries Software. *Emulation on Veloce*. Available at: https://eda.sw.siemens.com/en-US/ic/hav/
[6] Tasneem Awaad, Mohamed Ellethy, and Mohamed Abdelsalam. Exploring Software-Defined Vehicles through Digital Twin Simulation with Extensible Prototyping FPGA: A Tool Perspective, DVCon Japan, Tokyo, 29 August
[7] Abdelsalam, M., and A. Forrai. "A design methodology for compositional simulation: The digital-twin interconnect framework." 23rd Driving Simulation and Virtual Reality Conference and Exhibition, Strasbourg, France, 2024.
[8] Siemens Digital Industries Software. Innexis Virtual System Interconnect, eda.sw.siemens.com/en-US/ic/hav/innexis/virtual-system-interconnect, (accessed January 1, 2025).
[9] Siemens Simcenter PreScan. Available at: https://www.plm.automation.siemens.com/global/en/products/simulation-test/active-safetysystem-simulation.html
[10] Veloce proFPGA CS: https://eda.sw.siemens.com/en-US/ic/hav/veloce-cs/profpga-cs/
[11] AMD Versal™ Adaptive SoCs https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal.html