2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

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

SIEMENS

CAIRO UNIVRSITY

Outlines

- Motivation & Problem Statement
- Background
  - Digital twin Fabric: Innexis VSI
  - FPGA Prototyping: Veloce proFPGA CS
- Proposed ECU Design Workflow
- Case study & Results
- Conclusion

Motivation & Problem Statement

Electronic Control Unit (ECU) Design

Accelerate development and Improve Reliability

Seamless integration between pre-silicon and post-silicon design phases

Virtual SoC platform → FPGA prototyping platform → Edge device

Outlines

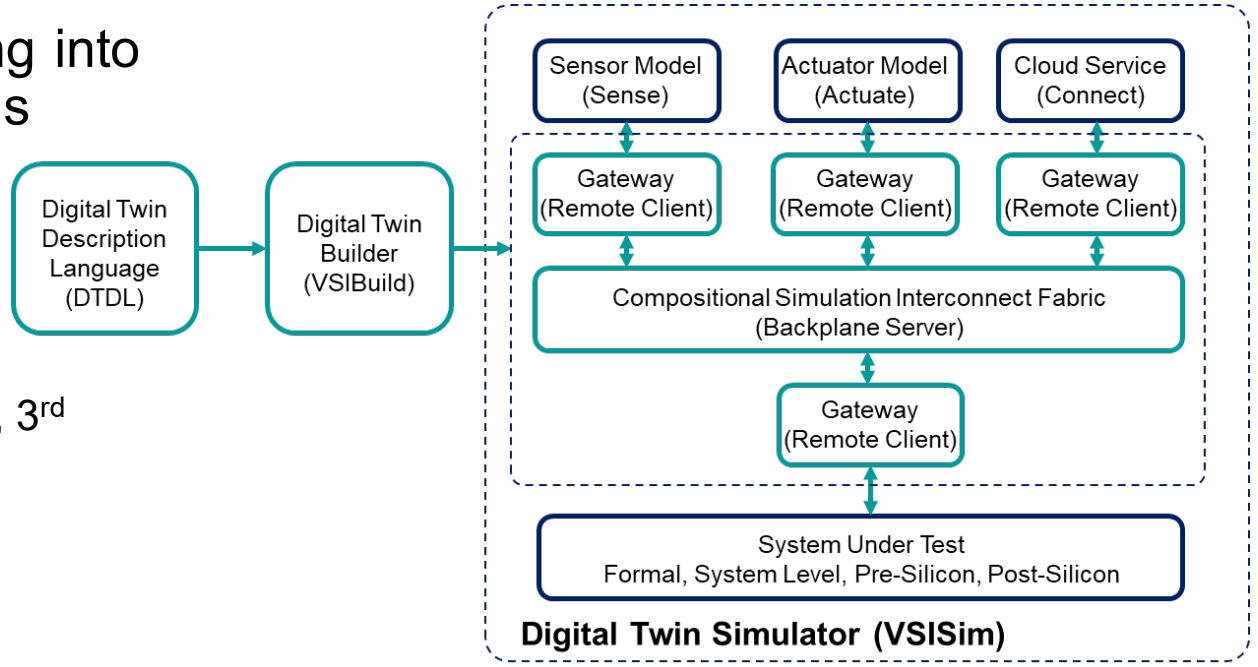
- Motivation & Problem Statement
- Background
  - Digital twin Fabric: Innexis VSI
  - FPGA Prototyping: Veloce proFPGA CS
- Proposed ECU Design Workflow
- Case study & Results
- Conclusion

Digital Twin Fabric

- Optimized interconnect enabling heterogeneous clients' connections to create digital twins
- Conformance with well-established electronic design automation standards: SystemC TLM 2.0, JModelica FMI 2.0, and Inter-Process Communication (IPC) connections
- Connections of different client types (C/C++, SystemC, Python), Virtual SoC Platforms, SoC RTL and HW boards, sensor/scenario simulators, mechatronic systems simulators, cloud services and dashboard SW
  - 3rd party tools, open-source tools interface using C/C++, Python SDK

Innexis Virtual System Interconnect (IVSI)

- Each digital twin component port is connected to IVSI backplane server through a “gateway”
- A gateway executes data conversion and routing into VSI simulated & physical communication protocols
  - Gateways are categorized
    - Language (C++/SystemC/Python/ROS)
    - Pre-Silicon (VPs, RTL)
    - Post-Silicon (HiL)
    - Specialized (Sensors, Actuators, Cloud Services, 3rd party Tools)
  - Communication Protocols
    - CAN-FD, Ethernet, AXI, UART, GPIO, SPI



Outlines

- Motivation & Problem Statement
- Background
  - Digital twin Fabric: Innexis VSI
  - FPGA Prototyping: Veloce proFPGA CS
- Proposed ECU Design Workflow
- Case study & Results
- Conclusion

Veloce proFPGA CS

- Veloce proFPGA CS is high performance software prototyping platform optimized for software verification and hardware/software system integration validation
- This is a highly flexible and affordable platform from single FPGA to multi-blade for very large designs
  - Versal AMD FPGA (VP1902)
  - Veloce proFPGA CS desktop system can offer performance in the range of 100 of Mhz

ICE Vs. TBX mode

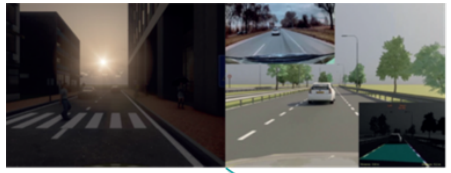
- ICE (Interface Communication Extension)
  - "ICE" mode, in the context of FPGAs, refers to a methodology where the debugging and verification logic is largely embedded within the FPGA itself
  - In ICE mode, you use actual hardware representing the required protocol (Ethernet extension board in our case)
  - Communication between Testbench and DUT on FPGA is using Data Cables

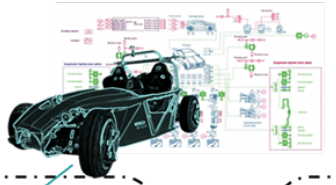
- TBX (co-model channel/Testbench Acceleration)
  - "TBX" refers to a methodology where the test bench logic resides primarily outside the FPGA. *“Think of it as an extension of your simulation environment but interacting with the physical FPGA”*
  - In TBX mode, the FPGA is treated more like a "device under test" (DUT) that communicates with an external host (like a PC) or a dedicated test system through a co-model channel
  - Communication between Testbench and DUT is using SV-DPI

Outlines

- Motivation & Problem Statement
- Background
  - Digital twin Fabric: Innexis VSI
  - FPGA Prototyping: Veloce proFPGA CS
- Proposed ECU Design Workflow
- Case study & Results
- Conclusion

ECU Design Workflow





CARLA AlpaSim, …

Co-Sim, FMUs

Prescan | Amesim | IVSI | *C/ROS Gateway* | SDV Application (C/ROS) | VPC

Software-in-the-Loop

Prescan | Amesim | IVSI | *VP Gateway* | Innexis ANA vSoC Platform | SOAFEE EWAOL Linux | SDV Application (Container Image) | VPC

Processor-in-the-Loop (Pre-Silicon)

On Cloud Verification w/Digital Twin Simulation

Prescan | Amesim | IVSI | *RTL Gateway* | SoC Platform (HDL) | Compiled Embedded SW | HW Assisted Verification Platform

ProFPGA Cs

Processor-in-the-Loop (Pre-Silicon)

On Edge Verification w/Digital Twin Simulation

OTA Update

Prescan | Amesim | IVSI | *HiL Gateway* | Compiled Embedded SW | ECU

Eval Boards

HW Integration (Post-Silicon)

On Edge Deployment w/Digital Twin Simulation

Outlines

- Motivation & Problem Statement
- Background
  - Digital twin Fabric: Innexis VSI
  - FPGA Prototyping: Veloce proFPGA CS
- Proposed ECU Design Workflow
- Case study & Results
- Conclusion

Case study

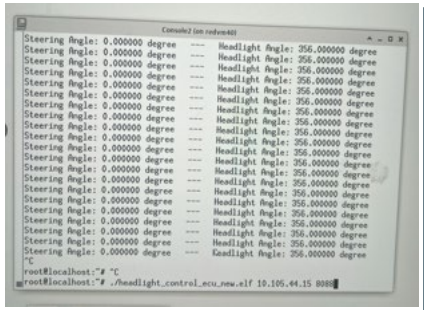
- Case Study is structured around Simcenter Prescan (a physics-based simulation platform that enables the development of virtual verification for Advanced Driver Assistance Systems (ADAS) and autonomous vehicles)
- Prescan connected to an ECU over the interconnect framework using proper gateways at each design phase

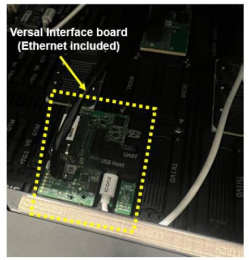
Case study

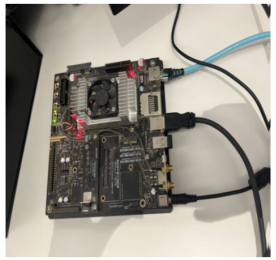
- Each configuration integrates with the same simulation scenario, ensuring consistency across testing environments
- Different Software platforms for the same application (Adaptive headlight) were used:
  - Ubuntu Linux for the RCAR M3 Virtual Platform
  - Petalinux/Zynq MPSoC for the ProFPGA Cs with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

Refinement Step 1

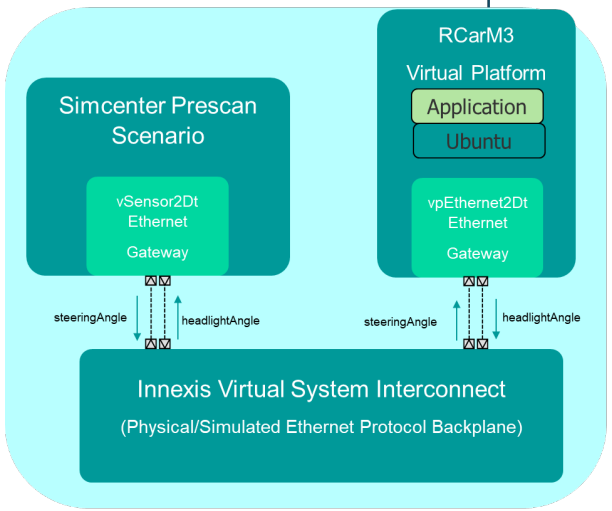
Refinement Step 2



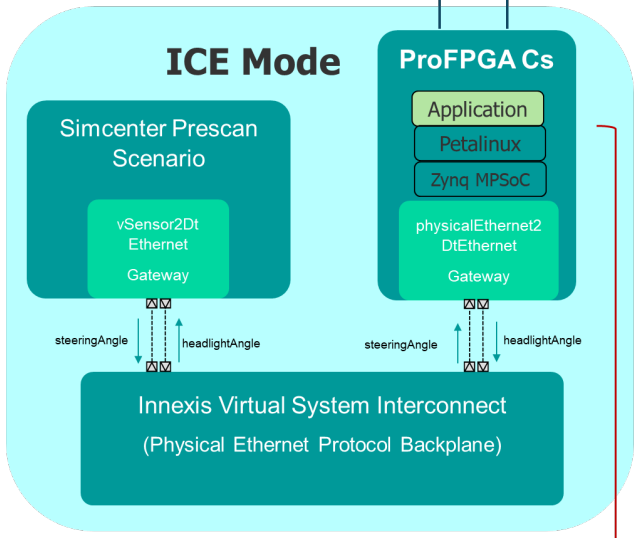


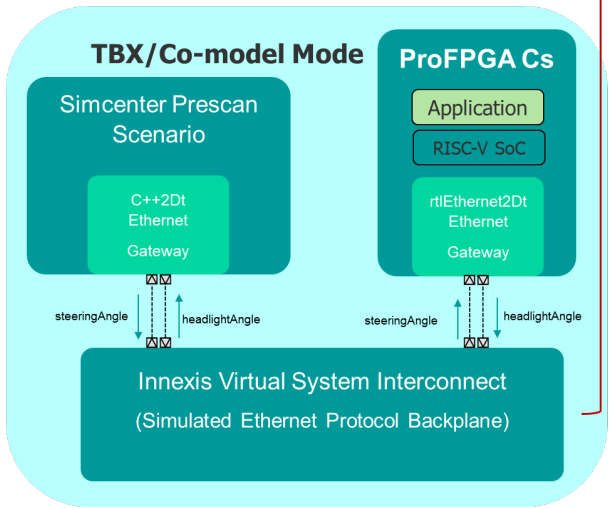


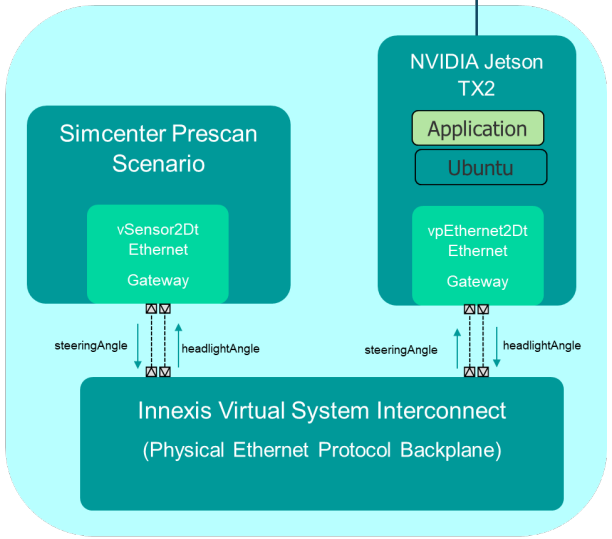
NVIDIA Jetson TX2
as edge device



Virtual platform
R-Car M3







Case study

- DTDL defines two ports with appropriate gateway types.

- 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).

```
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
```

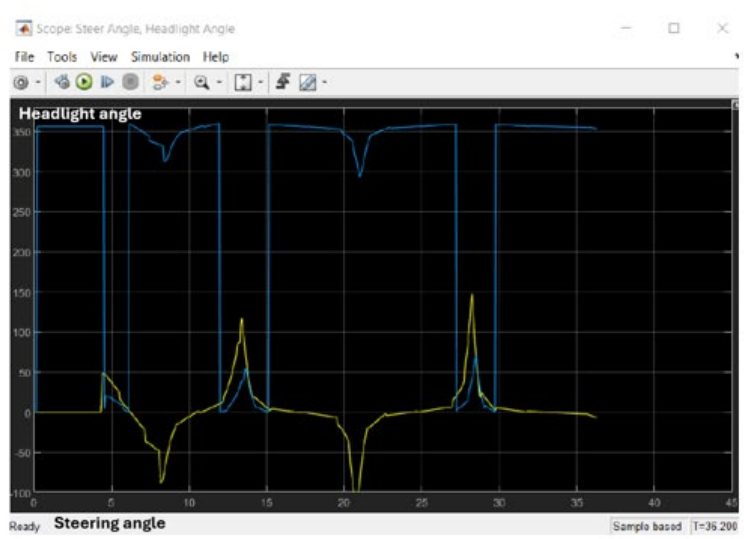
```
add component -type vSensor -vSensorType prescan -name steeringAngleSensor
add port -componentName steeringAngleSensor -gateway vSensor2DtEthernet -name
    steeringAngleSensor
add component -type Hil -name physicalBoard
add port -name physicalBoard -componentName physicalBoard -gateway
    physicalEthernet2DtEthernet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

The setup is considered a plug-and-play approach, with just changing the gateway type

Results

- Veloce proFPGA CS wall clock time is almost half the virtual platform wall clock time
- Prototyping platform wall clock time is approaching the edge device wall clock time
- Physical ethernet gateway in both proFPGA CS and NVIDIA Jetson cases is mimicking real time network effects

Application Use-Case (Adaptive Headlight) Simulink



Scope of Steering Angle Vs. Headlight Angle values from Simulink

Point of comparison/ Platform	Virtual Platform	Veloce proFPGA CS	NVIDIA Jetson TX2
Wall clock time if VSI Simulation time is 50seconds	2350.283 (longest time)	1291.818 (substantial improvement)	1274.58
Gateway type	Ethernet simulated	Ethernet physical	Ethernet physical
OS type	Ubuntu	Petalinux	Ubuntu

Outlines

- Motivation & Problem Statement
- Background
  - Digital twin Fabric: Innexis VSI
  - FPGA Prototyping: Veloce proFPGA CS
- Proposed ECU Design Workflow
- Case study & Results
- Conclusion

Conclusion

- Case study of an adaptive headlight ECU from pre-silicon phase to post-silicon phase in the context of digital twin simulation
- Workflow emphasizes on re-using the same scenario/sensor data as stimulus for pre-silicon and post-silicon phases by simply exchanging gateway connections on the interconnect framework
- Added more flexibility to hardware design and verification beyond Virtual Platforms using FPGA prototyping