

Research on Interface Methods between Virtual IP Models and Simulators in SIL Environments

Jaekyung kim¹⁾ Kyungnam Kim¹⁾ Subin Jung¹⁾ Haram Park¹⁾

*1) Hundai-Autoever
12, Teheran-ro 113-gil, Gangnam-gu, Seoul, Republic of Korea*

KEY WORDS: software and its underlying technologies, HILS, SILS, virtualization, Virtual IP Models, Interface Methods, Communication Performance, Latency Analysis, Resource Utilization(E3)

As the complexity of automotive electrical/electronic (E/E) systems increases, the importance of Software-in-the-Loop (SIL) environments has become increasingly prominent. In SIL environments, data exchange between simulators and virtual IP (Intellectual Property) models occurs repeatedly at every control cycle, and the interface method used directly impacts the system's real-time performance, resource efficiency, and scalability.

This research comprehensively evaluates three widely-used interface methods in industrial practice—FMI-based FMU, Shared Memory, and TCP Socket—and provides practical guidelines to help engineers select the optimal interface for their specific project requirements.

The FMU approach follows the international standard (FMI) established by the Modelica Association, and the simulator communicates sequentially with FMU through FMI APIs. Although no separate synchronization mechanism is required, data serialization/deserialization processes are involved.

The Shared Memory approach enables direct data exchange between simulators and IP models through a memory region allocated by the OS kernel. It is efficient due to the absence of API overhead, but context switching costs for thread synchronization are incurred.

The TCP Socket approach involves simulators (clients) and IP models (servers) maintaining independent memory spaces and transmitting binary frames through TCP sockets. It follows a request-response protocol and is well-suited for distributed environments.

Experiments were conducted on a virtual DC motor model (1 input, 3 outputs), revealing clear performance differences among the three methods. In round-trip latency measurements, FMU achieved an average of 4.99 μ s (lowest), Shared Memory 33.38 μ s, and TCP Socket 73.75 μ s (highest). These differences stem from the fundamental operating principles and system overhead inherent to each approach.

In terms of resource efficiency, FMU operates most efficiently, Shared Memory incurs context switching costs, and TCP Socket exhibits the highest resource consumption due to kernel mode transitions. Regarding scalability as signal counts increase, Shared Memory demonstrates the best scalability, TCP Socket shows adequate performance, while FMU exhibits linear overhead growth with increasing signal counts.

This research provides a comprehensive evaluation of three interface mechanisms for automotive SIL environments. The FMU approach achieves the lowest average round-trip latency (4.99 μ s) and superior resource efficiency, making it optimal for control logic verification with strict real-time requirements. Therefore, it is recommended when the simulator environment supports FMI standards and model reusability is critical.

The Shared Memory approach provides outstanding scalability with increasing signal counts, making it suitable for complex multi-model environments. It is an effective choice when integrating complex simulation models requiring large-scale signal exchange in a single-PC environment.

The TCP Socket approach offers distributed simulation capabilities, advantageous when considering future Hardware-in-the-Loop (HIL) and cloud-based simulation extensions. It is ideal for connecting simulators and IP models running on multiple PCs or in distributed environments that allow longer control cycles.

In conclusion, engineers can select the optimal interface mechanism based on project priorities (real-time performance vs. scalability vs. distributed deployment). The following comparison Table.1 summarizes the key characteristics of the three approaches to support decision-making.

Item	FMU	Shared Memory	TCP Socket
Mean Latency	4.99 μ s	33.38 μ s	73.75 μ s
Resource Efficiency	High (No OS intervention)	Medium (Two context switches)	Low (Kernel call)
Signal Scalability	Low (Linear VR lookup overhead)	High (Single call)	Medium~High
Recommended Use Cases	Strict real-time requirements, FMI-compliant simulators, Model reusability is critical	Large-scale signal exchange, Complex multi-model simulation, Single-PC environment	Distributed environments (Multi-PC), Extended control cycles, HIL/cloud expansion consideration

Table.1 Comprehensive Interface Method Comparison