Spirent™



WHITE PAPER

Understanding Latency for Hardware-in-the-Loop Testing

WHITE PAPER

Understanding latency for hardware-in-the-loop testing

Modern vehicles, whether autonomous or assisted, rely heavily on positioning, navigation and timing (PNT) information. Whether this is to deliver an enhanced user experience, to ensure safety and efficiency on the roads, or to deliver a weapon system or payload to the right place at the right time, the precision and reliability of PNT processing must be tested extensively.

GNSS technology has risen to meet this performance challenge, with hardware and services options ranging from multi-constellation and multi-frequency receivers, to terrestrial and space-based augmentation systems, to complex array antennas. One of the greatest tools for developers of PNT-dependent systems, though, is inclusion of additional PNT sensors – usually working in concert with GNSS. The need for multiple sensors to work seamlessly together is not only increasing the importance of testing, but is bringing greater challenges to the requisite testing processes.

There was a time when receiver developers could focus purely on the performance of the GNSS receiver itself. Testing was a matter of ensuring the receiver met certain criteria: solution accuracy, availability, continuity and integrity. All of these criteria are still fundamentally important – the receiver must be able to acquire and process GNSS signals in the way the customer expects, and receiver manufacturers must be able to produce documented evidence of their products' reliability in these key areas. However, in a world where PNT information is routinely safety- and mission-critical, and where the GNSS receiver is just one part of a complex system of sensors, actuators, software applications, and algorithms, developers need to ensure that an integrated product continues to meet standalone performance parameters.

One of the key methodologies available for vehicle, system, and sub-system developers and/or integrators is hardware-in-the-loop (HIL) testing, but this comes with its own set of challenges. A HIL system brings together multiple systems, and often multiple test instruments. Each of these test instruments takes time to process inputs and issue corresponding commands to stimulate or emulate signals or outputs. This time is referred to as latency, and is a key issue for some HIL configurations. In this paper, we'll look at what latency is, how it should be measured, and the impacts of increased levels of latency on a test setup.

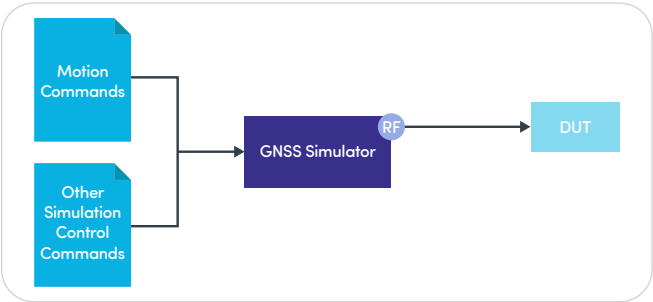
Test configurations

To understand the role of HIL testing and the importance of latency, it is first necessary to lay out three core test configurations. There are variations of each, but they form the options for lab-based testing of PNT equipment.

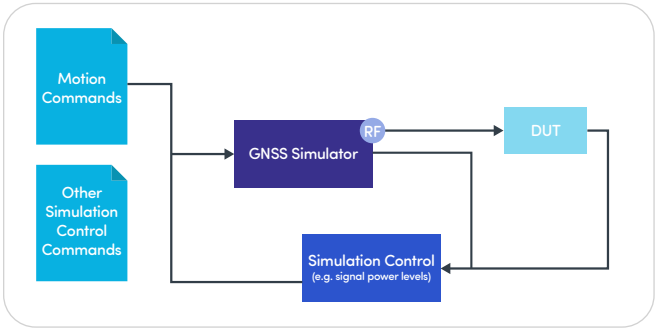
Open-loop configurations

In some open-loop configurations, all commands (including motion, signal control etc.) are known in advance of the scenario run. The simulator does not expect real-time feedback from remote systems that would change the predefined and known input prior to simulation iterations– essentially meaning data flows in one direction, and all characteristics of the test are as defined prior to the run. In this case, the remote system commands can be timestamped and buffered well in advance of each simulation iteration, enabling the simulator to compensate for any transmission and data latency. Barring some major error, system latency should always be zero in this situation.

For instance, if a simulator has an update rate of 1 kHz and has a known latency of 4 ms, it can process the motion commands 4 iterations ahead of schedule – meaning a latency of 0 ms is seen at signal output.

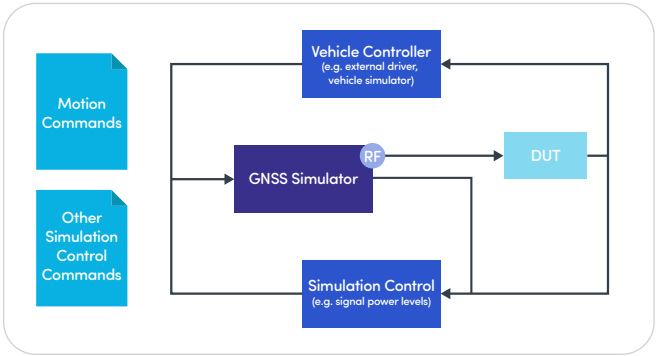


In another type of open-loop configuration the motion is known in advance, but some control commands are not. Typically, some parameters of the scenario (other than motion) will change during the run, depending on the device-under-test (DUT) response and feedback to the simulator. This could be changing the signal power levels of the simulator, for instance, based on a C/N0 threshold measured at the receiver. In this configuration, transmission and data latencies will influence the time at which issued commands are seen at RF, though this would not be considered critical, as the pre-defined motion is not changed and the effects would still be seen at the DUT.



Closed-loop configurations

In a closed-loop configuration, neither the motion nor the control commands are known in advance. The aim is typically to incorporate feedback from the DUT into the test. The remote system generates its commands responding to external inputs which cannot be known in advance, such as manoeuvres that a pilot performs using the joystick of an aircraft. In this configuration transmission and data latencies will influence the time that the issued commands are applied to the simulator's RF signal.



Defining latency

The main task of a GNSS simulator is to convert digital inputs into analogue RF outputs. The realism of the simulation is influenced by the fidelity of the models used, the capabilities of the simulator hardware and firmware to solve those models, and, in some instances, the total latency of the system (the existing delay from the input to the output stage).

Simulation latency becomes a critically important parameter in HIL environments. Realism in HIL requires a high level of synchronisation between all the hardware and simulators involved to maintain data and time coherence. Any inconsistency due to delays in the transmission or processing time of the different simulators involved will lead to uncertainty in the test results.

From the perspective of the simulator, in a HIL environment there are four key actions, with their timestamps, that help to determine the total latency:

- The time at which the message is sent from an external node or simulator
- The time at which the message is received and processed
- The time at which the model is updated in the software
- The time at which the corresponding RF or analogue signal is output

The intervals between these events determine the different latency values of the simulation (figure 1).

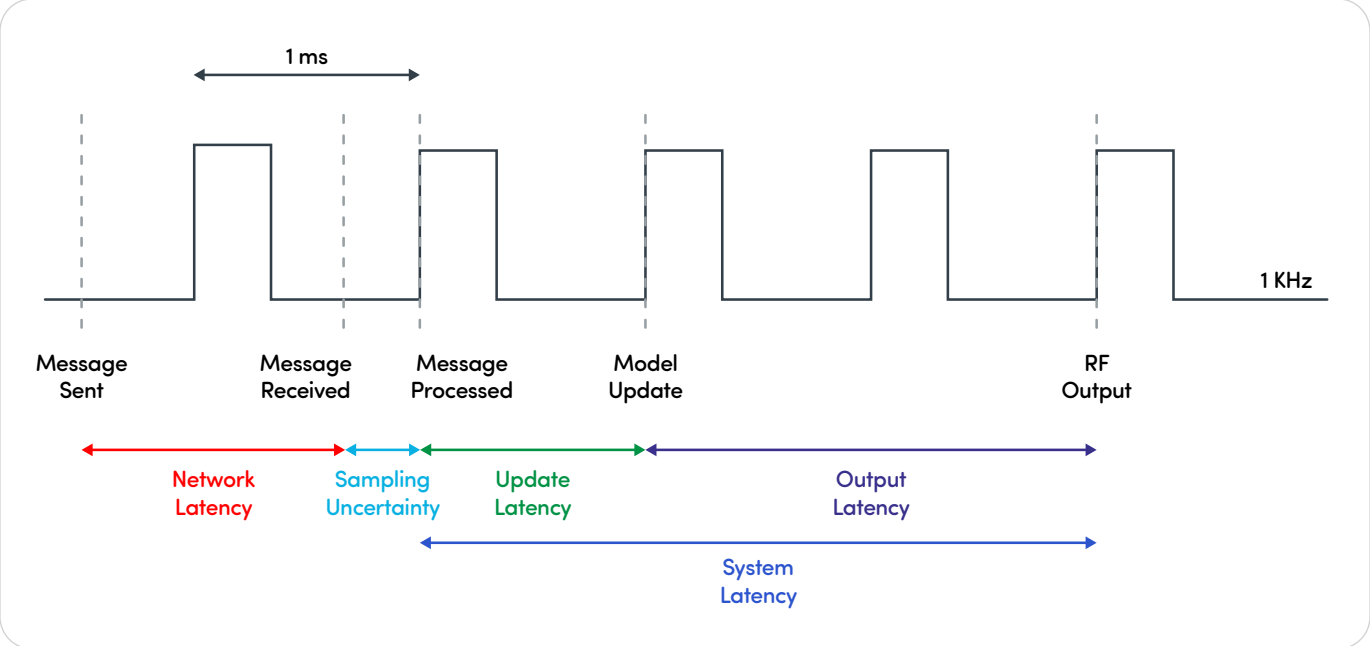


Figure 1. Representative HIL latency for a GNSS simulator with an update rate of 1 kHz

Network Latency

This is the delay between the transmission of a message from an external source – often a HIL controller – and that message being received by the simulator in question. This is entirely dependent on the network configuration, and is not impacted by the specifications of the simulator. For Ethernet interfaces, you can easily send UDP motion data with just tens of microseconds latency.

Sampling Uncertainty

GNSS simulators use discrete solvers to obtain the solution for every state of the models. Digital inputs will necessarily get sampled in between simulation cycles and, consequently, they are not processed until the next simulation step. Consistency of the sampling uncertainty, coupled with the ability to pre-process digital inputs, can help to reduce overall latency by aligning the digital input with the end of every simulation cycle. Sampling uncertainty measurements depend on the simulation iteration rate (SIR) of the simulator, and how it compares to the SIR of the remote system. The simulator will check the motion buffer each cycle (every 1 ms for a 1 kHz simulator). Provided the remote system has a SIR equal to (or faster than) the simulator (and the data meets the required setup and hold time), sampling latency will be <1 simulation cycle. If the remote system is 1 kHz, but the simulator is 100 Hz, then the buffer will be checked every 10 ms, giving a sampling latency of <10 ms, and the same is true of the reverse.

Update Latency

Once a message is received by the simulation software, it will take some time to update the model. This update latency is always expressed and realised as an integer multiple of the simulation cycle – ranging from a single simulation cycle to several. In the Spirent architecture, if the message is within the buffer ready to be used then the update latency is within the same SIR period. For example, if motion is received just before a 1 ms epoch then the SimGEN high rate thread will pick that motion up, process it and download the new numerically controlled oscillator (NCO) numbers or data to the firmware - all within the same millisecond.

Output Latency

This refers to the time delay between the model update and the RF or analogue signal output. The latency measurement here is highly dependent on the simulation hardware and firmware. The time taken to update models, generate the signals, and realise them at RF varies hugely depending on the processing capabilities of the simulator, and the level of calculations it has to make at each cycle. For this reason, field programmed gate arrays (FPGAs) will usually outperform generic hardware by in an increasing margin as the complexity of the simulation increases.

Extrapolation and zero-effective latency

One concept that can come up when considering hardware-in-the-loop test equipment is "zero-effective latency". In order to understand this concept, it can be easier to think of latency as uncertainty.

In a remote motion closed loop situation, the incoming trajectory data is already in use by the remote simulator and hence is inherently late. If a simulator operates at 1 kHz SIR and has system latency of 4 ms, then it will always be predicting (extrapolating) 4 simulation cycles ahead of the actual data it has. This is the measure of the uncertainty of the test. To what extent that uncertainty translates to error will depend on two factors:

1. The unpredictability/dynamics of the motion data
2. The quality of the extrapolation algorithms the simulator employs

In the case of a vehicle moving in a straight line at a constant speed, this extrapolation can (or should) be correct. In effect, the simulator will be outputting the correct motion data at the correct time. This is referred to as zero-effective latency. This is not a consistent feature – instead it can be viewed as occasions where the simulator guesses precisely where the next motion step will occur.

For most setups, zero-effective latency is not a helpful concept. Assuming that the simulator does not predict each step with complete precision - and depending on the level of error between the extrapolated prediction and the required value - then the value of latency would be inconsistent throughout the test. The only instance where zero-effective latency can be consistent and can be useful is in scenarios with no changes in speed or attitude. If that is the case, then an important question should be asked: does the test require real-time remote motion at all?

Does latency matter to my testing?

Whether or not latency is important to your testing depends on the test configuration - specifically, whether or not the test truly needs to use real-time remote motion that is not known in advance.

Yes

For testing where the motion depends on feedback from one or more sensors, or where human/autonomous control over an instrument such as a joystick or steering wheel is imperative, then latency is critically important. The greater the latency, the greater the uncertainty in the test. As the dynamics increase, the potential error inserted into the test by higher latency increases exponentially.

No

The majority of GNSS testing does not require real-time remote motion. Where using pre-defined motion and trajectories is viable, this should be preferred. However low the latency (and however high the performance) of the test equipment, using real-time motion adds uncertainty to test results. Using it when it is not strictly necessary should be avoided. Where real-time remote motion is not used, latency should not be a consideration.

Configuring a testbed to minimise latency

As outlined in the previous section, there are two parts of the testbed to consider when thinking about latency. Network latency and sampling uncertainty are strictly dependent on the network configuration, and not on the simulator providing the signals. When configuring the testbench, the choice of communications protocols, quality of cabling, and the performance of external units – such as the Ethernet switch – are important tools in the minimising of overall latency.

Having said this, the simulator can aid in minimising sampling uncertainty. Spirent's GSS9000 and GSS7000 GNSS simulation systems provide an external 1PPS (one pulse per second) signal input/output aligned with the simulation update rate. Using this 1PPS external IO, users can synchronise the internal simulation cycles with other systems and simulators, reducing both uncertainty and latency in sampling.

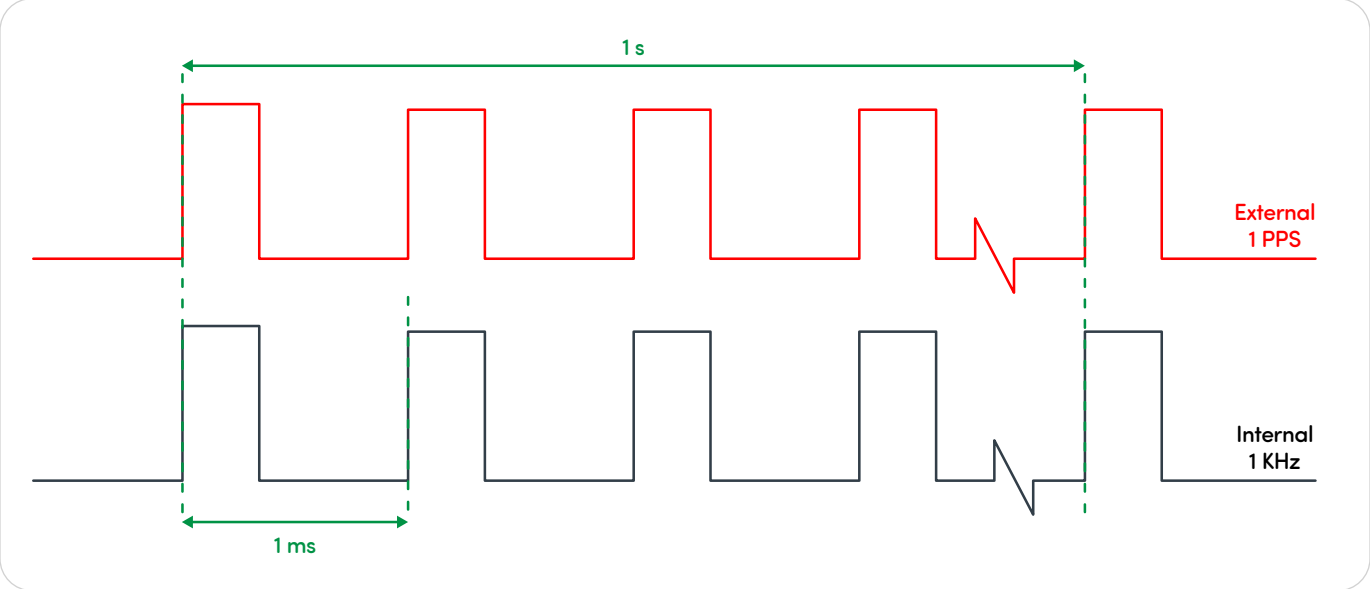


Figure 2. Sampling alignment using external 1PPS IO

Choosing a simulator

Spirent GNSS simulators run a real-time software-based trajectory prediction algorithm that help reduce the effect of network and sampling latency. Ideally, trajectory information should contain data that has no gaps, and has consistent linear and angular kinematics in the appropriate coordinate frame. Assuming the incoming data is self-consistent and predictable from epoch to epoch, Spirent's PosApp engine makes an extremely good approximation at extrapolating trajectory forward, hence effectively eliminating the effect of delay due to sampling uncertainty.

Only under extreme, unexpected dynamics, a small delay can be observed due to messages arriving late for the next update cycle. In this case, a delay of up to one simulation cycle will be seen. The selection of a higher update rate in the simulator helps to minimise this effect.

System latency – which can be defined as the sum of update and output latency – is wholly dependent on the specifications of the simulator. This value must be known in advance to ensure the test set-up meets requirements. For the same reason, the consistency of the measurement, or run-to-run bias, should also be known – a best-case latency of 10 ms is not viable when the worst case from the same system is 200 ms, for example.

In Spirent simulators, system latency is highly consistent, and is expressed as an integer of the simulation iteration rate (SIR) and hardware update rate (HUR) of the system in use. An example to highlight this is the variable SIR and HUR values of the GSS9000. When operated at 2 kHz the GSS9000 has an SIR and an HUR of 0.5 ms. In this configuration, the maximum system latency is <2 ms. When the GSS9000 is operated at 1 kHz it has an SIR and an HUR of 1 ms. In this configuration, the maximum system latency is <4 ms (see Figure 3).

Latency that is at this level and is highly consistent ensures that the vehicle trajectory is faithfully recreated, and the RF signals are generated accordingly.

It's important to note that update rates and latency are not the same across all systems. Spirent simulators operating at 1 kHz will always have system latency of <4ms, but another 1 kHz simulator could produce hugely different values. System latency depends on a number of factors, including the type of hardware used and the quality of cabling and connections.

Spirent Simulator	SIR	Frequency	HUR	Maximum System Latency
GSS9000	0.5 ms	2 kHz	0.5 ms	<2 ms
GSS7000 & GSS9000	1 ms	1 kHz	1 ms	<4 ms
GSS7000 & GSS9000	4 ms	250 Hz	4 ms	<16 ms
GSS7000 & GSS9000	10 ms	100 Hz	10 ms	<40 ms
GSS7000 & GSS9000	100 ms	10 Hz	10 ms	<220 ms

Figure 3. Configurations of Spirent GNSS simulators

The importance of latency in high dynamic application testing

The AIM-120 AMRAAM, the M982 Excalibur and the Lockheed Martin F-35a Lightning II are 3 air vehicles very likely to be subjected to high-dynamic manoeuvres during missions. All of them rely on GPS and INS, among other sensors, to deliver a reliable and precise PVT (position, velocity, time) solution.

- The AIM-120 is a medium-range air-to-air missile with a GPS+INS unit for navigation beyond line of sight.
- Excalibur is an extended-range artillery shell equipped with a GPS+INS unit. The trajectory described by this vehicle not only implies high values of velocity (up to 2,700 ft/s), acceleration and jerk, but also high rotational kinematics as it spins. When it's fired, fins are folded. These are strategically deployed to boost range and allow retargeting. This hybrid approach allows the projectile to hit its target precisely and consistently. However, the transition implies a change of the aerodynamics properties of the vehicle that involves some challenges from a guidance, navigation and control (GNC) perspective, requiring accurate PNT data for the fix and updates that happen before the fins are fully deployed.
- The F35a is a multirole combat aircraft that is intended to perform both air superiority and strike missions. It does it with a 7-element CRPA antenna onboard and a complicated arrangement of GPS+INS equipment. There is also an aviation compartment that runs all the sensor fusion algorithms.

The wide variety of missions these 3 vehicles are expected to carry out requires simulation equipment that can cope with 3D trajectories, multiple mission legs, diverse vehicle orientations and an extensive catalogue of manoeuvres (including spinning). Test requirements include rapid motion, directional changes, and situation changes, which, in turn, demand low latency trajectory and RF signal simulation in order to maintain precision and realism. Not only a close alignment between the RF signals and the input trajectory is needed, but also a source of INS data coherent with the GNSS and based on the same trajectory.

In all instances, the definition of the trajectory and the overall latency play crucial roles in defining simulation realism and test uncertainty. In simulation, trajectories are discretised and become functions of time. Consequently, capturing high dynamics using fine sampling (high SIR/HUR) rather than coarse sampling (low SIR/HUR) is key to ensure the generation of realistic trajectories and minimise latency within the test set-up. It is also critical to bear in mind that the complex computation involved in these scenarios could cause some simulators made with lower quality or generic hardware to operate with much higher latency than expected.

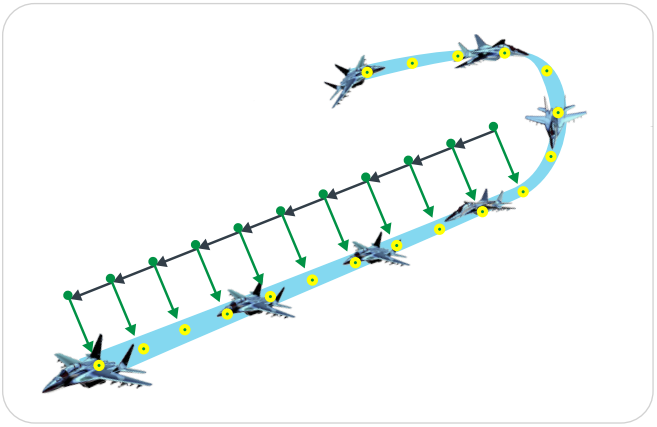


Figure 4. Trajectory sampling

The importance of latency in an autonomous vehicle HIL test rig

An autonomous vehicle HiL test rig usually involves multiple standalone simulators, a testing framework called co-simulation, and the DUT, which can be a GNSS receiver or a positioning ECU. A typical co-simulation setup includes the following simulators:

1. Driving Simulator: provides the ego vehicle information, such as its dynamics. This information is directly fed to the network simulators.
2. Traffic Simulator (optional): provides information about the other vehicles in the network (cars, trains, airplanes, animals, people, etc.). For simplicity, we are not using a traffic simulator in our example.
3. Environment Simulator: models the local environment around the DUT in 3D: the physical space with its physical properties in which the vehicles and moving objects will act.
4. Network Simulator (or sensor simulator): the point of contact between the driving simulator and the traffic and environment simulation. Consequently, it can also help with the interaction between the driving and the traffic entities. An example of a network simulator is a GNSS simulator or an inertial sensor data simulator. A lidar or radar simulator would also fall into this category.

Driving simulator

The driving simulator usually runs on real-time system hardware. In this case it acts as the primary controller of the system; this is where the user defines and runs test scenarios. The driving simulator sends two types of commands to the network simulator (PosApp, in this instance):

- Run commands: to start the GNSS simulation and provide information such as the type of vehicle or date and time.
- Motion commands: 6 DOF motion data messages, containing data obtained from the dynamic model run by the driving simulator.

Environment simulator

In this example, the environment simulator is Spirent Sim3D. It receives the position of the vehicle and the satellites from PosApp and sends back corrections to the power levels through signal control commands.

GNSS simulator

In this setup, the GNSS simulator acts as auxiliary to the driving simulator. It receives the run and control commands from the driving simulator, processes these commands, updates its internal models, and, finally, generates the RF output to the DUT. The GNSS simulator can be in an open-loop configuration, or a closed-loop configuration (as described earlier in this paper).

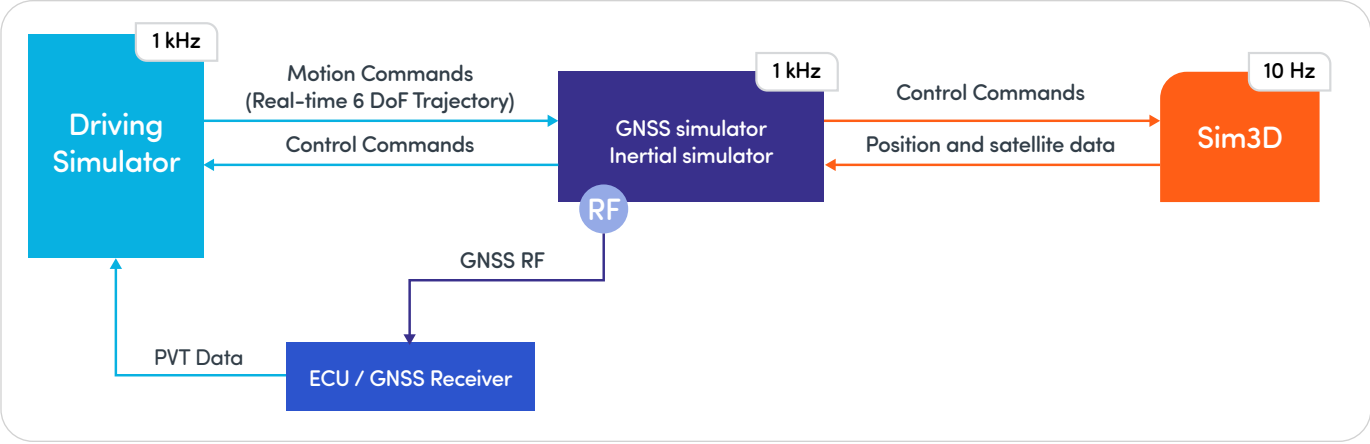


Figure 5. An example HIL configuration for an autonomous vehicle

In the configuration shown in Figure 5, the driving simulator is sending vehicle dynamics to the GNSS simulator. In this closed-loop scenario, the motion and control commands issued by the driving simulator are not known in advance by the GNSS simulator. The reasoning for this configuration is that the autonomous vehicle (represented by the driving simulator) is able to make decisions based on the feedback it receives from the GNSS receiver/ECU. For instance, assuming a scenario where a lidar simulator is included in the configuration, it could detect an obstacle in its path, and the driving simulator could decide to stop to avoid a collision. The occurrence of an unexpected or unpredictable action or command from the driving simulator is where latency becomes critical.

The above example is over-simplified. In reality, the latency of such a system is often much more complex to determine. As the required realism and the complexity of the platform increases, more simulators are required which will inevitably increase the overall latency of the HIL configuration. Each network simulator will have its own update rate and need to be synchronised with the others in the loop, meaning even combined sampling latency can become significant.

An illustrative example

Consider a GNSS simulator that brings with it 100 ms of system latency. To simplify the example, assume that network and sampling latency are zero. The vehicle described by the driving simulator is travelling down a highway at 120 km/h. There is a sudden and unexpected event – another vehicle comes to a sudden stop in the same lane as the vehicle under test – and this event is detected by the vision sensor. The driving simulator will need to take the decision to brake, and its ability to do so in a timely fashion would form an integral qualification for vehicle safety.

The importance of latency here is that the recorded position at which the vehicle made the decision to brake would be sampled 100 ms later than reality. At 120 km/h, 100 ms represents 3.3 metres – and this 3.3 metres is now the value of the test uncertainty.

About Spirent

Spirent Communications (LSE: SPT) is a global leader with deep expertise and decades of experience in testing, assurance, analytics and security, serving developers, service providers, and enterprise networks. We help bring clarity to increasingly complex technological and business challenges. Spirent's customers have made a promise to their customers to deliver superior performance. Spirent assures that those promises are fulfilled.

For more information visit: www.spirent.com

Conclusions

As discussed, latency is a critical aspect of HIL testing. In addition to being low, the inherent latency of the systems used, and of the network they are brought on to, must be known and must be consistent. An impressive best-case latency figure is of no benefit if that level of performance cannot be maintained throughout testing.

Low and consistent latency is achievable, but it demands a strong fundamental understanding of the test equipment in use, as the truth of these parameters may not always be splashed on the front page of a brochure or datasheet. With an understanding of the different areas of a loop in which latency can be introduced, it is a fundamental prerequisite that HIL developers should interrogate their systems under the full range of operating conditions in order to understand variance, tolerances, and the level of uncertainty that will be introduced at each stage. With these factors optimised, a HIL test rig can deliver on the efficiency and interoperable development that it promises.

Why Spirent?

Spirent's FPGA-based SDR is, by its nature, deterministic, which means system latency is deterministic, predictable and constant, and can be minimised. FPGAs enable low latency because they compute in parallel using logic gates instead of running sequential code in software (as in GPU-based SDRs, which have to perform many calculations in real-time and with no prior knowledge).

A crucial performance parameter for any GNSS simulation is the ability to guarantee that for any scenario, the simulator will continue to operate within its performance specification continuously and run after run. With Spirent simulators, you avoid any ambiguity or uncertainty that you might push the hardware beyond its limit. This assurance saves critical time and money in setup, during runs, and in engineer workload, avoiding the need to monitor for errors or changes, or account for unexpected buffer underruns that would diminish productivity and return on investment.

For more information on latency, hardware-in-the-loop testing, and how you can evaluate the performance of your systems, get in touch with Spirent.

Americas 1-800-SPIRENT
+1-800-774-7368 | sales@spirent.com

Europe and the Middle East
+44 (0) 1293 767979 | emeainfo@spirent.com

Asia and the Pacific
+86-10-8518-2539 | salesasia@spirent.com

© 2023 Spirent Communications, Inc. All of the company names and/or brand names and/or product names and/or logos referred to in this document, in particular the name "Spirent" and its logo device, are either registered trademarks or trademarks pending registration in accordance with relevant national laws. All rights reserved. Specifications subject to change without notice.

DWP0017 | Issue 1-02 | 09/23