8 Factors for Effectively Using Simulation at the Intelligent Edge

www.windriver.com

WNDRVR

WNDRVR

Executive Summary

The software simulation market has been growing and evolving for more than a decade. The use of simulation in embedded software development helps organizations achieve faster time-to-market, lower development costs, and enhance safety features.

Eight critical factors influence the choice and use of simulation tools. In this paper, we discuss what developers should consider when deciding on a simulation process to accelerate timelines and advance an organization's goals.

The Case for Simulation

Imagine a typical four-month test program completed in just three weeks, or the average time from code commits to defect detection reduced by 97% — from 4.5 weeks to 24 hours. Imagine shortening the time from code build to test from 55 minutes to less than one. Or a two-year test completed in three days, a 99% percent reduction.

These are all real results from customers who have leveraged today's simulation technologies and capabilities.

Simulation is becoming a crucial element of efficient and nimble software development processes. It reduces CapEx and OpEx and decouples software and hardware. It helps software developers and designers identify and resolve issues early in the development cycle while simultaneously reducing development time and lowering costs.

When you're using a virtual target in a simulation, there's no need to book a lab or equipment, since teams can use the virtual target to test anything at any time. Producing digital twins through simulation allows continuous testing, and insertion of faults for testing without risk of damaging the hardware. It supports maintenance and upgrades of designs whose service life is so long that its embedded systems have outlived their market availability — think of satellites orbiting Earth, for example.

These and many other use cases make a strong case for using simulation tools.

Testing Time Down 97%

Time from code commit to defect detection reduced from four and a half weeks to 24 hours.

Build Time Down 99%

Shortened time from code build to test from 55 minutes to less than one minute.

Equipment Testing Time Down 87%

A four-month test program now completes in three weeks.

Three Approaches to Simulation

Simulation tools take one of three different approaches, so it's important to understand how each supports your goals.

1. Hardware-centric Simulator:

This approach builds from the bottom up, creating a detailed and high-fidelity device model by simulating memory operation with a third-party processor instruction set simulator on top. Typically, these are high-fidelity simulators. One downside, however, is that they run slowly.

2. Full-System Simulator:

A full-system simulator takes a top-down approach. It can simulate multiple targets synchronously to create your own version of digital twins. It is software-centric, taking advantage of multi-threading, and it simulates the processor as well as the memory operation. The disadvantage, however, is that full-system simulations don't come down to the low-level device or cycle-accurate simulation level.

3. Open Source Simulator:

Sitting somewhere between the hardware-centric simulator and the full-system simulator is the open source simulator. Its design simulates a single-board system. It does not cover the system level as a full-system simulator would, nor does it have the high fidelity of the hardware-centric simulator. However, the source code is available for anyone to view, modify, and redistribute.

To choose the right simulation tools, developers should consider eight important factors: identifying objectives, understanding simulation needs, understanding users and collaborators, understanding ease of use, identifying the models used in simulation, considering portability and scalability, understanding basic vs. advanced features, and selecting a simulation platform based on budget and resources.

Factor 1 Objectives and Goals

The first factor to consider when starting a simulation process is your goal. Simulation can be used to improve software quality, accelerate project timelines, improve development efficiencies, and reduce CapEx and OpEx.

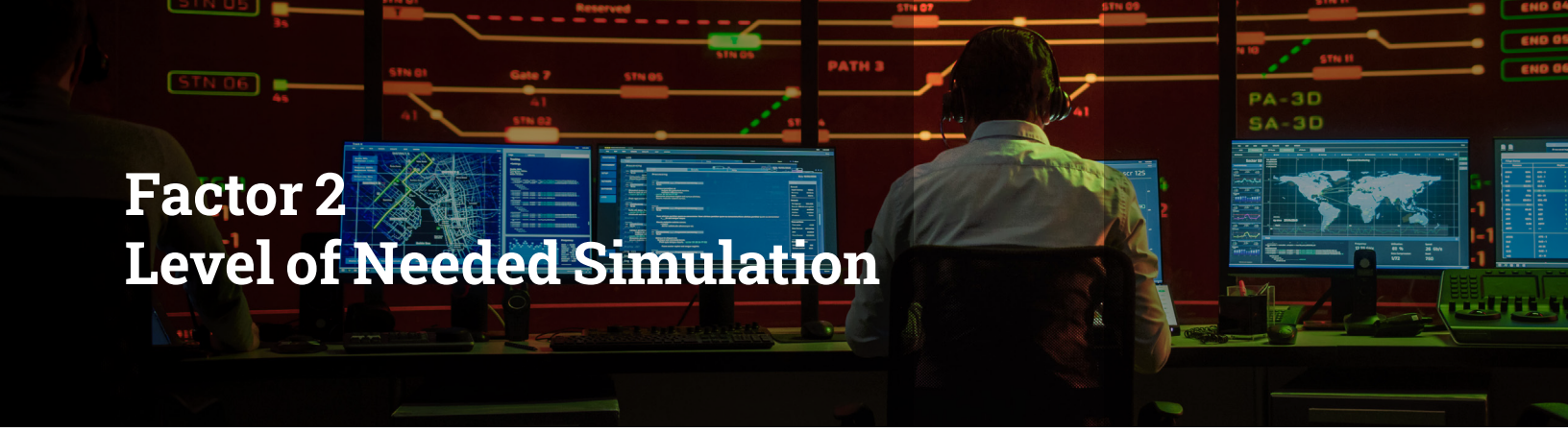
Important questions include:

- Is the timeline fixed or flexible?
- What are your costs for simulation?
- What design qualities are you hoping to improve through simulation?

For most teams, the quality of the software produced is the highest priority. Use of simulation tools can support this goal in less time and at lower cost.

USE CASES FOR SIMULATION

- Digital twins
- Enable true DevOps
- Cyber defense
- Advanced testing
- Multi-core system bring-up
- Maintain end-of-life hardware
- Threat modeling
- Safety certification acceleration



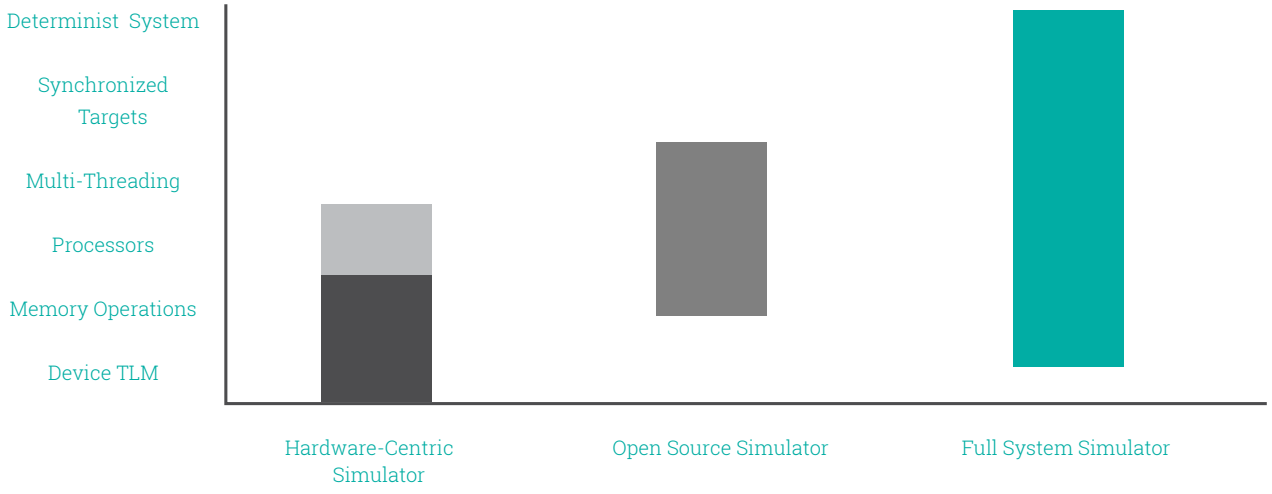
Factor 2
Level of Needed Simulation

Developers choose the level of simulation needed based on project requirements. Is it enough to simulate only at the processor level? Do you need to extend your simulation to cover custom models? Do you need to simulate the whole system, including multiple processors and devices developed by your team or by third parties?

Perhaps you will need to simulate at different levels at different times, thus needing access to every level of simulation over the course of the entire project. Ideally, your simulation tools will have the flexibility to meet your project's needs as it evolves from development to implementation and on to testing and deployment — and beyond.

> "Some designs have a service life longer than the market life of the devices that were used in that design. Simulation provides a way to support such designs long after the hardware is no longer available."
>
> — **Richard Burgin,**
> *Professional Services Organization Program Manager, Wind River*

Simulation Tool Options Available Today



Factor 3
Users and Collaborators

Next, it's important to consider the needs of the simulation users and collaborative partners.

Developers and testers need early access to the simulation to inject faults for testing, automate development and test workflows, and send the simulation to colleagues and customers.

System architects will need to understand how much of the code has been tested and will need to run that simulation for review and oversight purposes.

End customer teams will need access to the simulation to set up their use cases for their own design. They may also need to send the simulation back to the developers for support and assistance.

> "Software is teamwork. To provide ease of use, and more, is important for developers to be effective."
>
> — James Hui,
> *Product Manager for System Simulation, Wind River*

Factor 4
Ease of Use

Ease of use is another factor of utmost importance for effective development. Modern development needs include:

- **Cloud access:** Developers are cloud connected and need this common space for all users to access the simulation.

- **DevOps integration:** Large-scale development calls for the effective collaboration that can come from DevOps practices.

- **User interface:** The simulation interface should support the developer in accessing, loading, configuring, and modifying the simulation for the unique test case.

- **Repeatability:** Scripting is key for plugging simulation into automated workflows. The simulation tool needs to provide repeatability so developers can find issues using tools such as reverse-execution, replay, and checkpointing.

- **Configurability:** Simulation tools need to be configurable for on-demand setup and must support automation. Scripting is key to dynamically changing model behavior and the model itself.

- **Extensibility:** Scripting is also important for adapting workflows and extending model behavior to new use cases. A robust, usable model library helps projects run significantly faster and more cost-effectively.

> "You can take advantage of newer technology and simulation to do more in less time."
>
> — James Hui,
> *Product Manager for System Simulation, Wind River*

Factor 5
The What, Where, Who, and How of Models

A simulation is only as good as the models available. It needs to simulate the minimum devices used by the software and use cases required. When considering simulation, a key question is: Is my design available in simulation at the fidelity level I need?

If not, consider that processor architectures are complex to simulate. The benefits of finding a simulation solution with the needed processor architecture are significant in terms of easing model development complexity and reducing cost.

A simulation solution with a robust inventory of leverageable devices and reference boards is cost-effective. Device lists are commonly similar across a family of reference designs, so creating a new board model might be as simple as adding additional devices from the tool's currently available model library.

Or it may be necessary to create custom models. These can be more expensive and time-consuming to create, but they can provide more accurate simulation results.

$40+B

The projected market for simulation tools by 2030[1]

1. Globe Newswire, "Simulation Software Market Size to Surpass USD 40.5 Bn by 2030," September 23, 2022

Factor 6 Importance of Portability and Scalability in Simulation

Unique advantages of simulation include improving portability and scalability, and flexibility for developers to achieve simulation goals. Cloud connectivity and desktop simulation provide choices for where to run simulations. This helps designers, developers, and testers collaborate more efficiently and enhances the speed at which tasks can be executed.

Batch mode processing and checkpointing can also help maximize processing power and improve workflow performance. Checkpointing is another important feature of simulation tools. It enables the saving of simulation state, cloning, and running multiple tests in parallel.

Simulation tools provide unique levels of flexibility for developers to achieve goals and deterministic simulation results, providing the same result regardless of where the simulation was run and making it easy to transfer the simulation state for debugging.

> "The simulation needs to be deterministic so that the result, no matter whether you're simulating on the cloud or at the desktop, [is] always the same."
>
> — James Hui,
> *Product Manager for System Simulation, Wind River*

Factor 7
Basic vs. Advanced Features

Simulation platforms come with a range of capabilities. Basic capabilities typically include emulation of processor and board, use of production binary rather than specially compiled code, script creation to capture simulation knowledge, and command-line interaction for simulation control.

Advanced capabilities include built-in debuggers, playback of data, deterministic simulation, multi-thread capability, extensibility, customization with plugins, dynamic linking of models, and more.

The cost and complexity of simulation tools increase as additional capabilities are added, so it is important to analyze the needs of your particular project — and the capabilities of different simulation tools. Simulation is constantly evolving, so over time you can take advantage of newer tools to increase efficiency.

Evaluating Advanced Simulation Capabilities

- Does it have a built-in debugger so you don't need to rely on a third party for debugging?
- Does it allow data playback so that you can inject data and exercise your software's capabilities?
- Is it deterministic, so that no matter when or where you run the simulation, you get the same result?
- Does multi-thread capability allow you to run a simulation faster than real time (FTRT)?
- Is it extendable so that you can add your own plugin, custom to your program?
- Are you able to dynamically link one model to another so that you are creating system simulation?
- Can you create an artificial virtual target?

BASIC CAPABILITIES

- Emulation
- Run simulation using production binary
- Scriptable
- Command-line interaction

ADVANCED CAPABILITIES

- Built-in debugger
- Playback data
- Determinism
- Can run multi-threaded
- Extendable
- Dynamically linked to other models
- Fault injection
- Support for simulation lifecycle

Factor 8
The Need for a Certifiable Environment

Simulation is increasingly being used in software safety certification, particularly in the context of hardware replacement for systems with limited inventory. Systems that require certification are commonly expensive, in short supply, or both. Simulating the designs eases those concerns.

Another area of focus covers pre-certification tests and validation. The formal certification process can be costly, so reducing the number of needed iterations during the process lowers expense and speeds time-to-market.

Simulation offers several advantages in this context, including the ability to create and repeat test conditions not easily captured using real hardware, support for automated validation processes for short and long durations of time, developer access to edge case issues not easily reproduced, and tests that use hardware failure as success criteria.

Developers and designers must determine whether a certifiable environment is necessary, based on their specific requirements and the regulatory environment they operate in. If it is, simulation can be effective in reducing certification costs and time-to-market.

Simulation has become a crucial component in software development, particularly in the context of the evolution of development practices and the next-generation demands of compatibility with the intelligent edge. Understanding the eight factors explored here can help your team choose the best simulation tools for cost-effective development.

Interested in simulation capabilities for your team?

Contact Us.