White Paper

Modern XIL Challenges

genüen
ideas taking flight

Introduction

The exponential progress of technology of the last few decades equally creates opportunities and challenges for companies in the business of technology. The constant advancements in processing power, data storage and networking surely create fertile ground for innovation in all technical fields. In today's interconnected world powered by such advancements, the likelihood of a good idea to emerge in competing environments is much higher than it was decades ago. This fact imposes a necessity for companies to turn around their ideas very quickly into high-quality customer ready products. Being first to market in today's commercial environment can be the difference between a winner and a loser.

Engineers and scientists were challenged to come up with alternatives for the lengthy process of manufacturing, debugging, and testing of physical components of a technology-heavy system. Arguably, any physical system can be approximately defined by a set of mathematical equations that describe not only how the system interacts with the environment that it is a part of, but also how it responds to external stimuli; those being signals from a giving control system actuator and/or random external disturbances. Leaving quantum physics aside for a moment and constraining the analysis on Newtonian physics of the macroscopic world, the dynamics of such systems follows a set of mathematical equations.

The modeling of physical systems can be performed by a computer's implementation of the system's corresponding set of dynamic equations. As every computer is governed by a given digital clock, every computer-generated model is an approximation of the real physical system, and the implementation of its dynamic equations occurs in this quantized digital domain.

Bélanger, et al. [2] defines real-time simulation as model execution where internal variables and outputs are calculated within the same length of time as a comparable physical system reacts to its environment. In other words, this definition determines that if a model can be executed within a certain time interval that is sufficiently small to provide an accurate enough approximation of the real physical system, it constitutes a real-time simulation. Furthermore, it can be extrapolated that the execution of the modeling equations within a certain time deadline is directly correlated with the accuracy of the simulation.

This hypothesis was the one used by product engineers for the adoption of modeling and simulation as a suitable replacement for time consuming physical builds during the design process. The technique was embraced as fundamental in helping designers to tackle the development velocity-quality dichotomy of the modern world. Fagcang, et al. [1] presents a very interesting plot demonstrating the level of adoption of simulation by the scientific community over the years.

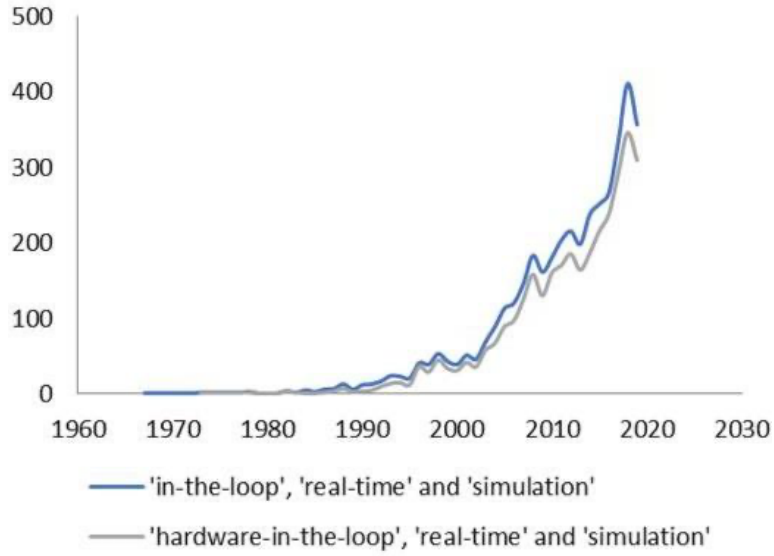

Figure 1. Rise of IL and HIL Publications Over Time

Real-time modeling and simulation tools saw a relatively proportional uptick in innovation. So much so that the task of simulating a system became much more comprehensive. Not only the plant, i.e., the physical system, was being modeled and simulated, but their control systems, actuators, test systems, and virtually all components of the expanded system. The term X-in-the-loop (XIL) was born, and it represented the combination of the different in-the-loop (IL) sub-systems that had to work together on an overall simulation of the entire system.

The development process for in-the-loop systems then evolved into a sequence of tasks similar to:

1. Controller and physical system models are created with the aid of simulation tools, the model-in-the- loop (MIL) phase, in Vijayagopal, et al. [3].
2. Controller models are translated to low-level code and driven to interface with the plant model, the software-in-the-loop (SIL) phase, in Bringmann and Krämer [4].
3. Controller firmware is deployed to embedded systems and driven to interface with the plan model still running in simulation environment, the PIL phase, in Mina, et al. [18].
4. plant model is deployed to appropriate hardware, exposing the overall system to a more realistic environment of interfaces with sensors and actuators, the HIL phase, in Brayanov and Stoynova [5].
5. Test data acquired during the HIL phase is cross referenced against data acquired during the MIL phase for system validation, the regression testing phase.

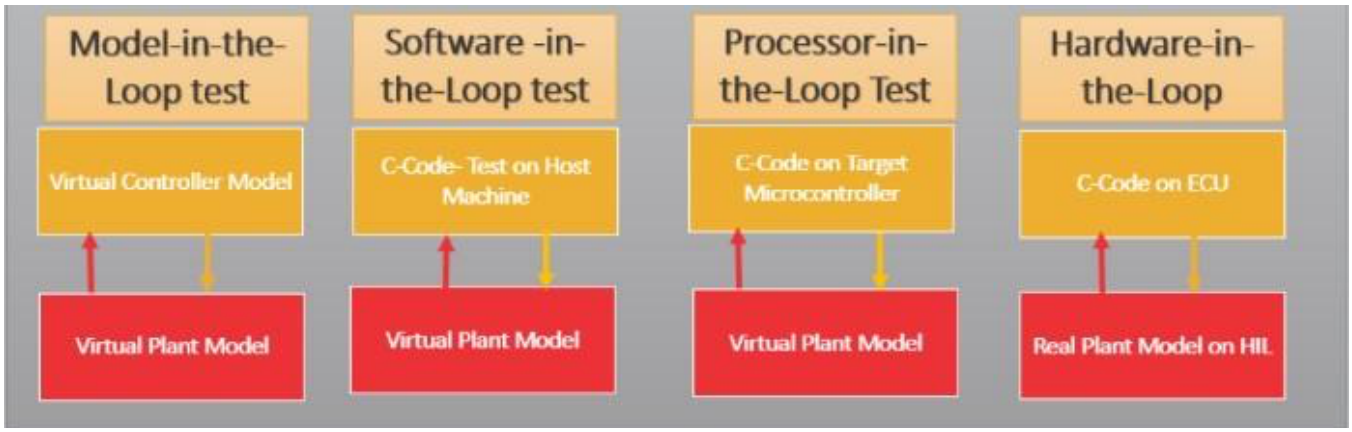

Figure 2. MIL, SIL, PIL, HIL Phases [17]

Modern Real-Time XIL

Up until a few years ago, the sequence of tasks presented in the previous section dominated the way real-time XIL simulation was implemented. Typical real-time in-the-loop control nodes, namely Engine Control Units (ECUs) in the more specific context of the transportation industry as an example and here on out used interchangeably with control nodes, were historically single-processor machines. They typically executed algorithms on inputs coming solely from sensors directly connected to their hardware inputs. These algorithms generated control signals that were routed to actuators, also directly connected to the ECUs hardware and to the plant model.

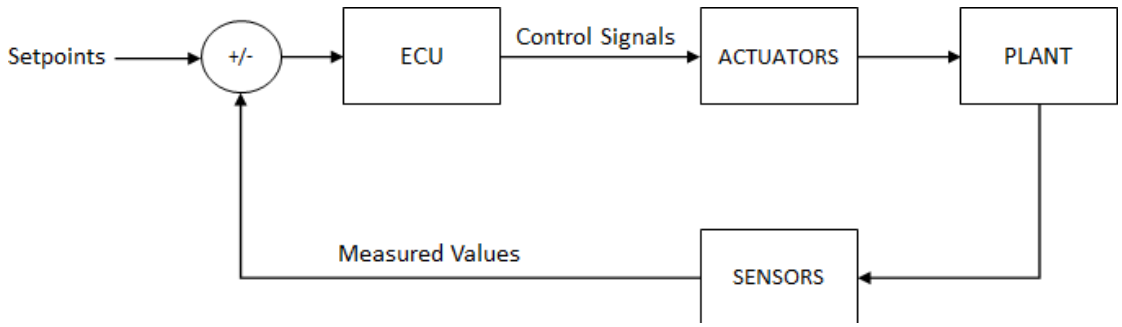

Figure 3. Typical Real-Time XIL System Configuration

The increased complexity of modern real-time systems has driven systems to distributed control architectures. In such designs, the overall system control loop is implemented through a series of interdependent control nodes. Each of these controllers executes real-time control loop algorithms utilizing inputs not only coming from sensors physically connected to its processing unit, but also require data generated by the other control nodes.

The distribution of the overall control system as physically separated, albeit interdependent, control units drove the corresponding split of a single plant model into several plant sub-models. Now, instead of having a simulated system as the typical one displayed in figure 3, systems started migrating to the distributed configuration presented by figure 4.

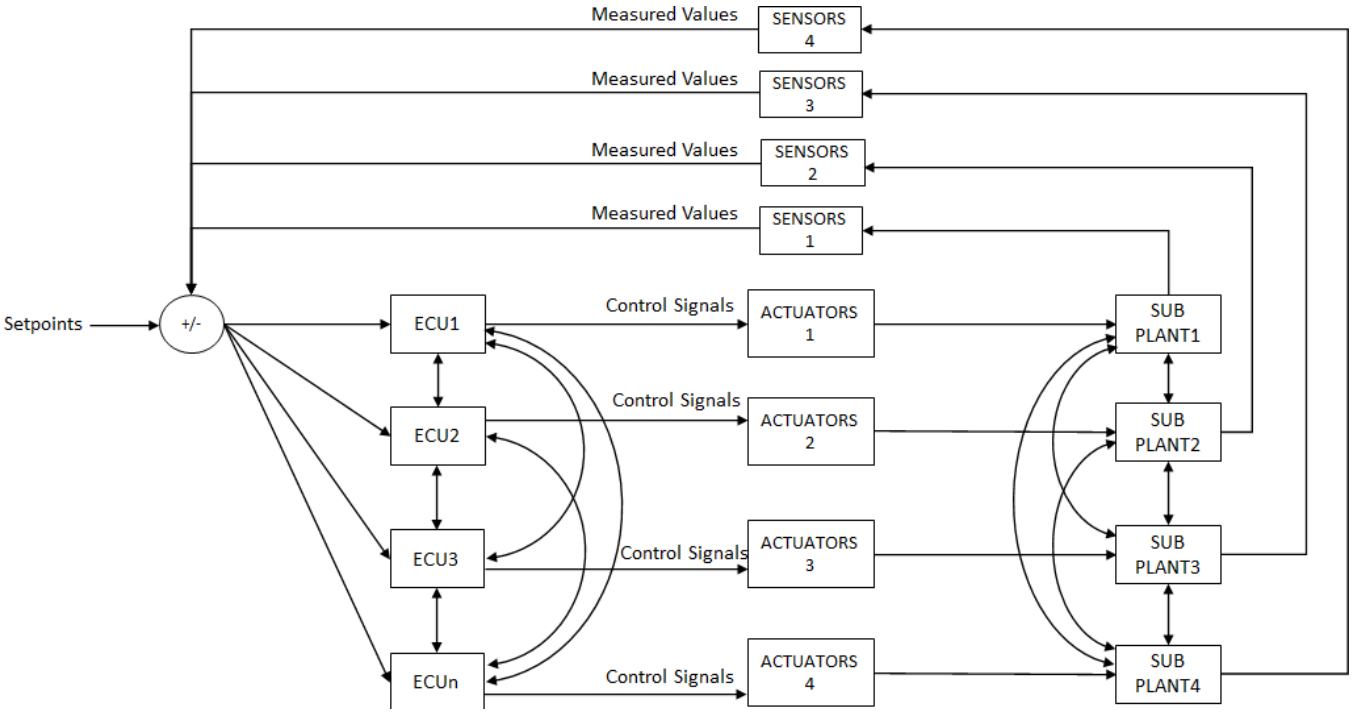

Figure 4. Distributed Real-Time XIL System Configuration

As it can be seen in Figure 4, even though the typical single-processor simulated system illustrated in Figure 3 has been broken down on multiple sub-system, they are in fact not independent from each other. Since they are all working in tandem to achieve a single system-level objective, the multiple ECUs and sub-plants are required to exchange data amongst themselves. Also, even though each ECU is responsible to generate control signals that will control its corresponding sub-plant, it needs to have information about the other ECUs state data to include in its algorithm. The same idea is valid for the sub-plants. Even though each sub-plant is running its own simulation model, it needs to have information about the other sub-plants state data to include in its own model equations. This allows the entire plant dynamics to be correctly approximated and simulated as a whole, even though multiple processor units are executing smaller chunks of the dynamics.

It is important to keep in mind that the goal of the real-time simulation is still the same as defined by [2], regardless of the type of configuration. Therefore, the distributed real-time XIL configuration illustrated in figure 4 suggests the need for a low-latency real-time network to ensure each ECU and sub-plant of the distributed system have all information they need to run their own models within their real-time iteration rates. Therefore, the challenge for the ubiquitous implementation of distributed real-time XIL systems can be focused in solving the problem or high-speed low-latency real-time network communications.

Distributed XIL Challenges

As the previous section concluded, the most important challenge on distributed real-time XIL systems is related to the real-time data transfer between the multiple ECUs and sub-plants that implement the entire system. Network technology has certainly evolved over the course of the last decade, allowing for increased data transfer bandwidth. However, latency on the data transfer remains a challenge for distributed real-time systems. The data transfer latency is a direct influencer on the performance of any real-time system, as such unforeseen communication delays may invalidate the model engineering if the physical system can no longer be approximated well enough by the digital equations.

Moreover, TCP-based networks add jitter on top of the extra latency of distributed systems. Some technologies can be used for the mitigation of this issue, namely UDP [6], EtherCAT [7] and Time Sensitive Networks (TSN) [8]. However, these techniques are insufficient for systems where models are required to execute at a real-time rate beyond the typical empirical 1-5KHz limit; the sweet spot for the techniques mentioned.

Reflective memory (RM) [9] is another technology that showed promise when it first appeared. In a RM network, a copy of the entire memory composed of the states from all nodes is kept locally by each network node. It is a special memory sharing system between multiple nodes in a network [10]. Figure 5 illustrates a typical RM network topology.

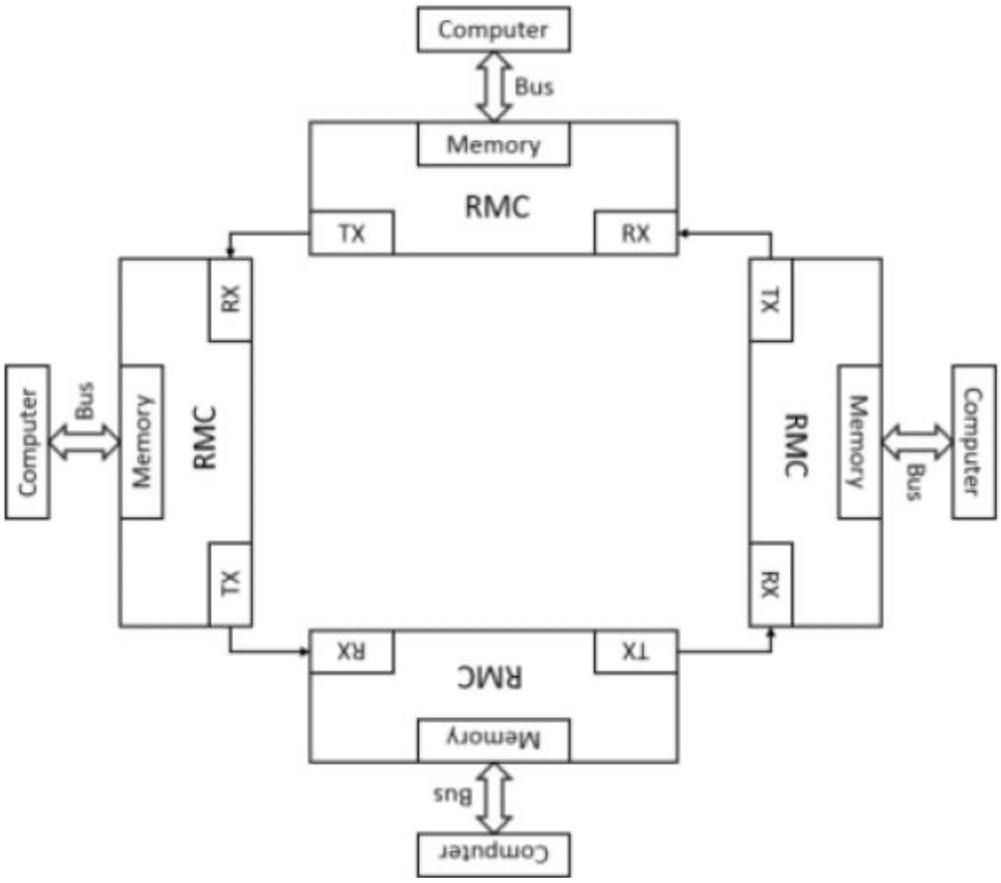

Figure 5. Reflective Memory Typical Network

When a node needs to make a write operation to the RM network, it places a typical TX request to the next node in the ring. The next node in turn reads the data and makes a TX request with the same data that it just read to the next node down the ring network. The process is repeated until the data makes a complete round in the network and is returned to the original writer node. One of the main advantages of this technique is that RM nodes can make updates to a mapped local memory range independently from the node's CPU, as in [11]. Therefore, the local RM copy can be updated on a much higher speed than a given application running on the local CPU that will in turn consume the data. This allows for a higher-speed hardware-based operation to guarantee a level of real-time determinism in the updating of the virtual local memory.

Even though it is possible for a RM network to be configured in a star topology, the extra latencies and data overhead added by the required routing tasks ends up approximating the real-time updating speed of the RM to the typical ones achieved by other technologies such as RT EtherCAT. This led to the use of a ring topology to increase real-time speed. However, the biggest problem of ring topology networks comes to play at the RM implementation; the entire network goes down if a single node is down.

Moreover, since each node needs to touch the data before an update of the RM space, it becomes obvious that the number of nodes and potentially their geographical location in relation to each other influence the performance of the real-time updates when it comes to speed and latency. Some attempts were made by the community to work around these constraints, such as in [12], [13] and [14]. However, they proved insufficient for the adoption of the technology on the implementation of real-time distributed XIL systems.

The way some companies decided to address these challenges was through the creation of customized and very specific equipment targeted at distributed real-time XIL applications. Invariably, such solutions involve either some sort of proprietary hardware backplane communication bus [15] or the implementation of an FPGA-based proprietary protocol solution [16] for the connectivity of the multiple nodes.

Both approaches are sub-optimal. First, a vendor-specific closed-architecture offering to address such foundational level problem as real-time low-latency communication constrains the user to only be able to select hardware and software components provided by this vendor or assume the risk of incompatibility between the multiple components of the distributed XIL system. This presents the obvious issue of the solution being constrained to the vendor's offerings as opposed to the most applicable components to the given application that may be a combination of different vendors.

Second, these types of solutions tend to be point-to-point in nature; especially the ones around FPGA- based communication protocols. This presents several limitations to the designer. First, the total number of nodes the overall system usually can only stretch to the maximum number supported by the point-to-point and data multiplexing hardware structure. Second, geographical location of the nodes becomes limited to the distancing specification of the given communication solution. Third, expandability of an existing system to include new components to implement modified systems become compromised. And lastly, work collaboration has increased significantly in the last few years.

Collaboration may require a comprehensive strategy for the integration of existing XIL systems between multiple departments, research groups and potentially even multiple contractors. The retrofitting of existing XIL systems to potentially turn them into sub-XIL systems of a larger XIL simulation is not possible when each sub-system may be provided by a different vendor.

Conclusion

This paper presented an introduction to real-time XIL simulation systems and how technology advancements are driving the industry in the direction of distributed real-time XIL systems. It demonstrated how the main challenge of such systems can be addressed through the solution of a multi-decade old problem, low-latency real-time network communications. It reviewed the existing technologies at the time of the writing of this paper that attempt to solve this problem and presented the limitations of each approach and how they are not a good fit for the implementation of distributed real-time XIL systems.

Vendor custom solutions partially address the problem, but on a very constrained manner. Unless vendors are willing to embrace the presence of competing solutions instead of trying to suppress them, it is very unlikely the industry will benefit from the freedom to select the components that are most appropriate for a given application. The continuation of this status quo is a clear barrier to the potential connection of existing XIL system from multiple vendors into larger distributed real-time XIL systems.

RM certainly carries a lot of potential, provided its limitations on number and distance of nodes as well as the typical risks of ring network topologies could be addressed. Ideally, what the industry currently needs is a technology that could implement some sort of super-low latency high-speed real-time reflective memory solution over a typical star network topology. A technology like this, implemented on off-the-shelf equipment and existing network infrastructure, that offers an open architecture to allow multiple vendors to co-exist and be mixed with minimum limitations could be the silver bullet the industry needs.

References

1. Fagcang, H., Stobart, R., & Steffen, T. (2022). A review of component-in-the-loop: Cyber-physical experiments for rapid system development and integration. *Advances in Mechanical Engineering*, *14*(8), 16878132221109969.
2. Bélanger, J., Venne, P., & Paquin, J. N. (2010). The what, where and why of real-time simulation. *Planet Rt*, *1*(1), 25-29.
3. Vijayagopal, R., Michaels, L., Rousseau, A. P., Halbach, S., & Shidore, N. (2010). Automated model based design process to evaluate advanced component technologies. *SAE technical paper*, 01-0936.
4. Bringmann, E., & Krämer, A. (2008, April). Model-based testing of automotive systems. In *2008 1st international conference on software testing, verification, and validation* (pp. 485-493). IEEE.
5. Brayanov, N., & Stoynova, A. (2019). Review of hardware-in-the-loop-a hundred years progress in the pseudo-real testing. *Electrotech Electron*, *54*, 70-84.
6. Larzon, L. A., Degermark, M., & Pink, S. (1999). *UDP lite for real time multimedia applications*. Hewlett-Packard Laboratories.
7. Jansen, D., & Buttner, H. (2004). Real-time Ethernet: the EtherCAT solution. *Computing and Control Engineering*, *15*(1), 16-21.
8. Finn, N. (2018). Introduction to time-sensitive networking. *IEEE Communications Standards Magazine*, *2*(2), 22-28.
9. Jovanovic, M., & Milutinovic, V. (1999). An overview of reflective memory systems. *IEEE concurrency*, *7*(2), 56-64.
10. You, T., Du, C. L., & Zhu, Y. A. (2009, August). Supporting technology for virtual numerical control system based on RTX and reflective memory network. In

2009 Fifth International Conference on Natural Computation (Vol. 1, pp. 319-323). IEEE.

11. Bian, Z., Xie, L., & Wu, B. The FPGA Design and Implementation of Reflective Memory Card Based on the PCIE Bus.
12. Shen, C., & Mizumuma, I. (2000). RT-CRM: real-time channel-based reflective memory. *IEEE Transactions on Computers*, *49*(11), 1202-1214.
13. Jacunski, M., Moorthy, V., Ware, P. P., Pillai, M., Panda, D. K., & Sadayappan, P. (1999, January). Low Latency Message-Passing for Reflective Memory Networks. In *International Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing* (pp. 211-224). Springer, Berlin, Heidelberg.
14. Hasegawa, M., Nakamura, K., Zushi, H., Hanada, K., Fujisawa, A., Mitarai, O., ... & Higashijima, A. (2015). Development of a high-performance control system by decentralization with reflective memory on QUEST. *Fusion Engineering and Design*, *96*, 629-632.
15. Backplane (2022). https://www.dspace.com/en/inc/home/products/hw/simulator_hardware/scalexio.cfm#175_58972.
16. FPGA (2022). https://www.opal-rt.com/fpga-and-i-o-expansion-box/
17. Shepard, Jeff (2022). How do MIL, SIL, PIL and HIL simulation and testing relate to MBSE. https://www.eeworldonline.com/how-do-mil-sil-pil-and-hil-simulation-and-testing-relate-to-mbse-faq/
18. Mina, J., Flores, Z., López, E., Pérez, A., & Calleja, J. H. (2016, June). Processor-in-the-loop and hardware-in-the-loop simulation of electric systems based in FPGA. In *2016 13th International Conference on Power Electronics (CIEP)* (pp. 172-177). IEEE.