Computers & Industrial Engineering 186 (2023) 109733

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Towards digital twins for safety demonstrations: Interfacing strategies for FPGA-targeted applications

Ludvig Björklund [a,*], Johannes Schick [b], Mary Ann Lundteigen [a], Markus Glaser [b]

[a] Department of Engineering Cybernetics, Norwegian University of Science and Technology, Norway
[b] High Integrity Mechatronic Systems, Aalen University, Germany

ARTICLE INFO

Keywords:
Digital Twin
Functional Safety
Simulation-based verification
Hardware Description Language
FPGA-based control

ABSTRACT

For facilitating a framework of digital twins for safety demonstrations, this research paper identifies feasible interfacing strategies to connect a digital twin of novel all-electric actuation systems with safety-critical functions targeted for an FPGA device. The framework can improve safety demonstrations with a lifecycle perspective and one-to-one twinning if the values of the framework are realizable. For selecting a suitable interfacing strategy, the paper proposes five performance criteria to preserve the framework's values while interfacing the controller logic with a digital twin. The paper introduces a safety-critical case study, thereby establishing reasoning, literature, and simulation testing of the interfacing strategies regarding relevant interfacing considerations and the performance criteria. We recommend using behavioral models during the design of FPGA firmware and commercial tools for verification against a reference model during the synthesizing steps. For the final development stage, we propose an FPGA-based digital twin, which offers high computational speed, scalability for integration testing, and comprehensive testing of the digital logical circuits. Implementing the recommendation requires practical implications, including adhering to FPGA configuration constraints and developing custom resources for interacting with the FPGA-based digital twin.

1. Introduction

The oil and gas industry is shifting towards increasing the digitalization and electrification of subsea equipment to reduce production costs and mitigate environmental impact. The expectation is that electric subsea production and control systems can bring about several benefits, including improving system reliability and availability, enhancing diagnostic capabilities, and increasing the overall operational safety (Mahler et al., 2019). A subsea system under consideration for electrification and digitalization is the actuation control system of a safety valve, which constitutes a critical part of the subsea Christmas tree system. The Christmas tree system, installed on top of the wellhead, comprises a network of pipes, valves, and fittings. Closing the flow of hydrocarbons is a critical functionality of the safety valves of the Christmas tree.

An all-electric actuation system can replace existing approaches of hydraulic-powered safety valves and thereby mitigate costs, reduce environmental impact, and provide extended controllability and diagnostics of the state of the valve. The primary function of the safety valve is to stop the hydrocarbon flow in the event of a production or emergency shutdown, making it a critical component in reaching a safe state of the production process. Available solutions for the safety valve are typically spring-based and rely on hydraulic power to store energy in the spring, transitioning the subsea production system to a safe state by a power cut. Fig. 1 illustrates a proposal for an all-electric actuation system to transition a safety valve and cut the hydrocarbon flow.

The figure illustrates the main components of the all-electric actuation system, including the communication lines to topside facilities and external power supply. The concept operates on an energize-to-close principle, requiring a power supply to the motor to reach a safe state. Design and control of the actuator control system of the safety valves must comply with functional safety standards such as IEC 61508 (IEC 61508, 2010). Safety Integrity Levels (SIL) are a classification system defined by IEC 61508 to quantify the risk reduction reachable by safety functions in a system. For actuation systems responsible for safety-critical valves in the Norwegian oil and gas sector recommended practices and standards stipulate SIL levels 2 and 3 (GL 070, 2020). The IEC 61508 standard is a generic standard commonly used for certifying novel designs and new products for use in safety systems.

To demonstrate functional safety, testing the embedded controllers, e.g., the Battery Management System (BMS) and the motor controller, is a critical activity to reach certification. Testing activities of software-dependent systems are resource intensive and require a manifold of test

* Correspondence to: Kinnvegen 5, 7045 Trondheim, Norway.
E-mail address: ludvig.bjorklund@ntnu.no (L. Björklund).

https://doi.org/10.1016/j.cie.2023.109733
Received 30 January 2023; Received in revised form 16 October 2023; Accepted 30 October 2023
Available online 2 November 2023

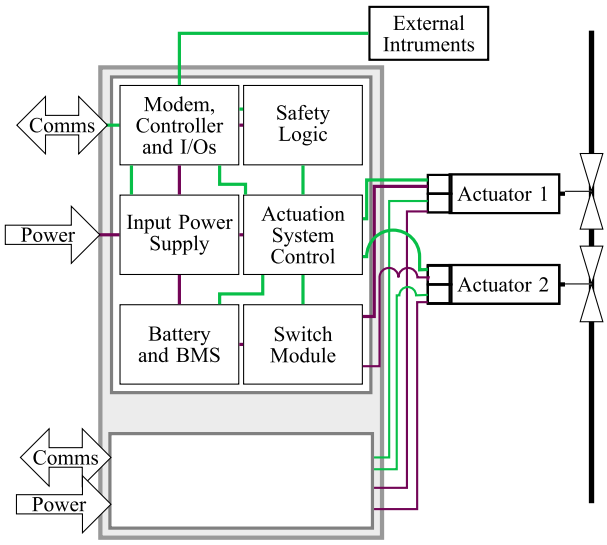

Fig. 1. A proposed architecture for an all-electric subsea actuation system.

cases to reach adequate test coverage (Kapinski et al., 2016). Rewriting code, removing errors, or addressing gaps require re-executing the extensive list of test cases (IEC 61508, 2010). A digital twin capturing safety-relevant behavior of the hardware components, directly interfacing with logic-embedded functionality, can demonstrate safe (or unsafe) behavior in software-dependent safety-critical systems. A digital twin can provide several benefits, simplifying experimental setup, reducing testing time, eliminating risks, and revealing insights about hidden states. The potential to scale up testing capabilities and generate evidence over the entire lifecycle are distinctive benefits for demonstrating the safe behavior of the system. Furthermore, the digital twin facilitates demonstrations of future control strategies and diagnostic tools made available via software updates. An opportune interface with the safety-critical software modules is critical for reaching the scalability and speed improvements of using digital twins for safety demonstrations and selecting appropriate interfacing strategies is crucial for the digital twin framework.

Implementing controllers and diagnostic applications on a Field Programmable Gate Array (FPGA) provides flexibility, high-speed parallel processing, and the ability to implement complex algorithms, ensuring fast response times and high accuracy (Monmasson et al., 2011). The logic modules, or "controllers" with an FPGA target platform, such as the battery management system and the motor controller, provide an interesting case study for arriving at suitable interfacing strategies. Unlike traditional software, executing on a general-purpose processor, FPGA firmware is a combination of hardware description languages (HDL) such as VHDL or Verilog for configuring the digital logical circuits on the FPGA. In this article, FPGA firmware specifically denotes the control and diagnostic functions implemented using HDL for execution on an FPGA. Once physically implemented on the FPGA board, these functions are still referred to as FPGA firmware, as executable on the FPGA device.

The design of novel safety-critical systems is often aided by following a standard, e.g., IEC 61508, for ensuring conformance and facilitating certification. The standard IEC 61508 provides recommended practices and heuristic approaches for verification and validation activities for the functional safety of electrical/electronic/programmable electronic safety-related systems (IEC 61508, 2010).

1.1. Objective

The paper contributes to digital twins for safety demonstrations by researching the development of FPGA-targeted controllers, identifying feasible interfacing strategies, and proposing a set of criteria for supporting the decision. The research aims to improve safety demonstrations in the complete lifecycle by evaluating and selecting interfacing strategies capable of preserving the values of digital twins for safety demonstrations. The case study will contribute to understanding existing solutions, provide guidance and recommendations for future implementations, and leverage the lifecycle perspective of digital twins for future firmware updates. The paper's contribution lies in facilitating the integration of a digital twin in the recommended developmental flow of safety-critical controllers for FPGAs.

1.2. Structure

In Section 2, the paper examines the concept of digital twins, simulation-based verification procedures, and outlining criteria for assessing the effectiveness of digital twins in safety demonstrations. Section 3 provides background on FPGAs, recommended design flow, and the challenges of simulating FPGA firmware on a processor. Section 4 introduces strategies for interfacing the digital twin with FPGA firmware and a set of criteria to evaluate the interfacing strategies. Subsequently, in Section 5, a case study introduces the challenges and opportunities of the framework and functions as a basis for determining the suitability of the interfacing strategies. Section 6 discusses the results, assumptions, and limitations of the research. The final section presents the conclusions from the study and a discussion of future work.

2. Digital twins for safety demonstrations

The digital twin concept was initially discussed by Michael Grieves, in a presentation in 2002, and he is often accredited as the conceptual father. Arguably, claims that the Apollo 13 mission principally operated a twin connecting to the actual asset indicate a debatable origin of digital twins (Mashaly, 2021). Regardless of the conceptual origin, the digital twin concept is gaining significant attention and interest with proposed utility for a range of use cases (Liu et al., 2021; Singh et al., 2021). This section provides a background literature review on digital twins, offering insights into general values and introducing a use case of safety demonstrations.

2.1. Defining the digital twin

Three defining components of the digital twin concept are a physical asset (capable of capturing information in the physical domain), a digital replica capable of integrating information about the asset, and a communication layer transferring information between the physical and the virtual domains (Grieves & Vickers, 2017). The fundamental idea of the digital twin concept is connecting a physical asset (i.e., a system, component, or process) to a digital replica. The connection, or the twinning, between the virtual and physical domain encapsulates a lifecycle perspective, from an immaterial asset to an operational instance of the physical asset (Jones et al., 2020). Twinning in the operational stage connects instances of the physical asset with a unique and custom digital twin instance. Fig. 2 illustrates the lifecycle perspective and the subsequent twinning on a one-to-one instance basis of the digital twin concept.

The idea of the life-cycle-driven view incorporated in the digital twin-concept is to update and carry information throughout the life-cycle of the physical asset. The digital twin environment is the virtual environment for the realization of the digital twin, and tools for implementing digital twins can differ depending on the use cases of the digital twin. All simulations of environmental effects and operating conditions in the physical domain are part of the digital twin environment (Grieves & Vickers, 2017). The term digital twin aggregate describes a collection of digital twin instances connected to instances of the physical asset. Aggregating the data from the collection of digital twin instances can improve predictive capabilities, particularly diagnostics, e.g., by integrating behaviors indicating a fault or failure (Glaessgen & Stargel, 2012).

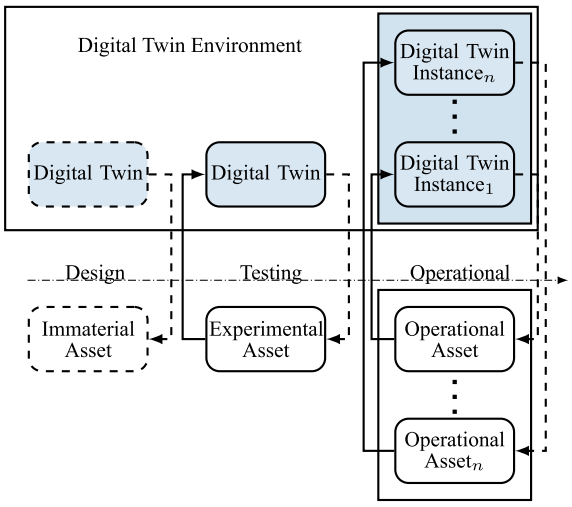

Fig. 2. The principal components in the lifecycle of a physical asset with a corresponding digital twin.

Rasheed et al. (2020) identify key technology enablers in the physical and the virtual domain for advancing the digital twin concept, including data compression, communication technologies, sensor advancements, high-fidelity simulators, and cloud computing. The improved capabilities of technology facilitate the development of digital twins capable of interacting with the physical asset. However, further technological developments are needed to achieve the ambitious vision described by Glaessgen and Stargel in Glaessgen and Stargel (2012), where integrated information from multiple digital twin instances and detailed physics-based models are employed. Nonetheless, implementing digital twins allows for initiating the construction of feasible applications on a smaller scale, offering practical benefits and improvement over currently existing alternatives without requiring a full-scale realization of the perceived integration of the concept.

Twinning is defined as "the act of synchronization" in the systematic literature review by Jones et al. (2020). The authors highlight the importance of syncing between the physical and the virtual domain to realize the digital twin concept, identifying it as a significant characteristic and critical act in creating digital twins. The twinning process also involves examining the physical asset and adjusting underlying model parameters that capture expected behavior (Björklund et al., 2022). The authors of Jones et al. introduce the terminology of "physical-to-virtual" to describe the process of transferring information from the physical domain to the virtual domain and "virtual-to-physical" in the opposite direction of the information flow. Continuously updating the model parameters of the digital twin, using the physical-to-virtual information transfer, enables tracking and monitoring the state of the physical asset. A continuous parameter adjustment enables the digital twin to stay synchronized with the physical asset, providing reliable and up-to-date information about its behavior. The virtual-to-physical communication would, for example, allow an operator of the digital twin to update control parameters, subsequently changing the control parameters of the physical controller (Jones et al., 2020).

The term digital twin in industrial and academic documentation spans a range from standalone models without direct effect to fully automated (change at either end of the twinning causes a change in the other, automatically) bidirectional information exchange as noted in a literature view of the utilization of the term (Kritzinger et al., 2018). As evidenced by the vast utilization of the term digital twin the exact definition of a digital twin is contested. The authors of Kritzinger et al. (2018) propose an integration-based classification scheme for digital twins, classifying proposals and applications of digital twins into three distinct categories: digital model, digital shadow, and digital twin. The communication layer of the digital model is considered entirely manual, i.e., a state change in the physical asset requires user intervention to update the state in the digital model. The digital shadow includes the capability to automatically update the digital models to track state changes of the physical asset and manually transfer information from the digital twin to the physical asset. The digital twin facilitates automatic state changes in the physical asset in response to changes in the digital twin and vice versa (Kritzinger et al., 2018). The term digital shadow, however, implies a mere mirroring or tracking capability, and alternative perspectives on classifying digital twins exist. Depending on the use case, automatically updating the parameters of the physical assets can become a barrier to development and decision-making. The term "twin" is more appropriate to cover the automatic physical-to-virtual direction, because it highlights the capability of simulations to go beyond the limitations and normal operational conditions. A well-designed digital twin has underlying computational models capable of accurately describing the properties of the physical asset and thereby capturing the behavior of the physical asset. The capability to compute an expectation of the physical asset behavior enables supporting decisions by producing evidence with the digital twin, even in scenarios rarely occurring in the physical asset.

Bonsch et al. (2022) propose an alternative to the integration-based classification methodology, a subject-oriented reference model for defining the underlying foundations of the digital twin concept. This reference model serves as a functional decomposition methodology, similar to the aforementioned digital twin element separation, allowing for a deeper exploration of the digital twin concept while maintaining flexibility in the technical specifications that define what constitutes a digital twin. The authors utilize several use cases to highlight that digital twins can span maturity levels of functionality. At the first level, the functionalities amount to collecting data and displaying the state of the physical asset. The second level of functional maturity includes data processing, and the third level additionally includes simulation functionality and enabling predictions of future states. The final level of maturity includes the ability to act on the predictions, either prescribing or implementing an action. Additionally, examples of digital twin use cases in the paper illustrate the different perspectives between various stakeholders and actors, indicating perceptions and miscommunication as a potential source of confusion surrounding the challenge of defining digital twins (Bönsch et al., 2022).

2.2. Digital twin elements

Conceptual descriptions of the digital twin include all information related to the physical asset, including requirements, specifications, bill of materials, 3D models, etc., (Glaessgen & Stargel, 2012; Grieves & Vickers, 2017). Capturing all information and behavior of an asset in various scenarios to support or implement actions for key decisions in modern cyber–physical systems can rapidly become unwieldy in terms of scale. In a recommended practice by DNV on the qualification and assurance of digital twins, the authors propose a functional decomposition of the digital twin into multiple elements. The functional elements work individually for a specific use, providing the necessary information and evidence to support key decisions. In an adaption of the illustration from DNV's recommended practices in Fig. 3, incorporating a connection between the physical and the virtual domain, the digital models for virtual experiments and then, one can argue that the functional elements are individual digital twins or digital twin elements.

The figure depicts the potential to connect a digital twin element with other digital twin elements to accomplish complex objectives. Additionally, the recommended practices emphasize the significance of quality indicators in assessing the trustworthiness of evidence generated by the digital twin. To make predictions, the digital twin typically employs computational models to capture the expected behavior of the corresponding physical asset. The scope of this article centers on

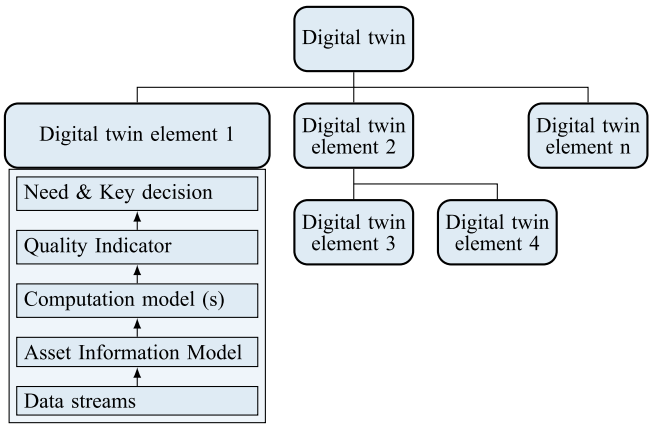

Fig. 3. Separating the digital twin into digital twin elements facilitates the management of various use cases.
Source: Adapted from DNVs guideline on qualification and assurance of digital twins (DNV GL, 2020)

digital twin elements that use underlying computational models for simulating the expected behavior of the physical asset. Digital twin elements containing information and facilitating virtual inspection using 3D models and geometrical models are outside of the scope. For a more general discussion on digital twin concepts, see the comprehensive works on digital twins (Glaessgen & Stargel, 2012; Grieves & Vickers, 2017; Jones et al., 2020; Rasheed et al., 2020). By utilizing underlying computational models, the digital twin can predict the behavior of the physical asset. Design of the underlying computational models are either data-driven, physics-driven, or a combination of both approaches (Grieves & Vickers, 2017; Rasheed et al., 2020). Data-driven digital twins utilize statistical models or machine learning to model the behavior of the physical asset based on historical data of the physical asset. Design of underlying models for physics-driven digital twins utilizes mathematical relationships of the physical parameters to capture the expected behavior of the physical asset (Liu et al., 2021; Rasheed et al., 2020). This article will be limited to physics-based digital twins, due to the inherent requirement of traceability and interpretability for demonstrating the safety of cyber–physical systems. In addition, safety functions rarely activate, and historical data pertaining to safety-relevant behavior represents only a minuscule portion of the operational data, becoming a challenge for creating data-driven digital twins.

2.3. Establishing trust in digital twins to generate evidence

Adil et al. (2019) discuss the limitations of physics-based digital twins stemming from the current understanding of underlying physical properties. Physics-based underlying models powering a digital twin element are bound by knowledge of physics and fail to capture behaviors or phenomena not fully comprehended within existing theories and governing equations. The uncertainties inherent in digital twins highlight the need for careful consideration in assessing the trustworthiness of the evidence they generate. All models, including the underlying computational models of digital twins, are imperfect representations of reality but can still provide valuable insights and prove useful.

Organizations such as DNV are proactively addressing the challenge of establishing trust in the digital twins to generate evidence. DNV has developed a set of recommended practices, DNVGL-RP-A204, outlining a guideline for a successful qualification and assurance process and a systematic approach to establishing trust in digital twins. Creators of digital twin elements can mitigate uncertainties, demonstrate conformance to the standard and subsequently establish trust in the evidence by following the recommended practices (DNV GL, 2020). The recommended practices emphasize that the level of qualification and assurance for digital twin elements should be proportional to the consequences of the key decision. Following the recommended practice mitigates uncertainties and enhances confidence in digital twins by incorporating quality indicators within the digital twin elements, quantifying a numerical value indicating trust in the evidence generated by the digital twin. It is important to note that the recommended practices explicitly exclude decisions related to autonomous safety-critical systems from the process. Instead, the authors emphasize the need to consider additional requirements from safety standards such as IEC 61508, highlighting the necessity to conform to established safety guidelines when utilizing digital twins in safety-critical contexts (DNV GL, 2020).

Use cases are diverse and span industrial sectors such as healthcare, aviation, and automotive industries, to mention a few (Rasheed et al., 2020). For example, in the manufacturing industry, digital twins have been proposed or employed for applications to optimize production processes, reduce defects, and enhance product quality (Tao et al., 2019). Furthermore, in transportation and logistics proposals for digital twins to optimize fleet operations, improve route planning, and enhance the supply chain by accessibility and virtual domain visibility (Glaessgen & Stargel, 2012). Jones et al. (2020) provide a comprehensive study, including an overview of use cases of digital twins while characterizing the digital twin (Jones et al., 2020). Additionally, for an in-depth perspective on the values of digital twins, including potential use cases, the article by Adil et al. (2020) offers valuable insights and discussion (Rasheed et al., 2020). The article "A classification proposal of digital twin applications in the safety domain" examines current and proposals for use cases in the safety field (Agnusdei et al., 2021). The authors propose a framework for assessing digital twin use cases in the safety domain, focusing on data acquisition systems, data processing models, and specific safety considerations. The result indicate a potential for digital twins to enhance safety management and control, improve efficiency in safety assessment, and support risk prevention measures.

2.4. A digital twin based framework for safety demonstrations

Safety valves are critical to reaching a safe state and shutting the hydrocarbon flow. In a software-dependent all-electric actuation system, reaching the safe state requires the correct execution of several components. In traditional systems, the removal of hydraulic power triggers the transition to a safe state. In stark contrast, the all-electric actuation system operates on an energize-to-close principle and requires supplying power to the motor for the successful transition to the safe state. The all-electric actuation system relies on several subsystems, diagnostics, and control algorithms embedded in controllers utilizing built-in logic to transition the valve and stop the hydrocarbon flow. Demonstrating the capacity of the novel and highly software-reliant system requires fulfilling testing and verification of the system for reaching a target SIL certification. Sufficient coverage of the behavior via testing and verification activities of logic modules controlling cyber–physical systems is a requirement to conform to IEC 611508. Experiments observe behavior in the physical domain, and simulations use computational models to estimate the behavior in the virtual domain. In an article on verification activities of embedded control systems (Kapinski et al., 2016), the authors provide a comprehensive exploration of the concepts associated with computational models for testing and verification activities, in the context of embedded control systems.

To achieve verification, the test engineer shall prove that the system satisfies a specific property for all sets of parameters and inputs, possibly infinite. Formal methods apply mathematical techniques to formally state requirements on the logical module and prove that it is conforming to requirements. Successful application of formal methods can prove that the system's behavior will be within acceptable conditions across all parameters and inputs. Formal methods, as noted by Kapinski

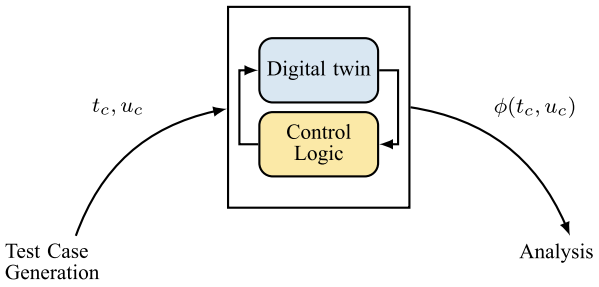

Fig. 4. A proposal for interfacing a digital twin with a safety-critical controller for demonstrating the capability of the system to demonstrate the safety-critical functions of the controller by evaluating the safety-related behavior.

et al. (2016) require mathematical expertise, are computationally complex, could be problematic in handling continuous systems, and the tool support can be a limiting factor (Kapinski et al., 2016). In a state-of-the-art survey on the application of formal methods for internet-of-things applications, several issues are common in the application of formal methods in practice. The authors note that simplified models are used to verify the software on a modular level (neglecting interactions), state space explosion limit model computational speed, and verification of timing issues are overlooked in several projects (Souri & Norouzi, 2019).

Simulation-based methods can scale up testing activities using mathematical models mirroring the expected behavior of a physical asset in various scenarios and operational conditions (Kapinski et al., 2016). Testing, in this context, aims to explore a subset of potential inputs and parameters for assessing the system's behavior against predefined requirements and conclude that satisfactory performance has been demonstrated based on these tests. Testing acknowledges the limitations of exhaustive testing and instead focuses on representative scenarios to make claims about the system's capabilities. Falsification can be viewed as a subset of testing, utilizing simulations of computational models to identify parameters, inputs, or combinations that falsify claims of the system. Since it is impossible to test all possible parameters using simulation-based, inputs and configurations of combinations, it is not possible to completely verify the safety of the system through testing. However, it is possible to achieve sufficient test coverage, as noted in the IEC 61508 part on software development (IEC 61508, 2010).

A digital twin element capable of capturing all safety-related behavior of the hardware, interfacing with the logic module, can facilitate simulations of safety-related scenarios and assess the system's response in the presence of known and unknown faults. Integration of the digital twin interfacing with the "controller" in a framework for safety demonstration is a critical aspect in the generation of evidence, i.e., capturing the safety-related behavior of the system as a simulation time series for analysis and automating the selection. However, logical module may be a more sensible descriptive term since the intention is to evaluate all the safety-critical logic solvers embedded in the all-electric actuation system. The simulation-based testing occurs offline, generating evidence in the virtual domain to support claims about safety. The safety demonstrations interface the digital twin with the logic modules within an offline setting and do not impact the operation of the physical asset. The testing environment of the framework is illustrated in Fig. 4.

To initiate the framework, a set of test case parameters denoted as $t_c \subset$ and an operational scenario u_c are sent to the simulation environment generating a time-series capturing the safety-related system of the behavior $\phi(t_c, u_c)$. However, the safety claim is only valid for the test case as the safety-related behavior $\phi(t_{c_i}, u_c)$ is a function of the test case parameters. Reaching sufficient test coverage requires extensive testing to cover hardware faults, physical degradation, and potential bugs in the controller and evaluate the safety-related behavior. The test case is one instance of the set of test cases required to reach sufficient test coverage, and by utilizing different test case parameters by selecting parameters from the set of model parameters $m_p \in M_p$, fault parameters $f_p \in F_p$ and logic module parameters $l_p \in L_p$. Even working from an assumption of a limited set of potential test case parameters, optimally selected to maximize the test coverage with a limited amount of test cases, the number of test cases is likely substantial. Therefore, an identifiable bottleneck in the framework is the simulation of the safety-related behavior illustrated in Fig. 4.

The test case parameters t_c consist of model parameters m_p, fault parameters f_p, and "logic module" parameters l_p. The model parameters m_p represent a subset of the identified physical parameters that describe the properties used to construct the mathematical equations that govern the behavior of the physical asset being twinned. The fault parameters f_p are associated with faults or issues in the physical system, either internal or external faults. The logic parameters l_p describe issues with the logic, which can include external commands from interacting controllers or information losses.

To reach sufficient testing of the system, selecting interfacing strategies capable of testing the final implementations of the controller is critical. The selection of interfacing strategies covering complete controller development can improve the development process through the transparency of the test case parameters. The feasibility of the interfacing strategies will be affected by the target platform for the controllers. Developing a controller for a microcontroller versus an FPGA device will require different development flows. This paper centers on interfacing strategies for connecting the digital twin element with controllers targeted for FPGA devices.

3. FPGA as a target platform

FPGAs are electronic devices that offer flexible and reconfigurable digital logic functionality. FPGAs provide a versatile platform for designing and prototyping complex digital systems by describing the circuitry in software, enabling rapid development and deployment of the final hardware implementation. Demonstrating the functional safety of the FPGA-based system requires rigorous testing and adherence to safety standards. This section reviews FPGA devices, including architectural capabilities and limitations. Subsequently, we explore recommendations for the developmental flow of FPGA firmware according to safety standards, highlighting the software-reliant nature of the development process. Understanding how the hardware design of FPGAs is primarily configured by software programming, utilizing a software-based language to describe the circuitry, and uploading a programming file to configure the FPGA device and enabling the functionality of the control system is of critical importance to identify feasible interfacing strategies.

3.1. FPGA as a target device

FPGAs are part of the family of programmable devices, facilitating field re-programmability by configuring digital logic circuits (Brown & Rose, 1996; Monmasson & Cirstea, 2007). Design using software tools describes the layout of the digital logical circuits. The term FPGA firmware describes the complete lifecycle of the logical modules, from initial software textual descriptions of circuits to the final implementation on the board.

Since the launch of FPGAs in the 1980s, integration of A/D converters and improvements in component density have driven a technological maturation. The technological evolution improves performance, and user-friendly tools facilitate the simpler design of FPGA firmware. FPGAs allow for a high degree of configurability and soft processors can be implemented on FPGAs, enabling configurations to function as a microcontroller. The inherent configurability can also enable true hardware design and facilitate real-time implementation of FPGA firmware with high sampling requirements and applications requiring

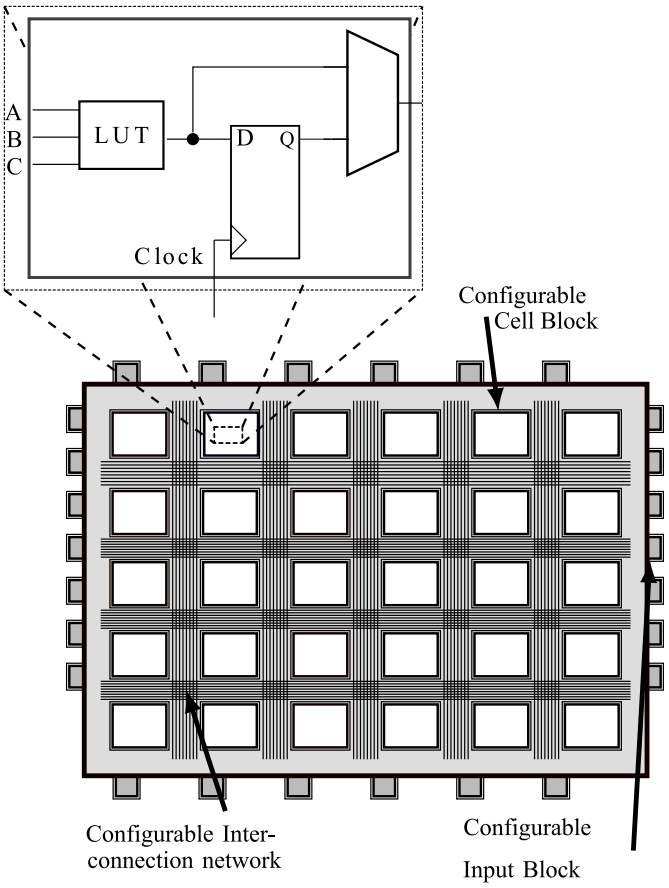

Fig. 5. The general architecture of an FPGA. The high degree of configurability is illustrated, with configurable cell blocks, interconnection networks and input blocks.

high-frequency switching. The general architecture of FPGAs consists of a pre-fabricated array of Configurable Logic Block (CLB), a programmable network of interconnections and configurable input/output blocks as illustrated in Fig. 5. Additionally, the FPGA typically has external or internal memory cells to control the configuration of the CLBs.

The programmability of the interconnection enables configuring the connections between different CLBs of the board, to custom applications. The CLBs consist of simple combinatorial logic, memories, and multiplexers. The I/O blocks of the peripheral I/O architecture of the board implement one or several communication standards. Inside each CLB there are several logic cells, consisting of a configurable Look-up Table (LUT), a D Flip-Flop at the output allowing the register of the cell output, and a multiplexer enabling bypassing the flip-flop. In the zoom-in on the cell, the architecture is illustrated in Fig. 5. Modern FPGA boards consist of hundreds of thousands of CLBs, even millions in some of the larger devices. The LUTs of the CLB are configurable and can be reprogrammed to realize different outputs upon the incoming signal. During design of control logic as FPGA firmware, the bits of the LUT are configured as binaries based on the target truth table. By customizing the index of the LUT according to the designed FPGA firmware can create any combinatorial gate logic.

The digital logical circuits implemented and downloaded to an FPGA have a physical architecture, operate in real-time, and have an inherent parallelism (Monmasson & Cirstea, 2007). Therefore, developing safety-critical FPGA firmware targeted for FPGA is dissimilar to sequentially executed software applications and, during the development, industrial standards require unique testing and management of FPGA-targeted firmware, e.g., embedding control logic in the FPGA-firmware.

3.2. Standards for development of FPGA firmware

A general standard aiming to define functional safety in a complete safety lifecycle with the implementation of Electrical, Electronics, and Programmable Electronics (E/E/PE) is IEC 61508. The standard covers hazards caused by the E/E/PE systems and applies to all E/E/PE systems regardless of application and is therefore applicable for design in many differing industries. The Safety Integrity Level (SIL) is a performance measurement for the safety-critical function defined in the IEC 61508, defining the risk reduction provided by a safety function, or serving as a target level of risk reduction for designing and verifying a safety-critical function. Depending on the target SIL the standard provides specific requirements and activities relating to testing and verification to cover in order to classify the risk reduction achieved by the safety function. IEC 61508 directly covers verification activities relating to FPGA firmware with the FPGA as a target device. In addition, the standard provides a recommended development process for Application Specific Integrated Circuit (ASIC) applications. Due to a manifold of similarities between ASIC and FPGA devices, i.e., sharing common programming languages and proven-in-use verification tools (Kuon & Rose, 2006), the same development process is considered applicable for FPGA firmware.

IEC 61508 recommends following the V-model development flow, which includes activities such as requirements specification, design, configuration management, testing, verification, and validation activities following the V-shaped software development process. Development of FPGA firmware starts by defining FPGA safety requirements and deriving specifications of the board. These requirements serve as the foundation for subsequent development stages. The architecture design ensures that the FPGA can reach the safety requirements and provides the necessary resources for implementing the desired functionality. The behavioral modeling stage aims to capture the high-level behavior of the FPGA firmware, i.e., the application, using modeling techniques such as Simulink. This behavioral model serves as a reference model for subsequent refinement stages.

The module design divides the FPGA firmware into modules, each serving specific functions and interfaces to fulfill safety requirements. Subsequently, the module is transformed, manually or automatically, into a hardware description language (HDL) representation, such as VHDL or Verilog. Automated tools, compliant with IEC 61508, can be employed. A test engineer can utilize the reference model to verify the preservation of the module's functionality after the conversion into HDL. Considering the physical properties of FPGA devices, the languages describing the circuits must achieve certain demands to be functional HDLs. A HDL language useful for implementing digital logical circuits requires a syntax capable of expressing the parallelism of the hardware, sequencing of operations, the delays of the signals, and representations of internal clocks. VHDL (IEEE, 1987) and Verilog (Open Verilog International, 1993), are two such languages. Fig. 6 illustrates the use of a generic controller algorithm as FPGA firmware.

In the early design phase, an FPGA-targeted controller algorithm requires testing at the behavioral level against the specifications. Assuming conformance to the specifications, testing of the controller (i.e., the FPGA-firmware) in HDL form is the subsequent step in the development. In the next step, the engineer refines the HDL to ensure synthesizability, i.e., the description of the circuits is realizable on an FPGA. Non-synthesizable HDL cannot be used to generate a gate-level representation of a digital circuit, however can facilitate simulations and enable verification activities. The FPGA firmware at this state contains expressive descriptions of data transfers in sequential and combinatorial logic, the clocked behavior. A synthesis tool synthesizes the HDL-code and generates a gate-level netlist or, in other terminology unrouted netlist, or post-synthesis netlist. In the unrouted netlist, the configuration of logical elements and interconnections on the target device are described, however, the physical location of the controller is absent. The mapping of each logic element and the wiring architecture

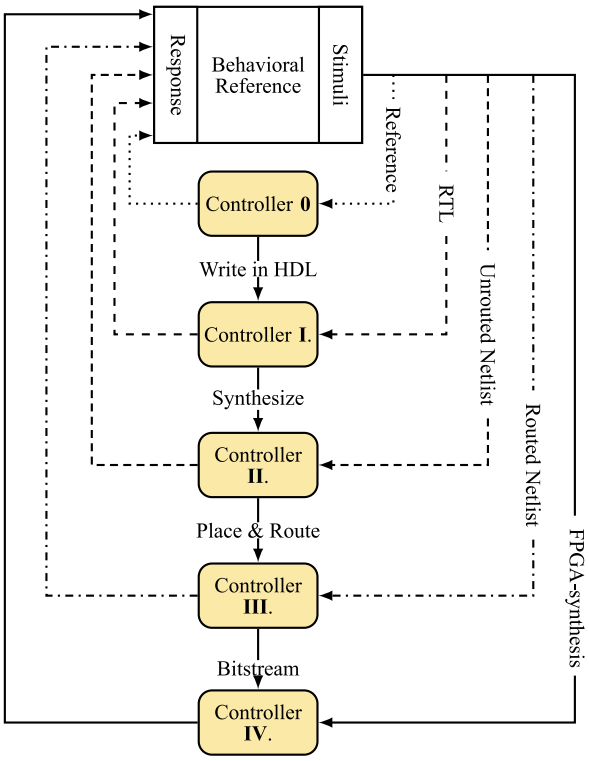

Fig. 6. A general design flow for model based design FPGA projects.

of the FPGA occur in the next step, the place & route step. After the place & route step, the resulting bitstream file of the FPGA firmware can be downloaded to the target device and used to configure the device. At each stage of this development, the behavioral reference can be used to verify that the latter stages of the controller still respond similarly enough to the same stimuli as the behavioral reference. This verification is enabled through simulations of the RTL-level, unrouted gate level-, and layout gate level netlist in a logic simulator (marked by **I-III**). The simulation of the controller receives an opportune stimulus which, according to recommendations in IEC 61508 (2010), should be selected such that a high coverage of the specified function is executed. After deployment to the FPGA the controller (**IV**), now completely implemented in digital logical circuits, receives and responds to a stimulus aiming to cover the execution of the circuit. The FPGA's response is compared with the reference response to verify the implementation's accuracy. Once the requirements are met, testing and verification activities extend to integrating additional modules into the FPGA design. The test engineer then performs validation activities on the FPGA, incorporating the final and complete set of modules.

Developing controllers and embedded functionalities for ASICs and FPGAs involves software and hardware development. FPGAs, although hardware devices, heavily rely on software coding and descriptions in HDL. IEC 61508 emphasizes the importance of clear documentation, traceability, and functional safety assessment throughout the ASIC and FPGA development lifecycle (IEC 61508, 2010). The standard recognizes the need for trustworthy verification and validation activities to ensure compliance with safety requirements and mitigate potential risks associated with ASIC/FPGA functionality. Integrating a digital twin element in the FPGA firmware development, from behavioral models to final logical circuits, can enhance the veracity of SIL claims, improve FPGA firmware development, and provide an up-to-date testing environment for introducing novel controller algorithms or software updates.

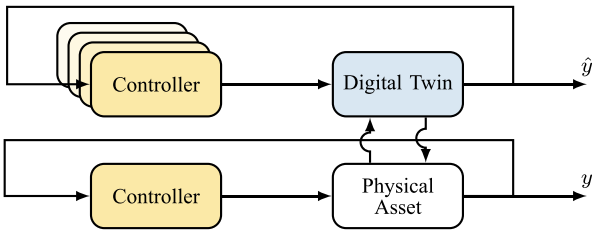

Fig. 7. Optimally, the engineer can connect development versions of the controller to the digital twin and subjecting them to testing to cover the development process.

4. Interfacing strategies of a digital twin and FPGA firmware

Digital twins can automate safety demonstrations by leveraging computational models to capture safety-relevant behavior. In this section, we explain the interfacing issue associated with the framework of digital twins. We identify several feasible interfacing strategies spanning the complete development of FPGA firmware. Finally, we propose criteria for determining the suitability of each interfacing strategy and reach the desires of the framework.

4.1. Interfacing a digital twin and a controller for safety demonstrations

In the proposed framework of digital twins for safety demonstration, the simulation environment plays a crucial role. The interfacing of the digital twin with the controller facilitates the generation of time-series data capturing the expected and safety-related behavior of the physical asset. However, the effectiveness relies on the appropriate interfacing between the controller and the digital twin. Selecting a suitable interface is essential to reach the benefits of the framework. A critical aspect in the framework of digital twins for safety demonstrations is the careful selection of interfacing strategies capable of efficiently covering the complete lifecycle of the physical asset, including the development flow of the logic modules. Improvements, algorithm optimization, or bug removal in the controller, inevitably result in different versions of the logical module. The concept is illustrated in Fig. 7, stacked blocks of the controller represent various versions of the control logic during the refinement in the development process.

4.2. Identifying feasible interfacing strategies

As previously noted, developing FPGA firmware involves multiple development stages. Interfacing each development stage of the FPGA firmware with the digital twin would facilitate testing over the complete development of the firmware. Fig. 8, adapted from IEC61508, illustrates three distinguishable developmental and simulation stages for the recommended development flow. We suggest dividing the development of FPGA firmware into three stages of development. The three stages are denoted in Fig. 8: the behavioral, the descriptive, and the implemented stages. Identifiable differences in strategies for interfacing the FPGA firmware with the digital twin delineate the separate stages. In the behavioral stage, the specifications are used to test a behavioral model of the FPGA firmware, verifying the behavior against the requirements. The first step in the descriptive stage is writing the HDL code, manually or using an automatic tool to generate HDL from a high-level language (Pedroni, 2010; Yankova et al., 2007). The HDL describes the layout, timing specifications, and logical circuits in textual form. In the implemented stage the FPGA firmware is uploaded and configures the logical circuits on a FPGA.

Strategy A utilizes an HDL simulator for co-simulating the digital twin and textual description of the controller during the descriptive stage. Strategy B and C are instead focusing on the latter stages of FPGA firmware development, the implemented stage.

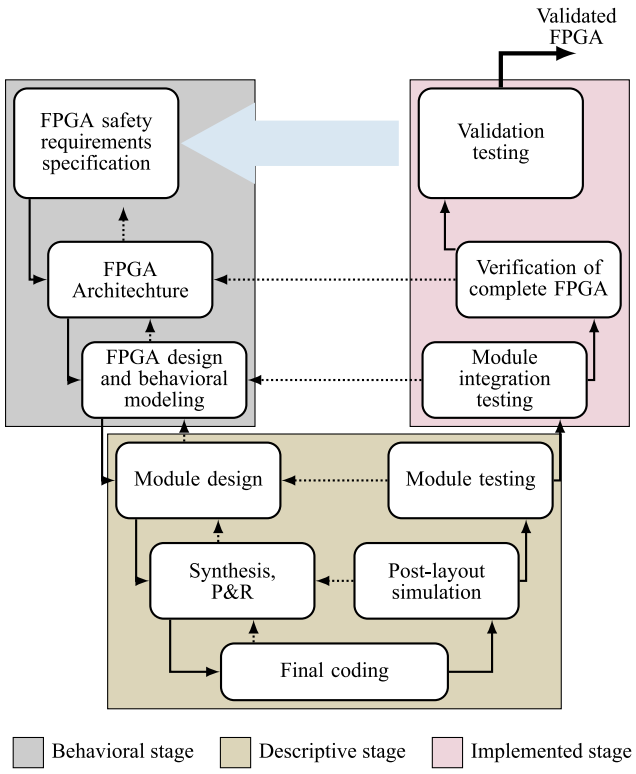

Fig. 8. The adaption, based on the developmental flow of IEC 61508, illustrate the suggestion of dividing the development into three stages; the behavioral stage, the descriptive stage and the implemented stage.

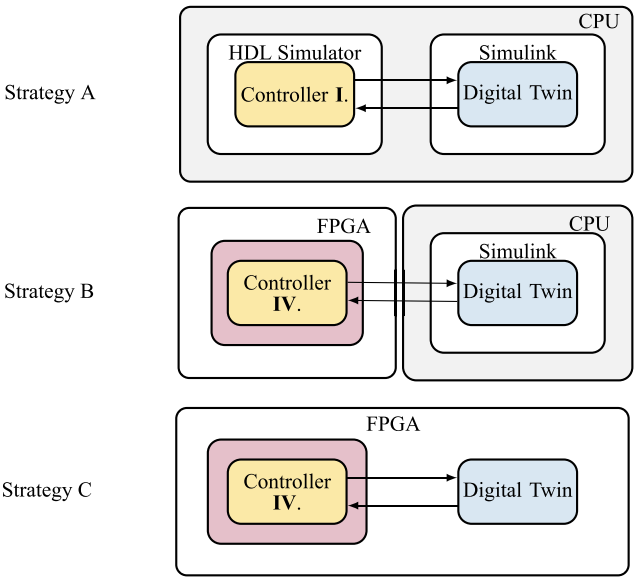

Fig. 9. The three identified strategies cover different stages of the control development and for Strategy B and C the difference is the platform for simulating the behavior of the physical asset.

Fig. 9 illustrates the connections between the digital twin with the controller in the three interfacing strategies.

Strategy A can be classified as a co-simulation strategy, representing the first method for interfacing the digital twin. The co-simulation executes a solver for the digital twin in a simulation environment such as Simulink, interfacing with the HDL simulator simulating the FPGA firmware.

Commercial tools can connect HDL simulators with simulation environments for the underlying such as Simulink. The approach to interfacing the two simulation environments can either be local and single-system, or network-based. The single system integrates a secondary simulation tool into the primary simulation tool, diverting timing and synchronization control to the primary simulation tool. The network-based approach requires a physical interface to link a data exchange between the simulation environments, facilitating the timing and synchronization to the linking tool. The network-based approach method is the more scalable of the approaches.

Strategy C involves implementing the digital twin on an FPGA. This approach requires converting the mathematical models into a suitable HDL, followed by synthesis. Constraints and requirements of FPGA firmware govern the development of the underlying models of the digital twin, such as fixed-number representations and timing designs. The digital twin then undergoes a similar process as the controller depicted in Fig. 6. Once the digital twin is able for FPGA implementation, placing the digital logical circuits capturing the behavior of the digital twin can either be realized on a separate FPGA device or on the same FPGA device as the FPGA-firmware. By implementing the digital twin on the same FPGA as the FPGA firmware, the digital twin, the FPGA firmware and the interface are completely hardware-based. There is also the possibility to implement the digital twin on a separate FPGA board and enable communication with other FPGAs through various interfaces, such as serial or parallel buses, Ethernet, or custom protocols.

4.3. Criteria for selecting suitable interface strategies

Efficiently selecting interfacing strategies is critical for a digital twin-based framework in safety demonstrations. The digital twin enhances testing by improving computational speed, scalability, ease of setup, and remote testing capabilities. It offers accurate and up-to-date evidence generation and enables software update safety demonstrations. The primary objective of the evaluation criteria is to preserve the advantages available with the framework of a digital twin for safety demonstrations.

Criterion 1: Computational speed. To sufficiently demonstrate the safety, extensive testing is required. Prioritizing a high computational speed of the simulation environment is important to improve safety demonstrations.

Criterion 2: Configurability. An effective interfacing strategy requires a high degree of configurability for test case parameters and operational conditions to facilitate comprehensive testing.

Criterion 3: Scalability. The framework of digital twins for safety demonstrations shall cover all the components in the all-electric actuation system. This extends to integrating multiple components and FPGA firmware modules.

Criterion 4: Usability. The interfacing strategy should be user-friendly, allowing engineers from diverse backgrounds to easily handle the digital twin for safety demonstrations without specialized expertise.

Criterion 5: Manageability. The criteria of limited effort and resources for implementing the interfacing strategy assesses the ease of managing and overcoming implementation and operational challenges.

Optimizing all these criteria are improbable likely requiring a trade-off.

5. A case study for selecting suitable interfacing strategies

The selection of an interfacing strategy goes beyond considering the type of strategy. In this section, we present general interfacing considerations and the challenges of defining the interfacing boundary and determining relevant components for the digital twin. We present a case study to illustrate these concepts, highlighting the differences between the virtual and physical domains and their implications on interfacing. We then evaluate each feasible interfacing strategy individually and finally directly assess them based on the proposed performance criteria.

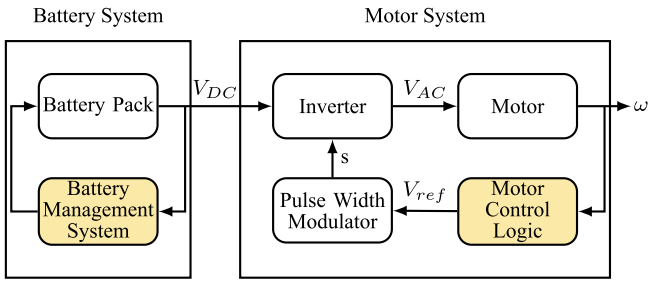

Fig. 10. The battery and the motor system are critical components in the capability of the all-electric actuation system to reach the safe state.

5.1. General interfacing considerations for safety demonstrations

The case study for selecting suitable interfacing strategies centers on a critical part of the all-electric actuation system; the battery system and the motor system. A simplified block diagram of the motor system and the battery system of the case study is illustrated in Fig. 10.

The specific version of the controller is not a significant factor in the initial general considerations of interfacing and is therefore not illustrated. effective management of the battery pack is crucial for maintaining sufficient energy reserves. The Battery Management System (BMS) plays a vital role in internal health management, optimizing charge and discharge cycles, and ensuring the overall functional safety of the system, including the ability to close the valve even under abnormal or malfunctioning conditions. The battery system is an illustrative example of the test case parameters of the framework. Model parameters describe specific properties of the battery pack, such as diffusion rate coefficients and internal cell resistances. Fault parameters encompass parameters related to faults, such as internal short circuits in specific cells. Logic parameters include factors associated with logic issues, such as data packet losses between the BMS and other control modules.

Establishing a useful interface between the digital twin and the BMS logic requires defining the boundary between the controller logic and the digital twin. The primary goal of cell balancing, whether passive or active, is to equalize the voltages across cells to prevent rapid aging or thermal runaway. Regardless of the cell balancing strategy, the methods are considered to be part of the BMS as illustrated in Fig. 11.

Defining the interfacing boundary between “hardware” components (e.g., resistors and transistors) and “software” elements (e.g., the cell balancing algorithm) when interfacing the digital twin with the BMS is critical. Including the cell balancing circuits and cell measurement circuits as test case parameters facilitates testing the system in the presence of faults and degradation in the balancing circuitry and variations in measurement accuracy.

The selection of safety-critical hardware sub-components to include in the digital twin can be exemplified by the motor system. The inverter plays a vital role in converting the DC voltage from the battery system into a 3-phase AC voltage for the motor. By employing a pulse width modulator (PWM) signal to control the width of electrical signals applied to the transistors in the inverter circuit, the motor controller can realize the required 3-phase AC voltage to the motor. The inverter and the PWM are essential for generating the AC voltage in the physical domain. Bypassing the inverter and directly accessing the motor models of the digital twin mitigates the computational cost stemming from the intense switching frequencies controlling the inverter. In addition to improving the computational speed, bypassing the inverter can simplify integration testing of multiple components and improve scalability.

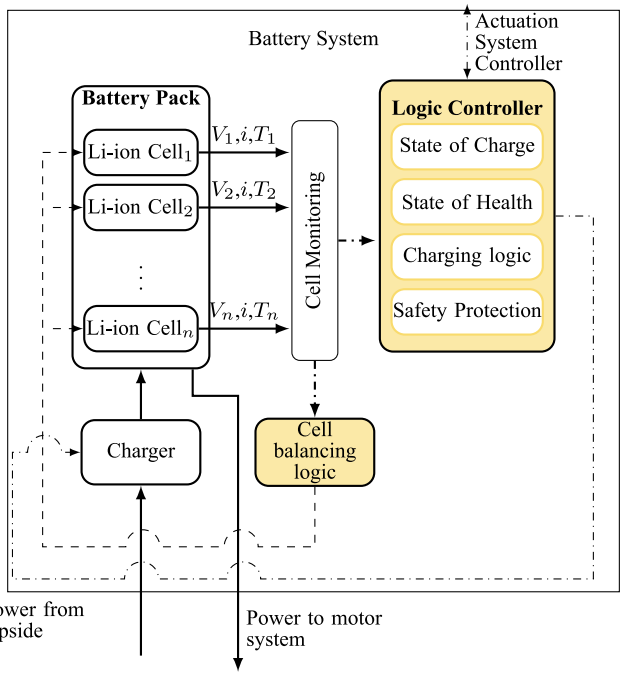

Fig. 11. The battery system illustrates the blurred boundaries of testing logical modules that also contain hardware actuators.

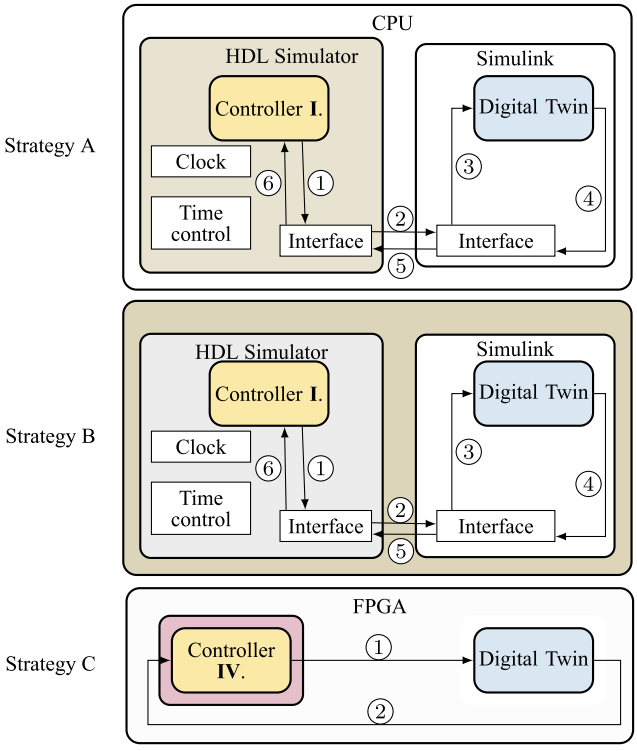

Fig. 12. The requirements for digital twin to controller inter-communication varies with the interfacing strategies.

5.2. Evaluating the strategies

Fig. 12 illustrates the timing and steps required to realize implementations of the interfacing strategies for strategy A, B and C.

Strategy A interfaces the digital twin with the controller at the descriptive level (I-III), i.e., the controller implemented as HDL. Since

HDL aims to describe physical hardware, including timing characteristics, co-simulation requires emulating the behavior of an actual digital logic circuit to account for the timing, signal delays, and sequencing of operations. Commercial tools such as Modelsim enable this interfacing solution by providing an interface that exchanges data, at fixed time intervals, between the HDL-emulator and the models, for example, implemented in the Simulink environment. Inherently, these solutions are dependent on the sequential exchange of information between the simulators.

- The gate-level description of the controller would be simulated in Modelsim enabling emulation of the behavior of the actual digital logic circuit and the ability to account for the timing, signal delays, and sequencing of operations (1). The controller generates the signals to drive the motor.
- The motor model, in Simulink, is connected to the gate-level motor controller simulated in Modelsim through a network connection. The connection uses a link for data exchange, to configure the timing and exchange (2).
- The gate-level motor controller sends the control signals to the motor model, which updates the motor's behavior and sends the sensor data back to the gate-level motor controller in Simulink (3–4).
- The gate-level motor controller in Simulink receives the sensor data and updates its control signals accordingly (5–6).

The process of co-simulation connects the gate-level netlist of the motor controller, with a motor model in Simulink. The two models communicate with each other at fixed time intervals, allowing the simulation to account for the timing characteristics of the hardware and the physical behavior of the motor. This process allows the user to observe the behavior of the motor and controller over time, and use this information for safety demonstration and other evaluations.

Commercial tools like Matlab offer local simulation capabilities, but their sequential execution limits performance for safety demonstrations. Network-based approaches provide better scalability. Power electronics research focuses on co-simulation techniques, combining FPGA firmware described in HDL with physical asset models. This enables the emulation of digital logic circuits, accounting for timing, delays, and sequencing. These approaches enhance the design and development of power electronic systems. One interesting example of this is the design of a permanent magnet synchronous motor including an inverter. The components are designed according to first principles and co-simulated with a controller, described in VHDL, which generates PWM signals to the inverter (Jiang et al., 2005). This experimental setup allows the authors to develop and simulate control strategies implemented in VHDL without damaging test equipment. Another approach focuses on co-simulation of a synchronous buck converter, with a digital controller implemented in VHDL (Zumel et al., 2010). The authors connect the model created in PSIM, a tool for designing electronic circuits with Modelsim, running the synchronizations inside Modelsim. Strategy B interfaces the digital twin, realized in Simulink, with the controller implemented on the FPGA. Despite the implementation of the FPGA firmware as a logical circuit on the FPGA, the information exchange is similar to the co-simulation approach in strategy A. The main simplification between strategy A and strategy B is the logical circuit not longer requiring simulation, due to the hardware implementation.

Strategy B requires an interface between the FPGA and the computer with the digital twin, and the communication exchange occurs at fixed time steps.

- The logical circuit generates a command, and the peripheral I/O utilizes a communication interface to send the control signal to the receiving interface of the PC-host (1–2).
- The control signal serves as an input to the actuators modeled as part of the digital twin, and the models generate a response based on simulating the digital twin (3–4).
- The signal is returned to the interface on the FPGA and used in the logical circuit to generate a response to the information about the state of the digital twin (5–6).
- The process repeats for the next simulation step.

Strategy B is expected to achieve faster computations inside the control algorithm and can improve the total iteration time of a simulation compared with the co-simulation. The implementation of the FPGA firmware in this stage enables scalability and the inclusion of several modules on the board (subject to the hardware restrictions of the board). The computational speed of the control algorithms benefits from the inherent hardware-based parallelism. However, the same hardware that accelerates the generation of commands from the FPGA firmware increases debugging complexity (Zkadek et al., 2015). In Zkadek et al. (2015), the authors propose a PCIe interface for speeding up Strategy B, including a dynamic debugging tool for debugging the RTL. While the experimental test shows a modest improvement of the transmission time compared with an Ethernet cable, the authors argue that overcoming the limitations of third-party internal data structure can greatly improve the concept (Zkadek et al., 2015).

To realize the FPGA-based digital twin approach, a digital twin designer can either implement the digital twin element on a shared FPGA, i.e., the same FPGA as the controller, or on a separate FPGA.

The option of placing the digital twin on a separate FPGA and connecting the boards will constrain the communication by the transmission capabilities between FPGA boards. One example utilizing strategy C realizes advanced analysis of a motor used for propulsion of electric vehicles and simulations in real-time (Ruba et al., 2016). The application also features an interface for communicating the simulated data to a graphical user interface on a laptop.

An early proposal of FPGA-based real-time simulators evaluates the characteristics of the strategy in relation to power electronic components (Matar & Iravani, 2009). The authors show that by implementing models of converters onto an FPGA, the simulations can handle very small time-steps (< 500ns) and scalability (virtually no extra simulation time despite increasing the total number of simulated converters while tracking the reference model satisfactorily. In Darba et al. (2012), the authors implement a electrical motor model onto the same FPGA as the digital control algorithm. The authors achieve significantly faster testing due to the integration strategy (which enables the block only at a necessary sample rate) and the absence of an analog-digital conversion process. To mitigate the risk of damaging equipment during the testing of different control algorithms (Ruba et al., 2019) suggest an FPGA-based digital twin twinning the behavior of the testbench . The authors design a digital twin of the testbench according to first principles, with experimental identification of key modeling parameters, with real-time simulation, and a plug-and-play capability of the FPGA-based motor controller. Another implementation is the application of an FPGA-based digital twin for monitoring and diagnosing faults in a power electronic transformer (Xiong et al., 2022). The authors use the digital twin as a reference model to enable fault diagnostics through comparing expected and actual responses during fault injections into the physical asset. A similar utilization of an FPGA-based digital twin as a real-time reference for fault diagnosis is the proposal to include probabilistic models to account for the uncertainty of the modeling phase and in the operational environment (Milton et al., 2020). The motivating factor for selecting an FPGA-based digital twin in these papers are the inherent parallelism and low latency of the computations and provides the ability of hardware acceleration of the simulation (Milton et al., 2020; Xiong et al., 2022). A network of FPGA-based models for simulating an energy conversion system has been researched (Milton et al., 2019). In the article, the authors demonstrate a small trade-off in performance when connecting mathematical models implemented on multiple FPGA devices and connecting the FPGAs to perform simulations on a larger scale.

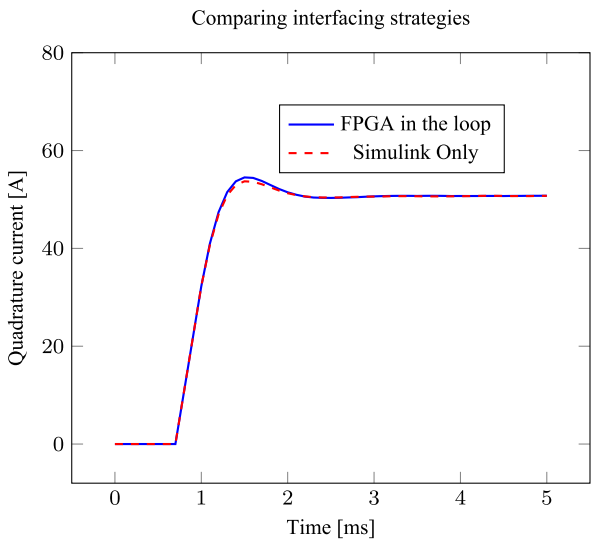

Fig. 13. The simulation results, comparison of FPGA in the loop (Strategy B) and (Strategy C).

5.3. Criteria evaluation

This subsection delves into the evaluation of three interfacing strategies—Strategy A, Strategy B, and Strategy C—based on predefined criteria functioning as benchmarks for assessing the individual performances of the strategies.

5.3.1. Criterion 1: Computational speed

Strategy A is the only suggested strategy that interfaces the descriptive level of the FPGA firmware. The literature and reasoning about the loss of the parallelism of the algorithm indicate a significant loss in the computational speed. Simulating descriptions of logical circuits is the slowest of all the strategies due to the loss of parallelism of the target platform. Additionally, the bi-directional information flow between the HDL simulator and the simulator environment executing the digital twin requires the simulators to hold. Using an FPGA for the FPGA firmware, as described in strategy B, not only allows the testing of the final version of the FPGA firmware but also accelerates the execution time. However, the device still requires an interface for communicating with a digital twin in a simulation environment, which introduces synchronization issues since the FPGA operates in real-time, as compared to the environment that executes simulations of the digital twin. The maturity of verification using an FPGA-in-the-loop includes the availability of commercial tools for overcoming synchronization problems. Similarly to strategy A, the commercial tools focus on verifying the consistency of the FPGA firmware against a reference behavior. The other alternative is to implement an FPGA-based digital twin element on the same device or interface two different FPGAs. The literature indicates that strategy C provide the fastest simulation iteration, due to interfacing real-time operating device or directly interfacing the logical circuit that realizes the FPGA firmware with the logical circuit realizing the behavior of the digital twin.

The computational cost of the various interfacing strategies involves experimenting by injecting a step change in the controller to compare the computational speed. A behavioral model of the motor interfacing with the motor controller responding to a step change can be seen in Fig. 13 for the motor system completely in Simulink and Strategy B.

Strategy A is omitted from Fig. 13 since the extractable data is the response from the simulation model rather than the controller. The time for simulating one iteration links directly to the computational speed criteria. The simulation results indicate significant variations in computational time across the different strategies. The behavioral approach exhibits a computational time of 89.641 s. In contrast, Strategy A requires substantially more time, totaling 405.360 s. Strategy B demonstrates the most efficient computational speed, completing the simulation in only 8.390 s.

One contributing factor in the increase of computational speed in strategy B is due to oversampling on the FPGA. The FPGA steps 250 times for a single simulation time-step, thus requiring substantially fewer simulation steps of the motor model in Simulink. However, while this sampling methodology can speed up the time for each test case, executing the simulation of the motor model in Simulink appears to be a significant limitation. In a separate experiment, the same test case on a busy processor shows significant increases in the computational time, despite a doubled oversampling. The behavioral simulation requires 139.823 s on a busy CPU. Strategy B completes the simulation in 10.605 s. The computational speed drop indicates that the digital twin running on the CPU is a bottleneck in the approaches.

Strategy C is absent from the experimental comparison due to requiring adjustments to the digital twin development to satisfy the constraints, fixed point representation, and timing specifications, stemming from requirements on FPGA firmware. Additionally, including the FPGA-based digital twin can be superfluous since FPGAs have realized real-time applications since the introduction of the technology. Based on the evaluation and literature, strategy C is the most effective among the three strategies for reaching desirable computational speeds. Strategy B is second in computational speed, while Strategy A significantly lags behind in this criterion.

5.3.2. Criterion 2: Configurability

Strategies A and B, which utilize Simulink for the digital twin, offer high accessibility for configuring test case parameters. However, configuring an FPGA-based digital twin in strategy C is more complex. Adjusting test case parameters either requires reconfiguring the FPGA or preparing the FPGA to accommodate parameter adjustments. Strategy C requires initial configuration involving integrating the ability to perform parameter adjustments into the FPGA device. Strategy C, which involves an FPGA-based digital twin, is the only strategy requiring a specific approach for parameter configuration. Relatively to each other, strategies A and B outperform strategy C in the configurability of the test case parameters.

5.3.3. Criterion 3: Scalability

Strategy A facilitates the connection of multiple modules of FPGA firmware in an HDL simulator, but executing HDL on a processor is not optimal. Connecting multiple modules of FPGA firmware to digital twins of multiple components increases the computational cost and introduces scalability issues due to executing increasingly extensive HDL. The data transfer limitations can significantly impact the scalability of strategy B, as discussed in previous studies (Zkadek et al., 2015). Strategy C, on the other hand, offers scalability by the possibility to use separate FPGA devices for the FPGA firmware and the FPGA-based digital twins, facilitating real-time simulations and testing of multiple modules of safety-critical FPGA firmware, applied in the plug-and-play approach of an FPGA-based digital twin modeling the behavior of a motor and connecting to FPGA application of the motor controller. Multiple connected FPGAs in the implementation of real-time FPGA simulation show no significant scalability issues and applications and are available in various domains (Matar & Iravani, 2009; Milton et al., 2019; Xiong et al., 2022). Testing novel control strategies for safety demonstrations can be done topside using an experimental setup with multiple FPGA modules. This approach offers reusability and flexibility for testing and validating software in a controlled environment. Strategy C is expected to be outperforming in scalability, due to the possibility of testing multiple modules of safety-critical FPGA firmware with multiple FPGA-based digital twins.

5.3.4. Criterion 4: Usability

In the researched literature, the usability is not distinguishable between the various interfacing strategies. Commercial tools exist for interfacing with a digital twin with the FPGA firmware, facilitating the implementation of strategy A and strategy B. However, for strategy A, the commercial tools focus on using stimuli to evaluate the behavior, and additional software is required to interface the digital twin in Simulink with the controller. Increasing the oversampling in strategy B was easy with the available tools. The usability of strategy C, in comparison to the other strategies, is predominantly impacted by the limitations of the FPGA board when implementing the digital twin alongside the control systems. The limited space on the FPGA device restricts the number of test cases, allowing only a limited set of test case parameters. To extend test case coverage, the user requires a reprogramming file of the board. Strategy B outperforms the other strategies due to the availability of current solutions.

5.3.5. Criterion 5: Manageability

For strategy A, most commercial tools focus on injecting stimuli into the HDL simulator and comparing the response against the reference model. However, strategy A offers the advantage of configuring test case parameters using Matlab and Simulink without requiring additional resources or user interfaces. Challenges in extending simulation time frames and managing dynamic interactions between the HDL simulator and the digital twin simulator require research efforts. Strategy B provides a straightforward solution by connecting an external test suite to inject test cases into the digital twin element in Simulink. This approach requires no additional steps to manage the framework of digital twins for safety demonstrations. In contrast, Strategy C, which involves implementing an FPGA-based digital twin, presents managerial considerations. It requires the development of an interface for data communication with the FPGA device. The literature on FPGA-based digital twins indicates a need for a custom graphical user interface for data collection, sampling, and writing operational scenarios. To fully utilize Strategy C, a user interface that allows executing and collecting data from multiple test scenarios without reprogramming the device for each case is crucial.

Strategy B excels in manageability due to the availability of tools that directly connect the digital twin in Simulink to the FPGA firmware, including sampling options. Strategy A demonstrates satisfactory performance but has challenges related to simulation time frame extension and dynamic interactions. Strategy C requires additional resources and efforts, placing it behind the other strategies in this criterion.

6. Discussion

Digital twins can add the benefits of simulation-based testing activities and augment the approach with a unique one-to-one twinning and a complete lifecycle perspective. Simulatable and physics-based models in the digital twin capture the safety-related behavior of the hardware components. This paper identifies values and benefits reachable by a digital twin-based framework for safety demonstrations. The framework of digital twins for safety demonstrations interfaces controllers and logical modules to demonstrate the capability of the system to reach a safe state in various scenarios. By adjusting test case parameters for simulating degradation, faults, and communication in the digital twin, the framework seeks to verify the system's behavior against safety requirements. To maximize the framework utility selecting a suitable interface between the digital twin and the controller is critical. This paper centers the research on suitable interfacing strategies for controllers targeted for FPGAs.

The paper researches the fundamentals of FPGAs and emphasizes the development of controllers for the devices. Digital logical circuits realize the controller logic as hardware, while a text-based software language describes the circuits during development. Digital twin-based testing can cover the entire development of this software–hardware

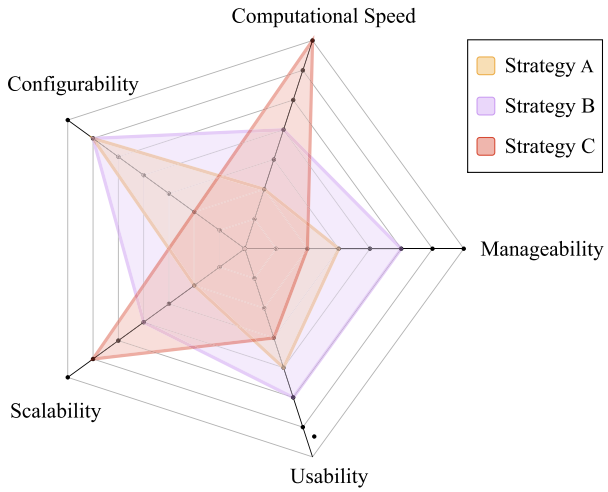

Fig. 14. The performance to each criteria for all the identified strategies.

duality of developing FPGA firmware. The paper proposes dividing the development flow recommendations from IEC 61508 into three manageable and distinctly different stages. The first stage consists of behavioral models of the controller, and interfacing the controller at this stage is elementary and out of scope for this article.

The first interfacing strategy, strategy A, covers the FPGA firmware, i.e., the controller, as a hardware description language of the logical circuits. The other strategies implement the controller on a FPGA device, but strategy B realizes the digital twin in Simulink, and strategy C realizes the digital twin on an FPGA. For opportunely selecting one or several suitable interfacing strategies, we introduce five performance criteria that aim to preserve the values of the framework: computational speed, configurability, scalability, usability, and manageability. The interfacing strategy will affect all the criteria, and maximizing the performance criteria will facilitate the framework of digital twins for safety demonstrations, as mentioned in the section on digital twins.

The case study highlights the importance of defining the boundary between the digital twin and the controller by using cell balancing to illustrate the issue. Encompassing the hardware actuators and the measurement circuits in the digital twin extends the test case parameters to increase the test coverage. Including and adjusting the values on model parameters describing the physical properties of the resistors used to bypass current from the cells during the charging can test systems behavior during varying or degraded resistors. Additionally, the case study covers the opportunity of the framework to bypass components with non-existing to low safety implications during integration testing of FPGA firmware. The inverter is computationally expensive due to the high-frequency requirement for realizing a 3-phase AC voltage in the physical domain, and unit testing the motor controller with a digital twin covering the behavior of the inverter can be sufficient.

Literature and simulations indicate significant differences in the strategies' performance to the criteria. The case study highlights the perception of the relative performance of the three interfacing strategies. Fig. 14 visualizes the perception of the criterion performances of the interfacing strategies.

As illustrated in the radar chart, not one of the interfacing strategies excels at every criterion. The framework of using the digital twin for safety demonstrations influences the importance of the criteria. The computational speed reachable with strategy C through implementing the digital twin on the FPGA would be a significant advantage and support the final validation of the FPGA firmware. The results additionally show that strategy C outperforms strategy B in scalability, a relative performance supported by the simulations and the literature. Literature evidence shows several implementations and applications of

Table 1
The expected performance of the strategies in relation to derived criteria.

	Strategy A: Co-simulation	Strategy B: FPGA-in-the-loop	Strategy C: FPGA-based digital twin
Criterion 1: Computational speed	Requires a HDL simulator, losing the real-time of FPGA firmware. Logical circuits are inherently parallel which is lost in HDL simulators.	Vast improvements in computational speed, experiments indicate the simulating the digital twin elements in Simulink as limiting factor	The FPGA-based digital twin would interface the controller directly on the hardware. Existing implementations show real-time capabilities.
Criterion 2: Configurability	The configurability of the co-simulation solution would only require the application developer to re-configure the RTL. Due to the capabilities of modern tools, this strategy would work well.	The configurability of the digital twin in strategy B is the same process as for strategy A.	Potential improvements of adjusting parameters, current solutions limiting.
Criterion 3: Scalability	Possible to add multiple modules of FPGA firmware. The scalability is limited by the processing power of the HDL simulators	FPGA firmware modules can share a FPGA, limited by the available circuits on the FPGA. The transfer of information is expected to be a bottleneck	Limited by the available circuits on the FPGA. Multiple FPGA-devices can be connected. Capability to integrate modules of FPGA firmware and digital twin elements
Criterion 4: Usability	Commercial tools exist for co-simulation of FPGA firmware	Commercial tools are available for interfacing Simulink with FPGAs.	Data collection is limited
Criterion 5: Manageability	Tools center on stimuli and comparing response, requires more research to interface a digital twin capable of safety demonstrations	No additional resources needed, select sampling time and increase time frames	Design a testing environment for extensive test coverage, customized user interface, sampling time

digital twins or simulation models realized on FPGA devices interfacing with other FPGA modules in real time. The simulations in the case study show a significant decrease in computational speed for strategy B with a busy processor, indicating the simulations of, and communication with, the digital twin on the processor as the bottleneck. Including several digital twin elements and integration testing of multiple controllers will require additional communication and increase the number of computations on the processor. Strategy A shares the scalability challenges with strategy B, while strategy C can overcome these challenges.

Strategy C underperforms relative to the other strategies in configurability due to a non-obvious solution to adjust test case parameters after implementing the digital twin as a digital logical circuit. Configuring the test case parameters is of paramount importance to realize the framework. However, further research could explore capabilities to store multiple test cases on the FPGA, utilizing the memory capabilities of modern FPGAs. Creating a custom soft processor capable of realizing multiple test cases without reprogramming the FPGA is an approach to reaching a higher configurability score. The potential to improve the configurability of strategy C indirectly influences the remaining criteria, usability, and manageability. The usability for engineers of various backgrounds can benefit from a custom test case interface to the FPGA-based digital twin. At the same time, both strategy A and B facilitates a commercial and well-documented user interface, supporting the usability of the strategies. The necessity to create a user interface for writing and reading from the FPGA-based digital twin in strategy C stresses the last criteria.

To fully implement strategy C, the results from the test case emphasize several practical implications on manageability. Developing an FPGA-based digital twin requires adhering to the constraints of the digital logic circuits. Fixed-point representations of the governing model parameters and equations and specifications of timing considerations are two constraints stemming from the practical implications of strategy C. Realizing the interfacing strategy requires significant resources and effort relative to the other strategies.

Equally noteworthy are the additional resources and effort for managing strategy A. In literature and applications employing co-simulation of HDL, the research centers on injecting a stimulus to verify the controller response against a reference behavioral model. The case study shows applications reaching sufficient code coverage by injecting a stimulus and comparing the response to a reference model. The key findings from the case study per the criteria have been summarized into Table 1.

Higher-level test cases are the objective for the framework utilizing digital twins for safety demonstrations. An illustrative example of this is the motor controller; the framework aims to cover a safety-critical scenario where a stuck valve suddenly becomes unstuck. In this scenario, an aspect of demonstrating the all-electric actuation system's behavior is examining how the motor system responds to the situation. A foreseeable control action is that the motor controller increases the torque to the rotational-to-linear actuator by revving up the motor. If suddenly the valve becomes unstuck, by a rapid decrease of frictional forces, the motor may ram the valve into the pipe walls, causing significant damage.

The literature indicates inadequate support in co-simulation tools of strategy A for these high-level test cases. Additionally, the improvements of interfacing a digital twin with the descriptive level, compared to injecting stimuli, are not evident. IEC5108 recommends the existing stimuli-based verification method. In an interfacing strategy covering the complete development of FPGA firmware, behavioral stage simulations only in Simulink are assumed to be elementary. Assuming an interfacing strategy capable of covering safety demonstrations in the first FPGA firmware development stage, i.e., the behavioral stage, and the last development stage, i.e., the implemented stage, demonstrations of the safety of the behavior with stimulus approach in the descriptive stage are sufficient. Furthermore, we believe that the advantages of strategy C make it more suitable for safety demonstrations of modules implemented on the FPGA. The performance, as evidenced by the case study, of key criteria, i.e., the computational speed the scalability, and the potential to overcome remaining challenges, leads to a recommendation of interface strategy C.

Bypassing the inverter in the digital twin environment could improve the scalability of system-level testing and safety demonstrations of the digital twin elements. Assuming the motor system has demonstrated safety sufficiently and supported the claim about safety at the modular level, integration testing by simplifying components, like the inverter can improve the framework. Modifying the FPGA firmware to operate in a digital twin mode, facilitating a bypass of computationally expensive components is an approach to reach the computational speeds. This approach enhances the scalability of all strategies.

At the behavioral stage of the FPGA firmware, the digital twin allows for configurations of the internal parameters. Test cases can generate evidence on the system behavior. The test cases range from broken or drifting sensors to degraded or completely broken components, i.e., the battery cells, the motor, etc. An opportune amount of cases can sufficiently cover safety-critical scenarios, and compounding the results can demonstrate the capability of the all-electric actuation system to reach a safe state. Upon reaching a sufficient test coverage, satisfying demonstrations at the behavioral stage can facilitate refinements of the FPGA firmware into HDL. In the descriptive stage we recommend generating stimuli for verifying the response against a reference model. A bitfile configures the FPGA and realizes the controller as a logical circuit. At the implementation stage, the recommended interfacing strategy requires synthesizing the digital twin, following the same procedure as the controller. At this stage, the FPGA-based digital twin interfaces directly with the controller, simulating the system's behavior and generating evidence of its safety. Moreover, the FPGA-based digital twin facilitates validating the implementation of the safety-critical FPGA firmware. Expectations include the hardware-accelerated computations and enhanced integration testing capabilities for multiple modules of FPGA firmware in safety demonstrations. A modification of Fig. 6 illustrates the full concept, highlighting the testing capabilities in Fig. 15.

The proposed development flow consists of fully testing the behavior of the controllers harnessed to a digital twin in Simulink. The simulations can demonstrate the safety and as the basis for the reference model. Stimuli can be used to test the FPGA firmware during HDL development, allowing for an evaluation of the response at each step. Finally, the implemented FPGA firmware can be tested in real-time by the FPGA-based digital twin which has been synthesized and placed and is realized as a logical circuit on an FPGA device.

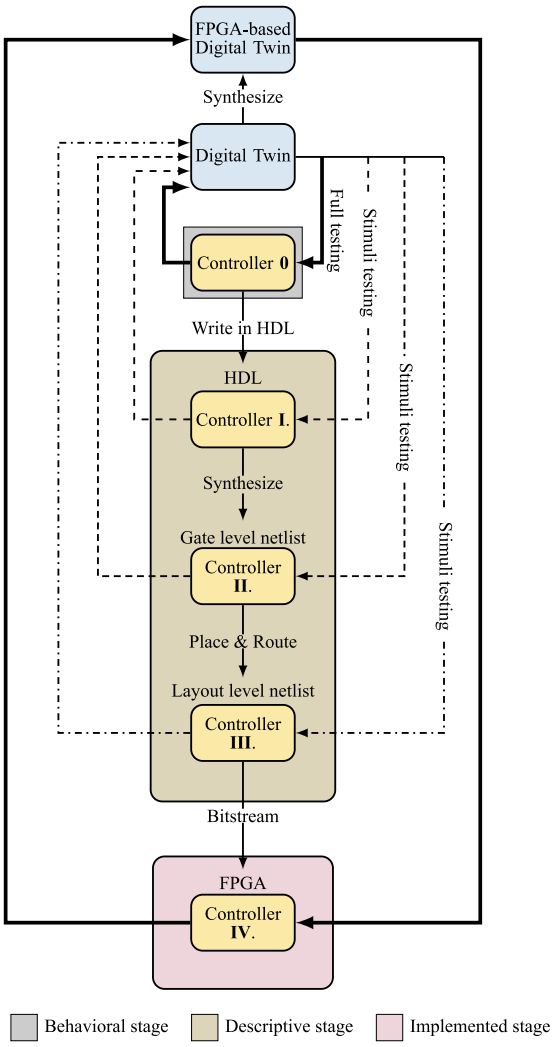

Fig. 15. A proposed flow for developing and testing FPGA firmware. The flow provides interfacing solutions to provide safety demonstrations for simulation-based verification and validation activities.

7. Conclusion

This research paper identifies three feasible interfacing strategies for connecting a digital twin with safety-critical functions embedded in FPGA circuits, capable of demonstrating safety-related behavior in the latter two identified stages of developing controllers as FPGA firmware. The first stage of creating an FPGA-based controller is out of scope due to simplicity and existing solutions for interfacing, and already a part of the framework of digital twins for safety demonstrations. The two latter stages of FPGA firmware development encompass software and hardware aspects, necessitating a multi-faceted approach to development and testing.

The introduction of five performance criteria aiming to maximize the values of the digital twin supports our recommendation for digital twin-based testing during the design and operational stage of the case study. We propose behavioral simulations to cover the behavioral stage and to utilize existing commercial tools to verify behavioral preservation during all synthesizing steps in the descriptive stage. For the final development stage, we recommend an FPGA-based digital twin for reaching high computational speed, scaling up to integration testing of multiple modules, and cover testing of the digital logical circuits realizing the controller on the FPGA device.

The practical implications of the recommendations require the adherence of the digital twin to the constraints of configuring FPGAs and creating custom resources for reading, writing, and sampling test cases to the FPGA-based digital twin. Overcoming the challenges of usability and configurability requires additional efforts and resources, but the identifiable improvements of the recommendation outweighs the challenges.

7.1. Further work

The recommendation of the paper is to use an FPGA-based digital twin for realizing safety demonstrations. Creating a platform suitable for late-stage testing and validation of FPGA firmware supports the framework of safety demonstrations using a digital twin. To incorporate a digital twin element onto an FPGA device, adhering to the requirements and constraints for developing a synthesizable digital twin is necessary. However, further research on an assurance and qualification process for the FPGA-based digital twin is a requirement for applying the digital twin to generate evidence about the system's safety-related behavior. Additionally, the case study provides valuable insights into the advantages of conducting safety demonstrations in a virtual domain. Bypassing computationally expensive components, e.g., the 3-phase inverter, improves the computational speed and scalability of the safety demonstrations. However, this would require modifications to the development of FPGA firmware to enable communication with separate interfaces and bypass the physical components. Creating a digital twin mode within the FPGA firmware would enhance the overall framework, offering additional benefits for safety demonstrations. Exploring continuous safety demonstrations through an online version of the digital twin, which interfaces with the operational logic module, requires researching the reliability of the interface to ensure safe operation.

CRediT authorship contribution statement

Ludvig Björklund: Conceptualization, Methodology, Writing – original draft, Visualization. **Johannes Schick:** Validation. **Mary Ann Lundteigen:** Writing – review & editing. **Markus Glaser:** Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgments

This work was carried out as a part of SUBPRO, a Research-based Innovation Centre within Subsea Production and Processing. The authors gratefully acknowledge the project support from SUBPRO (grant number 237893), which is financed by the Research Council of Norway, major industry partners and NTNU. Additional acknowledgment goes out to the team working on the ISSA project at the Institute for High Integrity Mechatronic Systems, Aalen University, for access to the case study.

References

Agnusdei, G. P., Elia, V., & Gnoni, M. G. (2021). A classification proposal of digital twin applications in the safety domain. *Computers & Industrial Engineering*, *154*, Article 107137. http://dx.doi.org/10.1016/j.cie.2021.107137.

Björklund, L., Glaser, M., Imle, S., Skofteland, G., & Lundteigen, M. (2022). Design of a digital twin of gate valves for partial stroke testing. In *Proceedings of the 32nd European safety and reliability conference* (pp. 3451–3458). Research Publishing Services.

Bönsch, J., Elstermann, M., Kimmig, A., & Ovtcharova, J. (2022). A subject-oriented reference model for digital twins. *Computers & Industrial Engineering*, *172*, Article 108556.

Brown, S., & Rose, J. (1996). FPGA and CPLD architectures: a tutorial. *IEEE Design & Test of Computers*, *13*(2), 42–57. http://dx.doi.org/10.1109/54.500200.

Darba, A., De Belie, F., Vyncke, T., & Melkebeek, J. (2012). FPGA-based real-time simulation of sensorless control of PMSM drive at standstill. In *International symposium on power electronics power electronics, electrical drives, automation and motion* (pp. 1063–1068). IEEE.

DNV GL (2020). Qualification and assurance of digital twins. In *Recommended practice DNVGL-RP-A204* (October 2020 ed.).

GL 070 (2020). *Guidelines for application of IEC 61508 and IEC 61511 in the Norwegian petroleum industry*.

Glaessgen, E., & Stargel, D. (2012). The digital twin paradigm for future NASA and US air force vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA* (p. 1818).

Grieves, M., & Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems* (pp. 85–113). Springer.

IEC 61508 (2010). *Functional safety of electrical/electronic/programmable electronic safety-related systems*. Geneva: International Electrotechnical Commission.

IEEE (1987). *IEEE standard VHDL language reference manual*. IEEE.

Jiang, S., Liang, J., Liu, Y., Yamazaki, K., & Fujishima, M. (2005). Modeling and cosimulation of FPGA-based SVPWM control for PMSM. In *31st Annual conference of IEEE industrial electronics society, 2005* (p. 6). IEEE.

Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, *29*, 36–52.

Kapinski, J., Deshmukh, J. V., Jin, X., Ito, H., & Butts, K. (2016). Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems Magazine*, *36*(6), 45–64.

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022.

Kuon, I., & Rose, J. (2006). Measuring the gap between FPGAs and ASICs. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on field programmable gate arrays* (pp. 21–30).

Liu, M., Fang, S., Dong, H., & Xu, C. (2021). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, *58*, 346–361.

Mahler, C., Glaser, M., Schoch, S., Marx, S., Schluenss, S., Winter, T., Popp, J., & Imle, S. (2019). Safety capability of an all-electric production system. In *Offshore technology conference*. OnePetro.

Mashaly, M. (2021). Connecting the twins: A review on digital twin technology & its networking requirements. *Procedia Computer Science*, *184*, 299–305.

Matar, M., & Iravani, R. (2009). FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients. *IEEE Transactions on Power Delivery*, *25*(2), 852–860.

Milton, M., Benigni, A., & Monti, A. (2019). Real-time multi-FPGA simulation of energy conversion systems. *IEEE Transactions on Energy Conversion*, *34*(4), 2198–2208.

Milton, M., De La O, C., Ginn, H. L., & Benigni, A. (2020). Controller-embeddable probabilistic real-time digital twins for power electronic converter diagnostics. *IEEE Transactions on Power Electronics*, *35*(9), 9850–9864.

Monmasson, E., & Cirstea, M. N. (2007). FPGA design methodology for industrial control systems—A review. *IEEE Transactions on Industrial Electronics*, *54*(4), 1824–1842.

Monmasson, E., Idkhajine, L., & Naouar, M. W. (2011). FPGA-based controllers. *IEEE Industrial Electronics Magazine*, *5*(1), 14–26. http://dx.doi.org/10.1109/MIE.2011.940250.

Open Verilog International (1993). *Verilog hardware description reference*. Open Verilog International.

Pedroni, V. A. (2010). *Circuit design and simulation with VHDL*. MIT Press.

Rasheed, A., San, O., & Kvamsdal, T. (2020). Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, *8*, 21980–22012. http://dx.doi.org/10.1109/ACCESS.2020.2970143.

Ruba, M., Hunor, N., Hedesiu, H., & Martis, C. (2016). FPGA based processor in the loop analysis of variable reluctance machine with speed control. In *2016 IEEE international conference on automation, quality and testing, robotics* (pp. 1–6). http://dx.doi.org/10.1109/AQTR.2016.7501375.

Ruba, M., Nemes, R. O., Ciornei, S. M., Martis, C., Bouscayrol, A., & Hedesiu, H. (2019). Digital twin real-time FPGA implementation for light electric vehicle propulsion system using EMR organization. In *2019 IEEE vehicle power and propulsion conference* (pp. 1–6). http://dx.doi.org/10.1109/VPPC46532.2019.8952428.

Singh, M., Fuenmayor, E., Hinchy, E. P., Qiao, Y., Murray, N., & Devine, D. (2021). Digital twin: Origin to future. *Applied System Innovation*, *4*(2), 36.

Souri, A., & Norouzi, M. (2019). A state-of-the-art survey on formal verification of the internet of things applications. *Journal of Service Science Research*, *11*(1), 47–67.

Tao, F., Sui, F., Liu, A., Qi, Q., Zhang, M., Song, B., Guo, Z., Lu, S. C. Y., & Nee, A. Y. (2019). Digital twin-driven product design framework. *International Journal of Production Research*, *57*(12), 3935–3953.

Xiong, J., Ye, H., Pei, W., Kong, L., Huo, Q., & Han, Y. (2022). A monitoring and diagnostics method based on FPGA-digital twin for power electronic transformer. *Electric Power Systems Research*, *210*, Article 108111.

Yankova, Y., Bertels, K., Vassiliadis, S., Meeuws, R., & Virginia, A. (2007). Automated HDL generation: Comparative evaluation. In *2007 IEEE international symposium on circuits and systems* (pp. 2750–2753). http://dx.doi.org/10.1109/ISCAS.2007.378622.

Zkadek, P., Koczor, A., Gołek, M., Matoga, Ł., & Penkala, P. (2015). Improving efficiency of FPGA-in-the-loop verification environment. *IFAC-PapersOnLine*, *48*(4), 180–185.

Zumel, P., García-Valderas, M., Lázaro, A., López-Ongil, C., & Barrado, A. (2010). Co-simulation PSIM-ModelSim oriented to digitally controlled switching power converters. In *2010 IEEE 12th workshop on control and modeling for power electronics* (pp. 1–7). http://dx.doi.org/10.1109/COMPEL.2010.5562420.