Towards Monitoring and Diagnosis of Quantum Digital Twins

Martin Sachenbacher [1] and Martin Leucker [2]

Abstract: As access to larger-scale quantum computers is still limited, it is useful to simulate quantum algorithms on specialized programmable logic hardware such as FPGAs to exploit analogies between quantum and digital circuits, and allow for analysis and efficient debugging of quantum programs. Such simulations can be viewed as digital representations of actual physical systems, commonly referred to as Digital Twins (DTs). In previous work, we applied formal methods from runtime verification to synthesize monitors capable of real-time analysis of conventional DTs. Specifically, these monitors supervise alignment with their physical counterparts and support model-based diagnosis to identify potential root causes of observed deviations. In this short paper, we present ongoing work aimed at adapting and extending our approach to hardware-based simulation and analysis of quantum software on FPGAs, with the objective of improving both accuracy and resilience.

Keywords: quantum simulation, digital twins, formal methods, model-based diagnosis

1 Introduction

In the current noisy intermediate-scale quantum (NISQ) era, real quantum computers are still rarely available and expensive. Therefore, accurate and efficient simulation and simulation-based analysis of quantum programs is an important topic. Quantum computing simulations allow to study the properties of quantum circuits, debug quantum programs, and develop novel quantum computing algorithms.

Digital Twins (DTs) [FGL24] are a concept in engineering where a physical asset is mirrored in a virtual system composed of adaptive (simulation) models, regularly updated with measured data. DTs have become an essential concept for monitoring and controlling complex physical systems across various industries, enabling task such as what-if analysis, predictive maintenance, and operational optimization. An important challenge in DTs is the synchronization problem [TM24]: ensuring that the digital representation accurately mirrors the physical part continuously, as any deviation between the digital and physical systems can negatively impact the prediction accuracy and the effectiveness of control decisions.

We have previously worked on DTs in a variety of domains, including energy optimization [Th23], rescue missions [LSV23], and multi-core embedded systems [Le23]. In the latter case, due to the time-critical nature of the application, we employed specialized programmable logic hardware (FPGAs) to perform live reconstruction and analysis.

[1] OTH Regensburg, Faculty of Computer Science and Mathematics, Prüfeninger Str. 58, 93049 Regensburg, Germany, martin.sachenbacher@oth-regensburg.de, https://orcid.org/0000-0002-5418-1885

[2] Universität zu Lübeck, Institute for Software Engineering and Programming Languages, Ratzeburger Allee 160, Germany, leucker@isp.uni-luebeck.de, https://orcid.org/0000-0002-3696-9222

In this context, our contributions spanned several key areas. First, we pursued a formalization of DTs that enables the application of formal methods from computer science to verify important properties of the digital representation. Specifically, stream-based runtime verification allows the specification of properties in temporal logic and the automatic generation of monitors to check them during system execution. These monitors can even be compiled to run directly on FPGAs, enabling highly parallel and efficient monitoring. Second, we demonstrated how techniques from model-based diagnosis can be used to identify root causes of discrepancies between a DT and its physical counterpart [Ho24], thereby improving the reliability and practical applicability of DTs.

In this paper, we aim to adapt and apply these methods to the domain of quantum computing. First, FPGAs present an attractive platform for the simulation of quantum systems, as their capacity for efficient parallelism, low latency, and hardware-level customization makes them well-suited for accelerating the computationally intensive task of quantum simulation. Second, hardware-based simulation of quantum programs can be augmented with formal runtime verification and diagnosis – approaches we have already applied successfully in classical computing contexts. In the following, we outline this approach and describe the initial steps we have taken to adapt and implement it for quantum software.

2 Quantum simulation and hardware acceleration

As access to quantum computing platforms is still limited, application-specific simulators allow to explore and validate the practicality of quantum programs in real-world settings and test them within the limits of the simulator's capabilities. Although other specialized hardware such as GPUs is used in quantum simulation, FPGAs appear as one of the desirable platforms. FPGAs comprise programmable logic blocks ("fabric") that can be interconnected to perform parallel processing, allowing each logic block to perform a specific task simultaneously. This also allows to exploit analogies between quantum and digital circuits. Several approaches for simulating quantum computers using FPGAs are described in the literature. In fact, hardware-based quantum circuit emulation using FPGAs has been successfully applied to simulate Deutsch's algorithm [PD19] or quantum machine learning [Su23]. In our work, we build on the relatively cheap AMD/Xilinx Ultrascale Kria System-on-Module boards[3]. The modules' integrated ARM processors allow to run also hybrids of quantum and classical programs.

3 Runtime verification and monitoring of programs

Runtime verification (RV) [Ba18; LS09] is a research area of computer science dealing with analyzing the behavior of programs. In its simplest form, it checks whether a system execution complies with a formally specified correctness property. Typically, given such a

[3] https://www.amd.com/de/products/system-on-modules/kria.html

property φ, a monitor is synthesized and used to observe the behavior of programs. The data stream of interest is transmitted to the monitor, producing a sequence of events called traces. The monitor compares this event trace against the specification φ and can report violation or validation of the desired specification. Its verdict may be reported on-the-fly, during the execution of the monitored program. Other than testing, RV does not actively generate inputs to drive the system. Compared to model checking, which is concerned with mathematically proving properties that must hold in all execution of the system, RV is more lightweight and only concerned with concrete traces of the running system. Research in RV has produced a variety of monitors that can be automatically generated from specifications φ formulated in variants of linear time temporal logic (LTL), and corresponding higher-level specification languages such as TeSSLa [Ka22].

The instrumentation and monitors run simultaneously to the execution of the system under test and might therefore distort the behavior of the application to some extent, as additional code (even if it only outputs variables) affects the behavior of the system. However, using FPGAs it is possible to synthesize both the simulation of a quantum algorithm and a monitor that observes the quantum computation running in parallel on the same hardware. This setup enables the inspection of intermediate states of the quantum algorithm without interfering with the algorithm itself—as would typically happen on a conventional computer—and, more importantly, without disrupting the computation through measurement, as is the case on a real quantum computer. In other words, our approach enables a form of non-intrusive monitoring that is not feasible on actual quantum hardware.

4 Model-based fault diagnosis

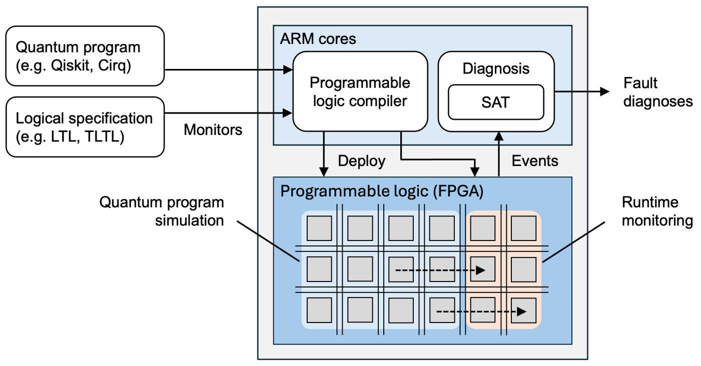
Model-based diagnosis (MBD) [dK03; Mo92] and the related field of fault detection, identification and reconfiguration (FDIR) focuses on identifying the root causes of system failures when deviations from expected behavior occur. MBD extends and complements runtime verification by not just detecting incorrect system behavior, but also localizing faults and determining their possible nature. It is based on formal models of the system's component behavior and interconnections to hypothesize (a minimal number of) faults in the system that would best explain the deviations between the values predicted by the model and the actual observations. Determining these causes is essential for formulating effective mitigation strategies. Diagnostic capabilities thus play also a crucial role in enhancing system resilience, which is the intrinsic capability of a system to maintain its required functionality when impacted by anticipated and unanticipated contingencies that may not have been explicitly considered during system design.

To date, a number of approaches have been proposed for bringing together RV and diagnosis. In particular, monitors that augment a system to check desired properties on actual trace data and report violation or validation at run-time can be used as a form of observations about the system. For example, [BLS06] proposed a framework for runtime reflection based on this idea, computing conflicts and diagnoses from a model of component dependencies.

In [Ho24] we built on this framework to develop an algorithm that diagnoses discrepancies between a DT and its physical counterpart. In this approach, a model of the underlying system and potential faults is encoded in propositional logic, and failure-identifying monitors assign truth values to some of the corresponding variables. SAT solving is then employed to derive a diagnosis that explains the observed behavior. These computations can be readily implemented on conventional CPUs. By selecting Kria boards as our testbed, we can efficiently run quantum algorithms and their associated monitors on the FPGA, while simultaneously performing SAT solving on the onboard ARM processor.

5 Current work of integration and prototyping

Fig. 1: Runtime monitoring and diagnosis of simulated quantum programs on FPGA-based platform



In a first step, a hybrid quantum-classical program will be simulated on the FPGA-based platform. This allows to check predefined correctness properties on the program; for this purpose, monitors are synthesized from formal, temporal logic specifications that indicate rule violations, and deployed on the simulated quantum system. In a second step, the possible causes of violations – such as faulty model components or parameters – are identified. For this purpose, our diagnosis algorithm for classical DTs [Ho24] will be adapted to a simplified hybrid quantum model for industrial manufacturing.

The prototype setup (Fig. 1) allows to execute simulated quantum programs and monitor their runtime behavior for debugging purposes. To demonstrate automatic self-diagnosis for typical fault scenarios (especially deviations between the physical and virtual model), we use algorithms for quantum optimization in process control from the TAQO-PAM project[4].

In the longer run, we aim to transfer the approach to an actual quantum computer. For this, the existing monitoring and diagnosis concepts must be adapted to quantum-specific challenges such as probabilistic and limited observability. One promising approach in this context is the use of so-called anticipatory monitoring techniques [Hi24].

[4] https://www.quantentechnologien.de/forschung/foerderung/anwendungsnetzwerk-fuer-das-quantencomputing/taqo-pam.html

References

[Ba18] Bartocci, E. et al.: Introduction to Runtime Verification. In (Bartocci, E.; Falcone, Y., eds.): Lectures on Runtime Verification: Introductory and Advanced Topics. Springer International Publishing, Cham, pp. 1–33, 2018, https://doi.org/10.1007/978-3-319-75632-5_1.

[BLS06] Bauer, A.; Leucker, M.; Schallhart, C.: Model-based runtime analysis of distributed reactive systems. In: Proceedings of the Australian Software Engineering Conference (ASWEC'06). IEEE, IEEE, 243–252, 2006.

[dK03] de Kleer, J.; Kurien, J.: Fundamentals of model-based diagnosis. IFAC Proceedings Volumes 36 (5), 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes 2003, Washington DC, 9-11 June 1997, pp. 25–36, 2003.

[FGL24] Fitzgerald, J.; Gomes, C.; Larsen, P., eds.: The Engineering of Digital Twins. Springer, Cham, 2024.

[Hi24] Hipler, R. et al.: General Anticipatory Runtime Verification. In: 36th International Conference on Computer Aided Verification (CAV). Springer Cham, Springer Cham, Montreal, Canada, 2024.

[Ho24] Hosseinkhani, E. et al.: A Model-Based Approach for Monitoring and Diagnosing Digital Twin Discrepancies. In (Pill, I.; Natan, A.; Wotawa, F., eds.): 35th International Conference on Principles of Diagnosis and Resilient Systems, DX 2024, November 4-7, 2024, Vienna, Austria. Vol. 125. OASIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2:1–2:15, 2024, https://doi.org/10.4230/OASIcs.DX.2024.2.

[Ka22] Kallwies, H. et al.: TeSSLa – An Ecosystem for Runtime Verification. In (Dang, T.; Stolz, V., eds.): Runtime Verification. Springer International Publishing, Cham, pp. 314–324, 2022.

[Le23] Leucker, M. et al.: Non-intrusive, Continuous Trace-based Monitoring and its Applications for System Correctness, Safety, and Resilience. In: Proc. Workshop on Principles of Diagnosis (DX). Proc. 34th International Workshop on Principles of Diagnosis (DX), 2023.

[LS09] Leucker, M.; Schallhart, C.: A brief account of runtime verification. The Journal of Logic and Algebraic Programming 78 (5), The 1st Workshop on Formal Languages and Analysis of Contract-Oriented Software (FLACOSâ'07), pp. 293–303, 2009, https://www.sciencedirect.com/science/article/pii/S1567832608000775.

[LSV23] Leucker, M.; Sachenbacher, M.; Vosteen, L. B.: Digital Twin for Rescue Missions - a Case Study. In (Hallerstede, S.; Kamburjan, E., eds.): Proceedings of Workshop on Applications of Formal Methods and Digital Twins (FMDT), Lübeck, Germany, 2023. Vol. 3507. CEUR Workshop Proceedings, CEUR-WS.org, 2023.

[Mo92] Mozetič, I.: Model-based diagnosis: An overview. In (Mřrík, V.; Štěpánková, O.; Trappl, R., eds.): Advanced Topics in Artificial Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 419–430, 1992.

[PD19] Pilch, J.; Długopolski, J.: An FPGA-based real quantum computer emulator. Journal of Computational Electronics 18 (1), pp. 329–342, 2019, https://doi.org/10.1007/s10825-018-1287-5.

[Su23] Suzuki, T. et al.: Quantum AI simulator using a hybrid CPU–FPGA approach. Scientific Reports 13 (1), p. 7735, 2023, https://doi.org/10.1038/s41598-023-34600-2.

[Th23] Thoma, D. et al.: A Digital Twin for Coupling Mobility and Energy Optimization: The ReNuBiL Living Lab. In (Hallerstede, S.; Kamburjan, E., eds.): Proceedings of Workshop on Applications of Formal Methods and Digital Twins (FMDT), Lübeck, Germany, 2023. Vol. 3507. CEUR Workshop Proceedings, CEUR-WS.org, 2023.

[TM24] Tan, B.; Matta, A.: The digital twin synchronization problem: Framework, formulations, and analysis. IISE Transactions 56 (6), pp. 652–665, 2024, eprint: https://doi.org/10.1080/24725854.2023.2253869, https://doi.org/10.1080/24725854.2023.2253869.