Full-stack Runtime Reconfiguration for Efficient Data Processing

Mario Porrmann,[‡] Marcelo Pasin,[x] Pedro Trancoso,[§] Jens Hagemeyer*
[‡]Osnabrück University, Germany — [x]University of Neuchâtel, Switzerland
[§]Chalmers University of Technology, Sweden — *Bielefeld University, Germany

***Abstract*—This document explores the concept of runtime reconfiguration at all hardware and software levels as a means to improve efficiency, sustainability, and dependability in future data-centric computing systems. We discuss the motivation behind this approach highlighting opportunities, the state-of-the-art, and the research challenges that must be addressed to fully realize its potential. This document serves as an expression of interest (EoI), providing a vision of future research directions in the area of *Cloud-to-Edge-to-IoT for European Data*, shaping the research agenda for 2025-2027.**

I. Motivation

In recent years, the need for adaptable and efficient computing systems has become increasingly apparent due to the growing complexity and diversity of both compute resources and application requirements. At the same time, data is being produced at a much larger scale than before and data movement is clearly one of the dominant performance bottlenecks in current systems [1]. Reconfigurability or adaptability is a powerful tool that allows systems to adapt their architecture to specific applications while being aware of the data locality, thereby improving efficiency and performance [2]. Such reconfiguration mechanisms can be applied over the full stack of the architecture, from the physical to the application level, providing a generic means of optimizing data processing, data locality and communication. By adapting the different architecture layers to changing application requirements in a holistic manner, these systems offer several benefits, including better hardware utilization, improved performance and efficiency, and increased sustainability and dependability, as systems can not only use the available hardware most efficiently, but also increase their overall lifetime.

Applying reconfiguration at runtime over the full architecture stack is defined as full-stack runtime reconfiguration, allowing to better adapt to dynamic changing application demands, which is crucial for achieving optimal performance and energy efficiency in various applications. Reconfigurable systems can improve performance by dynamically changing their architecture to suit the application's requirements, allowing for a more efficient use of resources and reducing the energy consumption of the system [3], [4]. Furthermore, runtime reconfiguration can lead to improved sustainability by allowing systems to adapt to new technologies and standards without the need for complete hardware replacement [5]. Figure 1 shows an abstract representation of a heterogeneous, distributed architecture, changing over time by means of runtime reconfiguration.

II. Current Status

Reconfigurability and adaptability are already available across multiple layers in existing systems. At the physical layer, traditionally, LUT-based cells are the basis for most of today's Field-Programmable Gate Arrays (FPGAs). With the acquisition of Xilinx by AMD, similar to the purchase of Altera by Intel in 2015, there is a strong trend for FPGA fabrics to become part of next-generation CPUs and SoCs, strengthening the general availability of reconfigurable logic

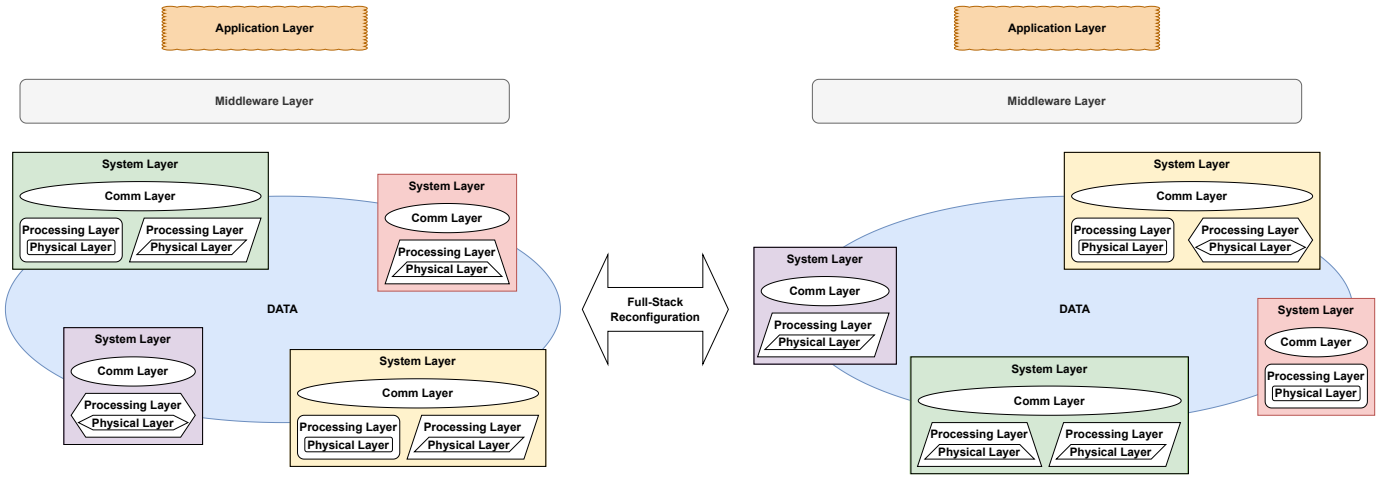

Fig. 1. Full-stack reconfiguration, showing an abstract representation of the different architecture layers over time in a distributed system. Different geometrical forms show different implementations for the respective architectural layers. Different colors represent different components of the distributed application.

in widely-used computing devices, similar to the GPUs today. Apart from traditional LUT-based cells, scientists also focus on new emerging technologies to decrease the inevitable overhead of reconfigurability, resulting in novel cell architectures, ranging from reconfigurable transistors (RFETS), devices that can be configured between an n-channel and p-channel behavior [6] to memristor-based LUT cells [7]. At the processing layer, runtime reconfiguration of FPGAs or Coarse-Grain Architectures has been demonstrated, further increasing the flexibility and efficiency of the designs [8], [9]. Large vendors offer today heterogeneous System-on-Chip (SoC) solutions that combine the flexibility of embedded or server-grade processors with closely-coupled reconfigurable accelerators, but their efficient use still suffers from proprietary development tools, high design time and the required expert knowledge. In parallel, reconfigurable processor architectures are developed, recently driven by the open and extendable RISC-V ISA [10]. On the communication layer, dataflow computing and data-centric computing (e.g., processing-in-memory [11]) enable efficient data processing and communication by focusing on the movement of data and its transformation. Modern operating systems can adapt themselves to dynamic hardware, taking into account runtime additions or subtractions of hot-pluggable (or virtualised) RAM, CPU cores, hard disk, network interfaces, etc. Finally, existing distributed middleware can enable context-aware computing in the entire IoT-edge-cloud continuum, adapting the computations deployments and behaviors to their environment, improving overall system performance, security, fault-tolerance, or even energy-efficiency.

Despite the advancements previously cited, reconfiguration has currently limited use. It is often used as a startup feature but not exploited at runtime, which hinders its full potential. Additionally, reconfiguration is mostly explicitly controlled by the application or the user, without the support of the system's abstraction layers that could streamline the whole process and enable a more efficient system operation. Finally, reconfigurability is mostly handled in an isolated manner on each layer, rather than being integrated across the entire stack from application to hardware.

III. Research Challenges

In order to fully exploit the potential of full-stack runtime reconfiguration, several research challenges must be addressed. The development of a comprehensive full-stack runtime reconfiguration solution requires the integration of reconfigurability across all architectural levels, from transistors to the application level. A co-design methodology is essential for achieving such integration, as it involves the collaboration between hardware and software designers to create efficient and adaptable systems. Runtime reconfiguration needs to be integrated into the high-level, software-centric development process to reduce time-to-market as well as the required skill set of the developer team. This implies new domain-specific languages and tools that account for the overhead in terms of reconfiguration time, resources and energy and in turn are able to optimize resource efficiency at the application level.

Dependability aspects, such as fault tolerance and robustness, should be considered when designing reconfigurable systems, as they contribute to overall system reliability and longevity. One potential approach to address these challenges is to develop a *reconfigurability-aware* middleware and runtime system, including abstraction layers and auto-reconfigurability features that can simplify the reconfiguration process and enable more efficient use of resources. Reconfiguration would then be applied anytime deemed necessary, automatically, without having to restart the system or the application. Additionally, porting and demonstrating the effectiveness of runtime reconfigurability on the application level is crucial to showcasing the benefits and motivating further research and adoption of such technology.

IV. Conclusion

In conclusion, full-stack runtime reconfiguration holds great promise for improving efficiency, sustainability, and dependability in modern computing systems. By addressing the research challenges outlined in this document, it is possible to create a new generation of adaptable, efficient, and dependable systems that can respond to the ever-changing demands of modern applications.

References

[1] Amirali Boroumand, Saugata Ghose, et al. Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. In *ACM SIGPLAN Notices*, ASPLOS '18, page 316–331, New York, NY, USA, 2018. ACM.

[2] Katherine Compton and Scott Hauck. Reconfigurable Computing: A Survey of Systems and Software. *ACM Comput. Surv.*, 34(2):171–210, jun 2002.

[3] Shaoshan Liu, Richard Neil Pittman, Alessandro Forin, and Jean-Luc Gaudiot. Achieving Energy Efficiency through Runtime Partial Reconfiguration on Reconfigurable Systems. *ACM Trans. Embed. Comput. Syst.*, 12(3), apr 2013.

[4] Markus Koester, Wayne Luk, Jens Hagemeyer, Mario Porrmann, and Ulrich Ruckert. Design Optimizations for Tiled Partially Reconfigurable Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(6):1048–1061, 2011.

[5] Christophe Bobda, Joel Mandebi Mbongue, et al. The Future of FPGA Acceleration in Datacenters and the Cloud. *ACM Trans. Reconfigurable Technol. Syst.*, 15(3), feb 2022.

[6] Thomas Mikolajick, André Heinzig, Jens Trommer, Tim Baldauf, and Walter Weber. The RFET - A reconfigurable nanowire transistor and its application to novel electronic circuits and systems. *Semiconductor Science and Technology*, 32, 12 2016.

[7] Shubham Rai, Pallab Nath, Ansh Rupani, Santosh Kumar Vishvakarma, and Akash Kumar. A Survey of FPGA Logic Cell Designs in the Light of Emerging Technologies. *IEEE Access*, 9:91564–91574, 2021.

[8] Stephen M. Trimberger. Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology. *Proceedings of the IEEE*, 103(3):318–331, 2015.

[9] R. Hartenstein. A decade of reconfigurable computing: a visionary retrospective. In *Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001*, pages 642–649, 2001.

[10] Nguyen Dao, Andrew Attwood, Bea Healy, and Dirk Koch. FlexBex: A RISC-V with a Reconfigurable Instruction Extension. In *2020 International Conference on Field-Programmable Technology (ICFPT)*, pages 190–195, 2020.

[11] Kazi Asifuzzaman, Narasinga Rao Miniskar, Aaron R. Young, Frank Liu, and Jeffrey S. Vetter. A survey on processing-in-memory techniques: Advances and challenges. *Memories - Materials, Devices, Circuits and Systems*, 4:100022, 2023.